Oracle Sales Cloud
Extending Sales
This guide also applies to on-premise implementations

Release 8

April 2014
Contents

1 What's New in Sales Cloud Extensibility?
   What's New in This Guide? ................................................................. 1-1

2 Oracle Sales Cloud Extensibility: Introduction
   Oracle Sales Cloud Extensibility: Overview .................................... 2-1

3 Customizing and Extending Oracle Sales Cloud
   Customizing and Extending Oracle Sales Cloud Applications: Overview 3-1
   Personalization: Explained ............................................................... 3-2
   Run Time Customizations and Extensions: Explained ......................... 3-2
   Customization Layers: Explained .................................................... 3-4
   Customization Layers: Examples .................................................. 3-5
   Selecting Customization Layers to Include: Examples ......................... 3-7
   What You Can Customize and Extend and with Which Tool: Explained .... 3-8
   Understanding the Customization Development Life Cycle .................. 3-12
   Customizing Navigation ................................................................. 3-17
   Customizing Help ........................................................................... 3-22

4 Application Composer: Using Sandboxes
   Using Sandboxes: Overview ............................................................ 4-1
   Using Sandboxes: Points to Consider .............................................. 4-1
   Sandboxes: Explained .................................................................. 4-4
   Managing Customizations Using Sandboxes: Explained ..................... 4-5
   Sandboxes: How They Work with Some Customizations and Features ..... 4-6
   Sandbox Development Lifecycle Components: Explained ................... 4-7
   Setting Up Sandboxes: Procedures ................................................ 4-11
   Using the Sandbox Manager: Explained .......................................... 4-12
   Supported Sandbox Manager Operations: Explained ......................... 4-14
   Multiple Sandbox User Conflicts: Explained ..................................... 4-15
   Publishing Sandboxes: Procedures ................................................ 4-16
   FAQs for Using Sandboxes .............................................................. 4-16

5 Application Composer: Using the Application Composer
   Using Application Composer: Overview .......................................... 5-1
   Extending Oracle Sales Cloud: How It Works .................................... 5-1
### 6 Application Composer: Extending Simplified Pages

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extending Simplified Pages: Overview</td>
<td>6-1</td>
</tr>
<tr>
<td>Oracle Sales Cloud Simplified Pages: Explained</td>
<td>6-1</td>
</tr>
<tr>
<td>Customizing the Springboard for Simplified Pages: Points to Consider</td>
<td>6-4</td>
</tr>
<tr>
<td>Customizing Simplified Pages Using Page Composer: Points to Consider</td>
<td>6-4</td>
</tr>
<tr>
<td>Changing the Look and Feel of Simplified Pages: Points to Consider</td>
<td>6-6</td>
</tr>
<tr>
<td>Creating and Adding Custom Action Links to Simplified Pages: Worked Example</td>
<td>6-6</td>
</tr>
<tr>
<td>Creating and Reordering Subtabs in Simplified Details Pages: Worked Example</td>
<td>6-11</td>
</tr>
<tr>
<td>Extending Simplified Pages for Assets: Explained</td>
<td>6-16</td>
</tr>
<tr>
<td>Extending Simplified Pages for Assets Using Application Composer: Worked Example</td>
<td>6-17</td>
</tr>
<tr>
<td>Extending Simplified Pages for Contacts: Explained</td>
<td>6-20</td>
</tr>
<tr>
<td>Extending Simplified Pages for Customers: Explained</td>
<td>6-22</td>
</tr>
<tr>
<td>Extending Simplified Pages for Leads: Explained</td>
<td>6-27</td>
</tr>
<tr>
<td>Extending Simplified Pages for Leads Using Application Composer: Worked Example</td>
<td>6-29</td>
</tr>
<tr>
<td>Extending Simplified Pages for Opportunities: Explained</td>
<td>6-34</td>
</tr>
</tbody>
</table>
Extending Simplified Pages for Contacts: Explained .............................................. 11-8

12 Oracle Sales Cloud Account, Contact, and Household Extensibility

Oracle Sales Cloud Account and Contact Extensibility: Overview ....................... 12-1
Customizing Oracle Sales Cloud Customer Center Pages: Explained ....................... 12-1
Customizing Oracle Sales Cloud Customer Center Using Page Composer: Explained ...
Customizing Oracle Sales Cloud Customer Center Using Page Composer: Worked Example .......................................................... 12-7
Extending Simplified Pages for Customers: Explained ............................................ 12-8

13 Oracle Sales Cloud Sales and Opportunity Management Extensibility

Oracle Sales Cloud Sales and Oracle Sales Cloud Opportunity Management Extensibility: Overview .............................................................. 13-1
Customizing Sales Pages Using Application Composer: Explained ....................... 13-1
Customizing Opportunity Pages Using Application Composer: Explained ............. 13-3
Customizing the Dashboard and Pages in Oracle Sales Cloud Sales Using Page Composer: Explained ........................................ 13-8
Hiding and Reordering Opportunity Fields Using Page Composer: Worked Example ................................................................. 13-13
Customizing the Sales Dashboard Using Page Composer: Worked Example ........... 13-17
Creating Opportunity Saved Searches Using Page Composer: Worked Example .... 13-19
Extending Simplified Pages for Opportunities: Explained ..................................... 13-21
Extending Simplified Pages for Opportunities using Page Composer: Worked Example ................................................................. 13-23

14 Oracle Sales Cloud Sales Forecasting Extensibility

Oracle Sales Cloud Sales Forecasting Extensibility: Overview ......................... 14-1
Customizing Oracle Sales Cloud Sales Forecasting Pages: Explained ................ 14-1

15 Oracle Sales Cloud Marketing Extensibility

Oracle Sales Cloud Marketing Extensibility: Overview ........................................ 15-1
Customizing Marketing Campaign Pages Using Application Composer: Explained ...
Customizing Marketing Campaign Stage Pages Using Application Composer: Explained ...
Customizing Marketing Campaign Member Pages Using Application Composer: Explained ...
Customizing Marketing List Pages Using Application Composer: Explained ...
Customizing Marketing Treatment Pages Using Application Composer: Explained ...
Customizing Marketing Response Pages Using Application Composer: Explained ...
Customizing Marketing Budget Pages Using Application Composer: Explained .... 15-12
16 Oracle Sales Cloud Quota Management Extensibility
Oracle Sales Cloud Quota Management Extensibility: Overview ........................................... 16-1
Customizing Oracle Sales Cloud Quota Management: Explained ........................................ 16-1

17 Oracle Sales Cloud Partner Management Extensibility
Oracle Fusion Partner Relationship Management Extensibility: Overview ......................... 17-1
Customizing Oracle Sales Cloud Partner Pages Using Application Composer: Explained .....
Customizing Partner Management Dashboard and Pages Using Page Composer: Explained
Customizing Partner Relationship Management Using Page Composer: Worked Example 

18 Oracle Sales Cloud Mobile Sales Extensibility
Oracle Sales Cloud Mobile Sales Extensibility: Overview ..................................................... 18-1
Oracle Sales Cloud Mobile Extensibility: Explained ............................................................. 18-1
Customizing Oracle Sales Cloud Mobile Sales: Worked Example .................................. 18-2
How can I add an Oracle Business Intelligence report to Oracle Sales Cloud Mobile Sales? ....................................................... 18-2
Testing Oracle Sales Cloud Mobile Customizations Using a Sandbox: Worked Example 

19 Exporting and Importing Customizations
Exporting and Importing Customizations: Overview ........................................................ 19-1
Exporting and Moving Customizations: Points to Consider ................................................... 19-1
Using Customization Migration to Move Customizations: Points to Consider ................. 19-3
Creating and Applying Customizations Using a Customization Set: Procedures .............. 19-5
Exporting and Importing Supported Customizations: Explained ...................................... 19-11
Supported Application Composer Customizations and Best Practices: Explained ................... 19-13

20 An End-to-End Configuration Example
End-to-End Configuration Examples: Overview ............................................................ 20-1
Creating a Sandbox: Example .....................................................................................20-1
Creating or Extending an Object: Example ................................................................20-2
Adding Child or Related Objects: Example .................................................................20-9
Advanced Field and Object Properties: Example .....................................................20-14
Adding Subtabs and Tree Nodes: Example ...............................................................20-21
Testing Custom Object Security Settings: Example .................................................20-23
Preface

This Preface introduces the guides, online help, and other information sources available to help you more effectively use Oracle Fusion Applications.

Oracle Fusion Applications Help

You can access Oracle Fusion Applications Help for the current page, section, activity, or task by clicking the help icon. The following figure depicts the help icon.

Note

If you don’t see any help icons on your page, then click the Show Help icon button in the global area. However, not all pages have help icons.

You can add custom help files to replace or supplement the provided content. Each release update includes new help content to ensure you have access to the latest information. Patching does not affect your custom help content.

Oracle Fusion Applications Guides

Oracle Fusion Applications guides are a structured collection of the help topics, examples, and FAQs from the help system packaged for easy download and offline reference, and sequenced to facilitate learning. To access the guides, go to any page in Oracle Fusion Applications Help and select Documentation Library from the Navigator menu.

Guides are designed for specific audiences:

- **User Guides** address the tasks in one or more business processes. They are intended for users who perform these tasks, and managers looking for an overview of the business processes. They are organized by the business process activities and tasks.

- **Implementation Guides** address the tasks required to set up an offering, or selected features of an offering. They are intended for implementors. They are organized to follow the task list sequence of the offerings, as displayed within the Setup and Maintenance work area provided by Oracle Fusion Functional Setup Manager.

- **Concept Guides** explain the key concepts and decisions for a specific area of functionality. They are intended for decision makers, such as chief
financial officers, financial analysts, and implementation consultants. They are organized by the logical flow of features and functions.

- **Security Reference Manuals** describe the predefined data that is included in the security reference implementation for one offering. They are intended for implementors, security administrators, and auditors. They are organized by role.

These guides cover specific business processes and offerings. Common areas are addressed in the guides listed in the following table.

<table>
<thead>
<tr>
<th>Guide</th>
<th>Intended Audience</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common User Guide</td>
<td>All users</td>
<td>Explains tasks performed by most users.</td>
</tr>
<tr>
<td>Common Implementation Guide</td>
<td>Implementors</td>
<td>Explains tasks within the Define Common Applications Configuration task list, which is included in all offerings.</td>
</tr>
<tr>
<td>Functional Setup Manager User Guide</td>
<td>Implementors</td>
<td>Explains how to use Oracle Fusion Functional Setup Manager to plan, manage, and track your implementation projects, migrate setup data, and validate implementations.</td>
</tr>
</tbody>
</table>
| Technical Guides                                | System administrators, application developers, and technical members of implementation teams | Explain how to install, patch, administer, and customize Oracle Fusion Applications.  
**Note**  
Limited content applicable to Oracle Cloud implementations. |


### Other Information Sources

**My Oracle Support**


Use the My Oracle Support Knowledge Browser to find documents for a product area. You can search for release-specific information, such as patches, alerts, white papers, and troubleshooting tips. Other services include health checks, guided lifecycle advice, and direct contact with industry experts through the My Oracle Support Community.
Oracle Enterprise Repository for Oracle Fusion Applications

Oracle Enterprise Repository for Oracle Fusion Applications provides details on service-oriented architecture assets to help you manage the lifecycle of your software from planning through implementation, testing, production, and changes.

In Oracle Fusion Applications, you can use Oracle Enterprise Repository at http://fusionappsoer.oracle.com for:

- Technical information about integrating with other applications, including services, operations, composites, events, and integration tables. The classification scheme shows the scenarios in which you use the assets, and includes diagrams, schematics, and links to other technical documentation.

- Other technical information such as reusable components, policies, architecture diagrams, and topology diagrams.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

Comments and Suggestions

Your comments are important to us. We encourage you to send us feedback about Oracle Fusion Applications Help and guides. Please send your suggestions to oracle_fusion_applications_help_ww_grp@oracle.com. You can use Send Feedback to Oracle from the Settings and Actions menu in Oracle Fusion Applications Help.
What's New in Sales Cloud Extensibility?

What's New in This Guide?

This topic provides information on what's new in this guide in this release. These changes include content that has been added or modified in this guide since the last release.

New Chapters in This Release

The following chapters have been added:

- Customizing and Extending Oracle Sales Cloud Applications
- Oracle Sales Cloud Quota Management Extensibility

New Sections in This Release

This table lists the sections that have been added in the relevant chapters.
<table>
<thead>
<tr>
<th>New Sections</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customizing and Extending Oracle Sales Cloud Applications: Overview</td>
<td>Customizing and Extending Oracle Sales Cloud Applications</td>
</tr>
<tr>
<td>Personalization: Explained</td>
<td></td>
</tr>
<tr>
<td>New Sections</td>
<td>Chapter</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Using Sandboxes: Points to Consider</td>
<td>Application Composer: Using Sandboxes</td>
</tr>
<tr>
<td>Creating a Sandbox: Example</td>
<td></td>
</tr>
<tr>
<td>Managing Customizations Using Sandboxes: Explained</td>
<td></td>
</tr>
<tr>
<td>Setting Up Sandboxes: Procedures</td>
<td></td>
</tr>
<tr>
<td>Publishing Sandboxes: Procedures</td>
<td></td>
</tr>
<tr>
<td>Creating or Extending an Object: Example</td>
<td>Application Composer: Using the Application Composer</td>
</tr>
<tr>
<td>Adding Child or Related Objects: Example</td>
<td></td>
</tr>
<tr>
<td>Adding Join Fields to the Sales Account Object: Worked Example</td>
<td></td>
</tr>
<tr>
<td>Record Type Fields: Explained</td>
<td></td>
</tr>
<tr>
<td>Advanced Field and Object Properties: Example</td>
<td></td>
</tr>
<tr>
<td>Adding Subtabs and Tree Nodes: Example</td>
<td></td>
</tr>
<tr>
<td>Making Saved Searches Available to All Users: Procedure</td>
<td></td>
</tr>
<tr>
<td>Testing Custom Object Security Settings: Example</td>
<td></td>
</tr>
<tr>
<td><strong>New Sections</strong></td>
<td><strong>Chapter</strong></td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Enabling or Disabling Drill Down Fields in Simplified Pages: Explained</td>
<td>Application Composer: Extending Simplified Pages</td>
</tr>
<tr>
<td>Customizing the Springboard for Simplified Pages: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Customizing Simplified Pages Using Page Composer: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Changing the Look and Feel of Simplified Pages: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Creating and Adding Custom Action Links to Simplified Pages: Worked Example</td>
<td></td>
</tr>
<tr>
<td>Creating and Reordering Subtabs in Simplified Details Pages: Worked Example</td>
<td></td>
</tr>
<tr>
<td>Extending Simplified Pages for Assets: Explained</td>
<td></td>
</tr>
<tr>
<td>Extending Simplified Pages for Assets Using Application Composer: Worked Example</td>
<td></td>
</tr>
<tr>
<td>Extending Simplified Pages for Sales Forecasts: Explained</td>
<td></td>
</tr>
<tr>
<td>Adding Drill Down Fields to a Subtab: Example</td>
<td></td>
</tr>
<tr>
<td>Making Custom Dynamic Choice List Fields Drillable: Worked Example</td>
<td></td>
</tr>
<tr>
<td>Dynamic Page Layouts: Explained</td>
<td></td>
</tr>
<tr>
<td>Page Layouts per Record Type: Explained</td>
<td></td>
</tr>
<tr>
<td>Page Layouts per Role: Explained</td>
<td></td>
</tr>
<tr>
<td>Page Layouts per Advanced Expression: Explained</td>
<td></td>
</tr>
<tr>
<td>How can I post announcements on the simplified home page?</td>
<td></td>
</tr>
<tr>
<td>Using Groovy Scripting: Example</td>
<td>Application Composer: Using Groovy Scripts</td>
</tr>
<tr>
<td>Configuring Custom Subject Areas for Segmentation: Points to Consider</td>
<td>Application Composer: Creating Custom Subject Areas</td>
</tr>
<tr>
<td>New Sections</td>
<td>Chapter</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Before You Customize Existing Pages: Points to Consider</td>
<td>Page Composer: Customizing Oracle Sales Cloud Services</td>
</tr>
<tr>
<td>Page Composer Views: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Objects That Appear on Multiple Pages: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Customizing the Same Page with Multiple Tools: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Changing the Layout of a Page: Explained</td>
<td></td>
</tr>
<tr>
<td>Page Component Properties: Explained</td>
<td></td>
</tr>
<tr>
<td>Working With Components in Page Customizations: Procedures</td>
<td></td>
</tr>
<tr>
<td>Customizing Popup Window Content: Procedure</td>
<td></td>
</tr>
<tr>
<td>Customizing the Global Page Template: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Pages Using Design Time Tools: Highlights</td>
<td></td>
</tr>
<tr>
<td>Customizing Task Lists: Procedure</td>
<td></td>
</tr>
<tr>
<td>Making Customizations Visible Based on User Roles: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Page Titles: Procedure</td>
<td></td>
</tr>
<tr>
<td>FAQs for Customizing Pages Using Page Composer</td>
<td></td>
</tr>
<tr>
<td>How do I back out or roll back customizations I made to a specific page</td>
<td></td>
</tr>
<tr>
<td>How can I work on customizations, but prevent users from seeing them until</td>
<td></td>
</tr>
<tr>
<td>the customizations are complete</td>
<td></td>
</tr>
<tr>
<td>What happens if my customizations or personalizations make the page</td>
<td></td>
</tr>
<tr>
<td>inaccessible</td>
<td></td>
</tr>
<tr>
<td>How can I reset a page or task flow to a previously saved version</td>
<td></td>
</tr>
<tr>
<td>How do I use metadata to perform customization-related tasks that cannot be</td>
<td></td>
</tr>
<tr>
<td>done in Page Composer</td>
<td></td>
</tr>
<tr>
<td>New Sections</td>
<td>Chapter</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Customizing Marketing Campaign Pages Using Application Composer: Explained</td>
<td>Oracle Sales Cloud Marketing Extensibility</td>
</tr>
<tr>
<td>Customizing Marketing Campaign Stage Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Campaign Member Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing List Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Treatment Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Response Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Budgets Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Budget Entry Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Fund Request Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Claim Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Marketing Claim Settlement Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Sales Lead Pages Using Application Composer: Explained</td>
<td></td>
</tr>
<tr>
<td>Customizing Oracle Sales Cloud Quota Management: Explained</td>
<td>Oracle Sales Cloud Quota Management Extensibility</td>
</tr>
<tr>
<td>Customizing Oracle Sales Cloud Partner Pages Using Application Composer: Explained</td>
<td>Oracle Sales Cloud Partner Management Extensibility</td>
</tr>
<tr>
<td>Exporting and Moving Customizations: Points to Consider</td>
<td>Exporting and Importing Customizations</td>
</tr>
<tr>
<td>Using Customization Migration to Move Customizations: Points to Consider</td>
<td></td>
</tr>
<tr>
<td>Creating and Applying Customizations Using a Customization Set: Procedures</td>
<td></td>
</tr>
</tbody>
</table>

**Changed Content in Existing Sections in This Release**

This table lists the existing sections where the content has been added or modified.
<table>
<thead>
<tr>
<th>Added or Modified Content</th>
<th>Section</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added: Ability to activate or inactivate a published custom subject area.</td>
<td>Creating and Editing Custom Subject Areas: Explained</td>
<td>Application Composer: Creating Custom Subject Areas</td>
</tr>
</tbody>
</table>
Welcome! You are reading this guide because you are interested in extending and customizing your Oracle Sales Cloud. This guide introduces you to the various tools that you can use to make application changes.

**Application Composer**

Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application developers, to customize and extend an Oracle Sales Cloud service. Make the type of data model changes which, for non-Sales Cloud, can only be made by application developers. For example, easily create a new object and related fields, then create new desktop 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 a Sales Cloud application, make your changes, and see most changes take immediate effect in real time, without having to sign back in to the application. Data model changes, such as the creation of custom fields, do require that you reauthenticate before you can see those changes.

**Note**

Application Composer is supported for use only in English.

To learn more about Application Composer, refer to all chapters in this guide whose titles are prefixed with "Application Composer". Also, review each product-specific chapter in this guide to read about which Application Composer tasks you can do in each Oracle Sales Cloud service.

**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 it 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 read about which Page Composer tasks you can do in each Oracle Sales Cloud service.

**BI Composer**

The Oracle Business Intelligence (BI) Composer is a tool that lets you 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 an Oracle Sales Cloud service, 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.
Customizing and Extending Oracle Sales Cloud Applications: Overview

Read this chapter to learn about how you customize and extend your application using the different tools available for customization. While Oracle Sales Cloud provides robust ready-to-use functionality, there may be areas of the applications that you must change to meet your business needs.

In this chapter, you will learn:

- 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
- What does the customization development lifecycle look like
- How you can customize navigation and help

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 frameworks provide a number of reporting capabilities. Identity management works at every level to control access. Each of these areas of an application can be customized and extended to suit your business needs. Additionally, Oracle 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 webbased 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.
Personalization: Explained

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

For composer-based personalizations, Oracle Fusion Applications includes Page Composer, which allows users to change certain UI pages to suit their needs. For example, they can rearrange certain objects on a page, and add or remove designated content.

Run Time Customizations and Extensions: Explained

Run time customizations and extensions include those that a business analyst can make to Oracle Fusion Applications at run time using browser-based composers and other tools. These customizations and extensions are visible and usable by all or by a subset of Oracle Fusion Applications users. The types of run time customizations and extensions range from changing the look and feel of a page, to customizing standard objects, adding a new object and associated pages and application functionality, changing workflows, defining security for new objects, and customizing reports.

Access to run time customization tools depends on your roles. If you are assigned a role with an administrative privilege, you can access most run time customization tasks.

Note

Some customization tools are available only for specific product families.

With run time customization tools, you can:

• Personalize and customize the UI
• Customize menus
• Create and customize objects
• Create and customize business flows for custom objects
• Add custom attributes to business objects using flexfields
• Customize reports and analytics
• Customize help

Personalizing and Customizing the UI

Both personalization and customization involve using Page Composer to make changes to an application page. For personalization, any user can drag and drop fields, rearrange regions, and add approved external content.
For customization, you also use:

- **Page Composer** to customize simplified and desktop pages for other users. You can add fields, add validation, change defaults, rearrange regions, add external content, and save queries. Page Composer allows you to work in a WYSIWYG view, and, in some cases, Source view.

- **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 functional area** to change the:
  - Look and feel of simplified UI.
  - Announcements on the simplified home page.

**Customizing Navigation**

Use the Manage Menu Customizations task in the Setup and Maintenance work area to customize the Navigator menu. You can also determine which dashboards to include on the home page (desktop UI only).

For simplified UI, use the Structure page in the Settings functional area to customize the springboard.

**Creating and Customizing Objects in Oracle Sales Cloud**

Application Composer lets you make more complex runtime customizations to Oracle Sales Cloud applications. In addition to customizing pages, you can customize objects and all the artifacts that support them (such as fields, pages, buttons and links, security, server scripts, and saved searches), and can also create completely new objects and artifacts. For more information on defining objects, see Oracle Sales Cloud: Extending Sales.

When new objects are created, you often also create associated work area pages for those objects. Using the Manage Menu Customizations task, you can add those pages to the Navigator menu so that they can be accessed in the same way as standard objects.

When you create a new object in Application Composer, you can define security policies for it. A security policy defines the end user’s level of access to the data records of the object. For more information about securing custom objects, see Oracle Sales Cloud: Extending Sales.

**Creating and Customizing Business Process Flows for Objects in Oracle Sales Cloud**

When you create a new object that is not a subclass of another object, you can also create a new object workflow to manage any business processes associated with it. For example, say you used Application Composer to create a marketing object and you want to create an associated approval flow. From within Application Composer, you can create the process that defines that flow by creating an object workflow, identifying the trigger conditions, and then setting up the Oracle Business Process Composer approval flow. You can also
create object workflows for standard objects. For more information on using object workflows, see Oracle Sales Cloud: Extending Sales.

Adding Custom Attributes to Business Components Using Flexfields

Most business components, except those in Oracle Sales Cloud products, support the use of flexfields to extend the object with custom attributes. Using a flexfield, you can 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 pre-reserved column in the application database. Roles with administrative privileges provide access for viewing, configuring, and deploying flexfields using tasks available in the Setup and Maintenance work area. The configuration of the flexfield is stored in the Oracle Metadata Services (MDS) repository and preserved across patches and upgrades.

Customizing Reports and Analytics

Oracle Fusion Applications comes with a complete set of reports. You can customize these reports (for example, change the layout) to fit your particular business needs. Additionally, if you customize or create a business object, you can create a new report for that object.

Customizing Help

If you have the appropriate job roles, then you can customize the help files in Oracle Fusion Applications Help. You can also determine which help files to show in specific help windows. Use the Manage Custom Help page to maintain both predefined and custom help files. You can open this page from any help window, or from the help site itself.

Aside from help files, you can also customize help that appear on the UI, for example hint text.

Customization Layers: Explained

Oracle Fusion Applications contains built-in customization layers that allow you to make customizations that affect only certain instances or users of an application. Before you create customizations, select the layer to which you want your customizations to be applied. Most of the tools that you use to create your customizations provide a dialog box where you can pick the layer for your customizations. You must be careful to choose the correct layer.

Available Layers

The exact customization layers available for an application depend on that application family. For information on product-specific customization layers, see assets with the Customization Layer type in Oracle Enterprise Repository for Oracle Fusion Applications (http://fusionappsoer.oracle.com). Use product-specific layers appropriately as documented.
However, all applications have the following customization layers:

- **Site layer**: Customizations made in the site layer affect all users.
- **User layer**: The user layer is where all personalizations are made. Users do not have to explicitly select this layer. It is automatically selected when users personalize the application.

**Note**

If you are not given the option to choose a layer before you customize, then by default your customizations are made to the site layer.

**Layer Hierarchy**

These 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. If customizations are done to the same object, but in different layers, at run time, the tip layer customizations take precedence. For example, if you customize in the site layer using Page Composer and hide a region, and a user personalizes the same page to have the region displayed, then the user layer will take affect for that user at run time.

**Where Customizations and Layer Information Are Stored**

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

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

**Customization Layers: Examples**

The following scenarios illustrate how customization layers work so that the correct customizations or personalizations are available at run time to the 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 of a specific job role, for example, a sales representative.

**Customization**

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

Now, whenever someone signs in to the application and requests an artifact, the customization engine in MDS checks the repository for XML files that match the requested artifact and the given context, and if there is a match, it layers the instructions on top of the base artifact. In this example, whenever the Sales home page is requested (the artifact) by someone who is assigned the role of Sales Representative (the context), before the page is rendered, the customization engine in MDS pulls the corresponding XML file from the repository, layers it on top of the standard Sales home page, and removes that pane. Whenever someone who is not a Sales Representative signs in (for example, someone with the role of Sales Manager), the XML file with your changes is not layered on top, and so the Quick Create panel is displayed.

This figure shows how the customization XML file is applied to the base document and is visible only to a sales representative.

**Personalization**

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

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

Selecting Customization Layers to Include: Examples

When you use the dialog box to select which customization layer to customize, you can also include lower layers, to view customizations from those layers while you customize.

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

You choose to edit the Job Role layer and select Sales Representative as the value for that layer. You also choose to include the Territory layer and select Southwest as the value. The Site layer is automatically included because it applies to everyone.

While you are customizing in Page Composer, you see customizations that apply to sales representatives in the Southwest territory, based on what was defined for each layer and 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 are making apply only to the edit layer, Job Role.

For example, 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.
Because Territory is higher than Site, you see the field displayed while you are customizing in Page Composer. However, you choose to hide the field as part of your customization, in which case, that customization applies to the Job Role layer, for sales representatives.

Users with other job roles in the Southwest territory might still see the field. However, because Job Role is a higher layer than Territory, the field is hidden for all sales representatives in any territory, unless a layer higher than Job Role applies to any of these users and has the field displayed.

**What You Can Customize and Extend and with Which Tool: Explained**

There are many scenarios for which you can customize and extend Oracle Fusion Applications.

The following tables identify for each scenario the artifacts that you can customize or create in Oracle Fusion Applications, and what tool you use:

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

**Note**

- Presented in the following tables are the top customization tasks, not a comprehensive list.
- Application Composer is available only if you want to make changes in Oracle Sales Cloud.
- Application Composer and other Oracle Sales Cloud customizations and extensions are described in Oracle Sales Cloud: Extending Sales.

The following tables list the types of customizations and extensions that business analysts can make. For more information about design time customizations and extensions by developers and administrators, see the Oracle Fusion Applications Extensibility Guide for Developers.

**Note**

Design time customizations and extensions are not available in Oracle Cloud implementations.

**Page Customization**

This table shows some types of customizations you can make to pages and the corresponding tools to use.

**Note**
While you can customize pages in Page Composer and Application Composer, only certain pages are configured to allow it. If the customization that you want to make is not available in Page Composer, then developers can use JDeveloper to make the customization (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>Add fields, buttons, and links to a standard page (Oracle Sales Cloud).</td>
<td>Application 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 properties for UI components on a standard page (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Customize the UI Shell template.</td>
<td>Page Composer</td>
</tr>
<tr>
<td>Customize a text string wherever it appears across all pages.</td>
<td>Customize User Interface Text page</td>
</tr>
<tr>
<td>Customize the look and feel of simplified pages.</td>
<td>Appearance page in the Settings functional area</td>
</tr>
<tr>
<td>Change the announcements on the simplified home page.</td>
<td>Announcements page in the Settings functional area</td>
</tr>
</tbody>
</table>

### Branding Customization

This table shows some types of customizations 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>
</tbody>
</table>
### Object Customization

This table shows some types of customizations 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>Change the logo and application name in simplified UI.</td>
<td>Appearance page in the Settings functional area</td>
</tr>
<tr>
<td>Customize report layouts.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Customize objects (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<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>Create objects (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Add a business object page to the Navigator menu.</td>
<td>Setup and Maintenance work area</td>
</tr>
<tr>
<td>Add custom object work area pages to the Navigator menu (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Add validation to an object (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Customize saved searches for a custom object (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Customize object workflows for an object (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Create object workflows for an object (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
</tbody>
</table>

### Business Process Customization

This table shows some types of customizations you can make to business processes and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Customization</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a BPMN approval process from Application Composer (Oracle Sales Cloud).</td>
<td>Application Composer and Business Process Composer</td>
</tr>
<tr>
<td>Customize BPM projects created from Application Composer (Oracle Sales Cloud).</td>
<td>Business Process Composer</td>
</tr>
</tbody>
</table>
Security Customization

This table shows some types of security customizations 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 data security to a custom object.</td>
<td>Setup and Maintenance work area</td>
</tr>
<tr>
<td>Grant access to custom objects (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Enable elevated privileges customization.</td>
<td>Application Composer</td>
</tr>
</tbody>
</table>

Business Intelligence Customization

This table shows some types of customizations you can make to business intelligence analytics and reports, and the corresponding tools to use.

<table>
<thead>
<tr>
<th>Customization</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create report layout.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Customize report layouts.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Customize style templates.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Create a report.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Translate a report.</td>
<td>Oracle BI Publisher</td>
</tr>
<tr>
<td>Create a report subject area (Oracle Sales Cloud).</td>
<td>Application Composer</td>
</tr>
<tr>
<td>Customize analytics.</td>
<td>Reports and Analytics pane</td>
</tr>
</tbody>
</table>

Help Customization

This table shows some types of customizations 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>
</tbody>
</table>
Understanding the Customization Development Life Cycle

Customization Life Cycle: Explained

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

Business analysts using Page Composer and Application Composer can make application customizations in a sandbox. Sandboxes store the customizations in Extensible Markup Language (XML) files in a separate Oracle Metadata Services (MDS) repository that is available only when you work in that particular sandbox. The changes can be done in a test-only sandbox (that is, the code in the sandbox is for testing only, and is never deployed), or they can be done in a sandbox that is then published to the 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.

Project managers can monitor, import, and export customizations. The entire environment with all customizations can then be tested, as shown in the figure below.
Tip
Depending upon the need, developers may allow users having access to the Oracle Fusion Functional Setup Manager, to configure the customizations and extensions made to Oracle Fusion Applications.

Run Time Customization Workflow: Explained

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

Note
There are restrictions when more than one user works in a sandbox.

You can also use a sandbox when you define 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, the sandbox can be reviewed and approved by others, and then published to the full test environment where your customizations become part of that repository.

Note
A flexfield sandbox is for testing only and cannot be published. Instead, you deploy a flexfield to the full test environment using the flexfield UI. To test a flexfield configuration before deploying it to the full test environment, deploy it to a flexfield sandbox. The changes that you deploy to a sandbox are isolated from the full test environment and can be seen only by those who make the flexfield sandbox active in their session. After you are satisfied with the changes in the sandbox, you can deploy the changes to the full test environment.

When you publish a sandbox, the published customizations are labeled. Labeling can act as a save point, meaning that if a future customization causes issues, you can use the Manage Customizations dialog box to remove all customizations done after that point by promoting the last known good label back to the tip.

You can also use the Manage Customizations dialog box to view others' customization metadata files, and to download those files to manually move them to another environment or to diagnose any issues. You can also upload others' customization metadata files to your environment.

Note
The navigator menu and report customizations do not use an Oracle Metadata Service (MDS) repository.

This figure illustrates the use of sandboxes when customizing pages, objects, and security using Page Composer and Application Composer and when configuring flexfields.
Viewing and Diagnosing Run Time Customizations: Points to Consider

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

If you are unable to display the page that contains the customizations, choose Manage Customizations from the Administration menu, and then, in the Manage Customizations dialog box, type the page, page fragment, or task flow in the Search text field, and click the Search icon.

Note
Before you begin viewing customizations, ensure that you have administrative privileges to access the Manage Customizations dialog box.

Customizations for a user are visible under the Current Context column on the Manage Customizations dialog box.

Tip
After you are in the Manage Customizations dialog box, you can change the page, page fragment, or task flow for which you are viewing customizations using the Search field.
Developers too may be assigned to specific roles and can view only those customizations that are permitted for that role. However, administrators can view all the 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 might need to view a personalization that was made by another end user. For example, a user might have made an error while personalizing a page and that page is no longer displayed for the user. Because the user cannot access the page, the user cannot 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.

**Logging Page-Level Customizations**

You can use logging that is applied to a page or you can use the Manage Customizations dialog box to diagnose customization issues to determine whether customizations have been applied to a page.

If you suspect that a problem might have been caused by a customization on a page, such as a user interface component disappearing from a page, you can export the page’s customizations and examine the document file.

To turn on run time logging for customizations that are applied to a page, set the log level for the `oracle.mds.custmerge` module to `FINEST`. You can set the application’s log level by choosing Troubleshooting from the Help menu. You might need to ask your administrator to give you privilege to set the log level.

If you have administration privileges, you can also use Fusion Applications Control to set the log level.

**Managing Customizations: Points to Consider**

The Manage Customizations dialog box displays the customizations of the task flows in a page. You can access the Manage Customizations dialog box from Page Composer and from the Administration menu in the global area of Oracle Fusion Applications. You can use this dialog box to perform the following activities:

- Backup and roll back customizations
- Delete page customizations

**Backing Up and Rolling Back Customizations**

Metadata labels are created when you save customizations made in Page Composer. Labels identify the state of the objects in an MDS repository at a given point in time and can serve as save points to which you can roll back your customizations, if the customizations create problems. Therefore, before you make customizations, create a backup of a known good state by creating a label in Page Composer.

For a specific page, to revert to a label that you backed up, you roll back the customization to that label by making that label become the latest version. This
action is often referred to as promoting the label to the tip. You can perform this task in the Promote Documents dialog box accessible from the Manage Customizations dialog box.

---

**Note**

When you use the Manage Customizations dialog, you are rolling back only the customizations for the page and its `pageDef` file. You are not rolling back the other customizations made at the label's save point.

---

### Deleting Customizations

You can use the Manage Customizations dialog box to delete customizations. In the page that contains the customizations, select the page fragment or task flow and choose **Manage Customizations** from the **Administration** menu. In the **Name** list, select the correct layer, find the page, task flow, or fragment that contains the customizations, and click **Delete** for the customization document that you want to delete.

---

**Advanced Customization Life Cycle Tasks: Highlights**

Move customizations from one environment to another by exporting, downloading, and uploading customizations. You can preserve customizations by backing them up or delete customizations if you do not need them. Also, configure log settings and examine log files to diagnose problems associated with the movement of customizations.

**Customization Tasks**

- The logging functionality for customization set migration is different from the standard logging functionality for Oracle Fusion Applications. 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 Troubleshooting Guide.

  See: Adjusting ApplSession Log Levels for Troubleshooting

- Customizations that are created in a sandbox are automatically labeled when the sandbox is published. You need to identify such labels to be able to promote them to the tip. For more information, refer to the Oracle Fusion Applications Administrator's Guide.

  See: Managing Oracle Fusion Applications-Specific Labels in the Oracle Metadata Repository

- For non-Cloud implementations, customization metadata is created from JDeveloper using the Oracle Fusion Applications Administrator Customization role and then packaged and deployed to the source Oracle Fusion Applications environment. For more information, refer to the Oracle Fusion Applications Extensibility Guide for Developers.

  See: Design Time Customization Workflow
Customizing Navigation

Menu Customization: Explained

You use the Manage Menu Customizations task to customize the navigator and home page menus. This task is available from the Setup and Maintenance work area, which is accessible from the Administration menu in the Oracle Fusion Applications global area. Select either Customize - Navigator or Customize - Homepage to proceed with the customization activity.

Note
To perform menu customization at run time, it is important that you have the required privileges.

You customize the menus at the site level and your changes affect all users (or all users of a tenant if in a multi-tenant environment).

Tip
If you are making minor changes, such as adding or editing one or two nodes, then you can hide the changes until you have completed your customizations. However, if you are making more than minor changes, such as rearranging several nodes, you might want to instead create a sandbox before customizing menus.

Navigator Menu Configuration

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

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

The Manage Menu Customizations task displays the menu groups as expandable nodes, with which you can traverse the menu hierarchy.

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

The following table lists the Navigator menu customization tasks that you can perform at run time as well as the tasks that you cannot perform.
<table>
<thead>
<tr>
<th>Permitted Tasks</th>
<th>Restricted Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add and delete custom groups.</td>
<td>You cannot add menu items (links) as top-level nodes. You can add nodes to only</td>
</tr>
<tr>
<td>Edit any group.</td>
<td>the groups in the top level and subgroups.</td>
</tr>
<tr>
<td>Add and delete custom items.</td>
<td>You cannot delete nodes that are delivered with the product. Instead, you can</td>
</tr>
<tr>
<td>Edit any item.</td>
<td>hide them.</td>
</tr>
<tr>
<td>Specify navigation for an item:</td>
<td>You cannot move nodes. Instead, you must duplicate the node and hide the original</td>
</tr>
<tr>
<td>Specify navigation to a UI Shell page in an Oracle Fusion application.</td>
<td>node.</td>
</tr>
<tr>
<td>Specify navigation to an external web page.</td>
<td></td>
</tr>
<tr>
<td>Hide or show groups and items.</td>
<td></td>
</tr>
</tbody>
</table>

**Home Page Menu Configuration**

The home page menu is the set of tabs that are displayed in the Oracle Fusion Applications global area. The home page menu displays tabs for all the items in the menu for which the end user has access privileges. You can add tabs to the home page menu, as well as hide and show them.

The following table lists the Home page menu customization tasks that you can perform at run time as well as the tasks that you cannot perform.

<table>
<thead>
<tr>
<th>Permitted Tasks</th>
<th>Restricted Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add and delete custom items.</td>
<td>You cannot add menu items (links) as sub-nodes. All nodes are top-level nodes.</td>
</tr>
<tr>
<td>Edit any item.</td>
<td>You cannot delete nodes that are delivered with the product. Instead, you can</td>
</tr>
<tr>
<td>Specify navigation to a UI Shell page in an Oracle Fusion application.</td>
<td>hide them.</td>
</tr>
<tr>
<td>Hide or show items.</td>
<td>You cannot move nodes. Instead, you must duplicate the node and hide the original</td>
</tr>
</tbody>
</table>

**Adding Navigator Menu Group: Points to Consider**

You arrange the navigator menu by building a hierarchy of nested groups.

Use the View menu to expand or collapse a group of nodes. You can also right-click a node and access similar actions to facilitate tree navigation.

**Adding Groups**

To add a group, you can insert a group above or below a peer group or insert a child group. You edit a group by defining a label and specifying whether the
group should be rendered. You typically hide the group until all changes have been completed.

**Adding Menu Items: Points to Consider**

The home page menu items are URL links to home pages in Oracle Fusion applications. The Navigator menu items can either be links to UI Shell pages or links to external applications and web sites.

In the menu hierarchy, the home page menu items are always top-level items. Whereas, you can add Navigator menu items to top-level groups and to their subgroups but you cannot add navigator menu items as top-level nodes.

**Adding a Home Page Menu Item**

To add a home page menu item, navigate to the place where you want the item to appear and insert it above or below the existing item. You can also duplicate an existing menu item and position it at the required location. You must provide a label for the menu and link the menu item to a UI Shell page.

**Adding a Navigator Menu Item**

To add a Navigator menu item, you navigate to the item’s group and insert the item above or below another item.

You can also duplicate an existing item. You must provide a label for the menu and either link the menu item to a UI Shell page or link it to an external web site or application.

You can link a Navigator menu item to the following:

- A UI Shell page in an Oracle Fusion application.
- A dynamic URL of a page outside of Oracle Fusion Applications where the host, port, or context root might change.
- A Static URL of a page outside of Oracle Fusion Applications where the host, port, or context root does not change.

**Linking to a UI Shell Page**

If the new item points to a UI Shell page in an application, then you must provide the name of the web application and the view ID of the target page. The web application name and view ID can be obtained from an existing menu item that links to the same UI Shell page.

In a non-Cloud implementation, you also can obtain the web application name from the context root for the application, and you can obtain the view ID from the id attribute for the page's `<view>` tag in the product's public_html/WEB-INF/adfc-config.xml file.

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

If there is another menu item that points to the same page, then you can get the secured resource name and application stripe from that item. In a non-Cloud implementation, you also can obtain the application stripe from the $policies.store.applicationid$ parameter in the application's weblogic-application.xml file. Examples of application stripes are crm, fscm, and hcm.

For non-Cloud applications, you can determine the secured resource name by obtaining the name of the web page's page definition file. By default, the page definition files are located in the view.PageDef package in the Application Sources directory of the view project. If the corresponding JavaServer Faces (JSF) page is saved to a directory other than the default (public_html), or to a subdirectory of the default, then the page definition will also be saved to a package of the same name. An example of a secured resource name is oracle.apps.view.pageDefs.CaseList_Form_Attach_UIShellPagePageDef.

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

**Linking to the Dynamic URL of an External Web Site or Application**

Linking a menu item to a dynamic URL is beneficial in cases where the host, port, or context root to which you point frequently changes. Instead of updating the link to each application, you can update the details of the web application in the topology registration, and that change affects all menu items that contain dynamic links pointing to that web application. For example, you would need a dynamic URL to link to a test version of an application and you will need to change the host and port when you move the application from a test environment to a production environment.

To link to a page outside of Oracle Fusion Applications where the host, port, or context root might change, you must first register the web application in the topology using the Register Enterprise Applications task.

While creating a new menu item on the Create Item Node dialog box, select the Dynamic URL option and provide the details of the web application as per the following example.

When the complete URL to be linked is: http://example:9011/myApp/faces/Page1,

- The name of the web application added to topology would be: myApp (the value that would eventually appear in the Web Application list) and the protocol host, port, and context root values of the URL would be: http://example:9011/myApp
• The value to be provided in the Destination for Web Application field would be: /faces/Page1

Once the menu item is linked to the dynamic URL, the target page appears in a new browser window or tab when you click the menu item.

**Linking to a Static URL of an External Web Site or Application**

This option is used when you link a menu item to a page outside Oracle Fusion Applications where the host, port, or context root remains constant. For example, you can use a static URL to link to http://www.oracle.com.

**Hiding or Displaying Menu Nodes: Points to Consider**

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

**Working with Nodes**

The Manage Menu Customizations page shows all nodes. The Rendered check box is selected by default for all nodes that are added and are visible.

To hide a node, clear the Rendered check box. You can edit the node anytime to either display or hide it.

If you want a menu group or a menu item to appear only if certain conditions are met, you can use an expression language (EL) command to make the node to appear. For example, `#{securityContext.userInRole['ADMIN']}.

A node that appears in italics either contains an EL command or the Rendered check box beside it was cleared, and therefore is hidden from end users.

**Tip**

For major changes that need to be tested and approved, you might want to use the sandbox manager instead of hiding and showing nodes.

**Design Time Menu Customizations: Highlights**

The menu customization feature provides several options to add, modify, and organize the Navigator and home page menus during design time. You must have developer rights to perform these customizations.

**Note**

Design time menu customizations are not applicable to Oracle Cloud implementations.

An overview of customizing the Navigator menu and home page is provided in the Oracle Fusion Applications Extensibility Guide.
Customizations

- Use Oracle JDeveloper to customize the Navigator and home page menus at design time.
  See: Customizing Menus

- Define translations for your customizations in the locales you support.
  See: Translating Menu Customizations

- Customize the page template to display the Navigator menu groups as separate menus, each of them displaying their list of menu items. Refer to the Oracle Fusion Applications Developer's Guide.
  See: Rendering the Navigator Menu as Dropdown Buttons

Troubleshooting Navigator Menu: Highlights

If the Navigator menu does not display customizations as expected, use the following troubleshooting tips to verify the changes.

Issues and Resolutions

- If an expected menu item does not appear in the Navigator menu, verify whether the menu item has been hidden from view.

- If a custom menu item was added and the browser does not display the page indicated by the URL, open the Manage Menu Customizations task and verify whether the web application name is the same as the context root for the application, and that the view ID is the id attribute for the page's <view> tag in the product's public_html/WEB-INF/adfc-config.xml file. The URL should not contain the .JSPX suffix.

- If you see a "webApp value not define" error message when you choose an item in the Navigator menu, verify whether the application is in the topology tables. Refer to the Oracle Fusion Applications Administrator's Guide.
  See: Viewing the Routing Topology of an Oracle Fusion Applications Instance, Product Family, or Product

Customizing Help

Help File Customization: Overview

If you have the appropriate job roles, then you can customize the help files in the help site. Use the Manage Custom Help page to maintain both predefined and custom help files. You can create, duplicate, edit, and delete custom files, or set their status to Active or Inactive. For predefined files, you can only duplicate them or set their status.
For each help file, predefined or custom, use help locations to determine where the help file appears in the application and in the help site. You have various options in how you add custom help, for example by uploading a file or specifying a URL. You can upload files of any type, and your URL can also point to files of any type.

**Note**
To 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 and you must select the following objects to export: Help Configuration and Help Topic.

**Customizing Help in Help Windows**
Many help files can be accessed from help windows in the application. If you want to customize help in the context of a help window, for example create a custom help file and add a link to it from a specific help window, then start by opening that help window. When you click the Manage Custom Help link, you go to the Manage Custom Help page, and the help location fields are automatically populated with values that correspond to the help window. This way you can easily select existing files to add to the same help location, and when you create a new file, the same help location appears by default.

**Restriction**
Aside from links to help files, some help windows also display informational text. You can’t use the Manage Custom Help page to edit this text. Your technical administrators can do so using developer tools (not available in Oracle Cloud implementations).

**Customizing Help by Help Location**
Open the Manage Custom Help page directly from the home page of Oracle Fusion Applications Help or from search result pages.

**Tip**
When you search in the Manage Custom Help page, make sure that the Custom Help Only check box is not selected if you are looking for predefined help.

**Editing Specific Files**
To edit a specific file, you can either find it in the Manage Custom Help page, or open the file itself and click the Edit link.

**Customizing Glossary Terms**
You can find glossary terms in the Manage Custom Help page, or go to Navigator - Glossary in Oracle Fusion Applications Help to open the Glossary tab, search for the term, and click Edit.

**Adding UPK to the Settings and Actions Menu**
If your enterprise has purchased Oracle User Productivity Kit (UPK) content, then your administrator can also add a UPK item to the Settings and Actions
menu in the global area of Oracle Fusion Applications. When users select this menu item, they access UPK content specific to the page that they are on.

**Help Types: Explained**

Oracle Fusion Applications Help contains various types of help content, including demos, examples, FAQs, glossary terms, help topics, and PDF guides. A business process or product can be supported by some or all of these help types.

**Demo**

Demos are Oracle User Productivity Kit (UPK) topics that visually demonstrate how to use the application to complete a short task or portion of a task. Demos can also provide an introduction to complex dashboards and work areas.

**Example**

Examples provide real use cases of features to illustrate how and when to use the feature, or scenarios to illustrate abstract concepts. Worked examples show exactly what you need to do to achieve a specific result, emphasizing decisions that you make and values that you enter.

**FAQ**

FAQs, or frequently asked questions, provide brief answers to questions that you might have regarding a task or page. For example, they can briefly explain what a term means, why something happened, how you can perform an action, or what happens if you perform the action.

**Glossary**

Glossary terms provide definitions for words or phrases used in help. You can search or browse glossary terms in the Glossary tab of Oracle Fusion Applications Help. Where the links are available, you can also see the definition when you hover over the term in help content for other help types.

**Help Topic**

Help topics explain key concepts, illustrate how application components work together, or assist in decision-making by explaining points to consider or the options you have. Help topics can also provide reference, overview, and other information.

**PDF Guide**

PDF guides in Oracle Fusion Applications Help provide, in a book format, information usually not found in other help types. There are other guides that
present a collection of help content from the other help types, except demos, in an organized and logical format. These guides, for example, address specific business processes and setup offerings. You can find these guides by going to Navigator - Documentation Library in Oracle Fusion Applications Help.

Help Locations: Explained

Help locations determine where users can find help files, custom or not, from either the application or the help site.

Help locations include:

- Page or section values
- Help hierarchies
- Primary locations

Page or Section Values

The value in the Page or Section field on the help customization pages represents where users can click a help icon to open a help window that contains a link to the help file. In most cases, this value represents a page or region header in the application. Help windows are also available on specific tabs or windows, and in the Setup and Maintenance work area for specific task lists or tasks. You can associate a help file with multiple page or section values, or with none at all.

The page or section value reflects the logical navigation to the help window. For example, Edit Opportunity page, Revenue tab, Recommendations window does not mean that the help file is available in three different places. The help icon is in the Recommendations window, which is accessed from the Revenue tab on the Edit Opportunity page.

If the value suggests multiple locations, for example Create and Edit Opportunity pages, then the help file is available from the page header of both the Create Opportunity and Edit Opportunity pages. If the page or section value is, for example, a dashboard region that appears in multiple dashboards, then the value does not specify the page name but just the region. The help file is available from that region in multiple dashboards.

Help Hierarchies

Help files are associated with help hierarchies, which are used to categorize help files and aid users in finding help. Each help file can have multiple hierarchies, with at least one of type Business Processes. The business process hierarchy is based on the Business Process Management model. Every page or section value is predefined with a specific business process hierarchy. If you select a page or section without entering a business process hierarchy, the predefined hierarchy appears by default.

The Common Tasks navigator is based on the Welcome hierarchy type. The level 1 nodes represent categories of functional areas common to all users.
The Business Processes navigator in the help site is based on the business process hierarchy. For example, if you assign two business process hierarchies to a help file, users can find the file in both locations in the navigator. When the user clicks More Help from a help window, all help files assigned to the same business process hierarchy as the page or section value are returned as search results.

Similarly, the Products navigator is based on the Product hierarchy type, in which level 1 is the product family, level 2 is the product, and level 3 is the business activity owned by that product.

The Functional Setup navigator is based on the Functional Setup hierarchy type. The level 1 nodes for this hierarchy are:

- Functional Setup Manager, which includes help about using the Setup and Maintenance work area.
- Offerings, which contains level 2 nodes for each setup offering, and lower levels for the main task lists in the offerings. Help for the task lists and tasks are included.

**Primary Locations**

The primary location of a help file designates the hierarchy that is displayed for the help file in search results and within the help content as breadcrumbs. You cannot change the primary location of a help file that came with your help installation. Primary locations of predefined help are based on the business process hierarchy, while custom help files can have primary locations based on hierarchies of any type.

**Editing Predefined Help and Glossary Terms: Points to Consider**

When you open any predefined help file, including glossary terms, that came with Oracle Fusion Applications Help, you can see an edit option if you have roles allowing edit access. When you edit predefined help, keep in mind:

- What happens to the original help file
- Where predefined help appears
- Considerations specific to glossary terms

**What Happens to the Original Files**

When you edit predefined help, you are actually creating a new custom help file based on the original file, with the same help locations. The customized version replaces the original, which becomes inactive and hidden from users. You can display both versions by reactivating the original in the Manage Custom Help page.

**Note**

In the Search Results: Existing Help region on the Manage Custom Help page, there is no option to edit predefined help. You can duplicate a predefined help file, edit the copy, and optionally inactivate the original.
Where Predefined Help Appears

All predefined help comes with preassigned help locations, including at least one based on the hierarchy of type Business Processes. Many also have predefined page or section values that indicate where the help can be accessed from help windows in the application.

To change where predefined help appears, either in the help site navigators or in the application, create a duplicate in the Manage Custom Help page. Change or add help locations to your custom copy, and inactivate the original.

Even though glossary terms do not appear in the help site navigators, you still need to enter at least one help location to categorize the glossary term.

Considerations Specific to Glossary Terms

When you edit a predefined glossary term, the original term becomes inactive. Existing links to the glossary term, from other predefined and custom help files, will automatically point to your custom version. If you later inactivate the custom glossary term, make sure to activate the original term so that the links still 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 are working on custom help created by duplicating a predefined help file, then you may see existing links from the original file in the Help Content section. 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 source type. In other words, you must type the help content directly in the application or use an HTML file that you upload to help. 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.

Glossary Terms

Customizing and Extending Oracle Sales Cloud 3-27
• You can delete this code to remove all related help, or delete title IDs to remove individual links (for example, CREATE_AUTOMATIC_POSTING_CRITERIA_S_0000).

• To replace existing links or add new links, you need to retain the code syntax and enter desired title IDs. To find title IDs, search for the help files on the Manage Custom Help page. Title IDs are displayed in the search results, but the Title ID column is hidden by default.

**Standard Hypertext Links**

You can create standard hypertext links to any file or Web site as long as you ensure the stability and validity of the links, including links to other help files, custom or not. These links can appear anywhere in the body of your help file as long as they come before any related help links.

In the Help Content section, highlight the text that you want to use as link text and click the Add Link icon button.

For links to other help files, open the file to which you want to link, and click the E-Mail link. Use the URL in the autogenerated e-mail text as the link to the file.

**Note**

Use the full URL, for example http://www.oracle.com, when creating links.

**Links to Documentation Library Content**

The syntax for links to HTML files in documentation libraries is:

```
<span class="HP_topic-link_bridgeDocument-linkToSTDoc_">?ofa linkToSTDoc(WCSUG4636) ?\span class="HP_topic-linktext_">Understanding Tags</span><?ofa endLink /></span>
```

WCSUG4636 is the anchor ID and Understanding Tags is the link text. You can:

• Modify the link by replacing the existing anchor ID or 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.

**Important**

To ensure that you are linking to a supported documentation library, enter anchor IDs only from documentation libraries that are linked from predefined help topics.

**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.

`0feGlossaryTerm("accounting period", ACCOUNTING_PERIOD_00001)`
In this example, accounting period is the link text, or glossary term, and ACCOUNTING_PERIOD_001 is the identifier, or title ID.

- To remove the link but retain the text, delete all the code except the term itself.
- To add glossary term links, you must follow the link syntax and use the correct title ID for the glossary term. You can find title IDs in the search results of the Manage Custom Help page.

Note
If your custom help has glossary terms and the source type is Desktop File, then make sure before uploading that the quotes around the glossary term 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 Oracle Fusion Applications Help. This guide is currently not available from any help window in the application.

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 a chapter and remove a section in that chapter, to hide content about a particular subject</td>
</tr>
<tr>
<td>Which help window should the customized guide appear in?</td>
<td>The help window for the entire Welcome dashboard of Oracle Fusion Applications</td>
</tr>
<tr>
<td>Which help navigators should the customized guide appear in, and on which node?</td>
<td>Same as the original guide, plus the path 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>
</tbody>
</table>

Edit a copy of the original PDF guide, and use the Manage Custom Help page to replace the original PDF guide with your new file.

Copying and Editing the PDF Guide

1. Open the original PDF guide from the help site and save a copy to your desktop. Leave open the help file for the guide.

2. Using a PDF editor application, change the title of the chapter wherever the chapter title 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. Add a help location with the Business Processes hierarchy type and select **Information Technology Management** as the level 1 node, **Manage Enterprise Application Software** as the level 2 node, and **Use Applications** as the level 3 node.

7. Select **Welcome page** in the **Page or Section** column.

8. Click **Save and Close**. The help file for the original PDF guide is automatically set to inactive.

### Adding Custom UPK Content to Help: Worked Example

This example demonstrates how to add custom Oracle User Productivity Kit (UPK) topics as demo help files. These help files function like any predefined help file for demos. You can search and include these files in help windows and navigators as you would other help.

In this scenario, you are adding one demo about activity streams, to appear in a help window on the Welcome dashboard.

**Note**

Your demo must be made with UPK 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 UPK content do you want to add to help?</td>
<td>From a UPK module containing five topics, add one as a custom demo on the help site</td>
</tr>
</tbody>
</table>
| Which help navigators should the demo appear in, and on which node? | Because the demo is about activity streams:  
Search by Common Tasks navigator, under the Collaboration node  
Search by Business Process navigator, under Information Technology ManagementManage Collaboration Manage Collaborative Communities |
| Which help window should the demo appear in? | On the Welcome dashboard of Oracle Fusion Applications, in the help window in the Activity Stream region |
| Do you want to limit access to the help file for the demo? | No |
Generate a report of UPK document IDs, which you will use when creating custom help, to identify the UPK topic that you want to add. Publish the UPK module as a player package, then create custom help for the UPK topic that you want to use as a help demo.

**Generating a UPK Document ID Report**

1. In the UPK 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 column. Close the Column Chooser dialog box.
4. From the File menu, select to print, and save the output as a Microsoft Excel file to your desktop.

**Creating the Player Package**

1. From the UPK Developer, make sure that the topic that you want to add as a demo 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, to ensure that the custom help file includes a text-only version of the UPK topic.
5. Finish the publishing process, after optionally setting other options.
6. Navigate to the location 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 UPK content.

**Creating Custom Help for Demos**

1. Open the help window in the Activity Stream region on the Welcome dashboard of Oracle Fusion Applications, and click **Manage Custom Help**.
2. On the Manage Custom Help page, the page or section and hierarchy values are populated with the values for the Activity Stream region.
3. Click **Create**.
4. On the Create Help page, complete the fields in the General Information section, as shown in this table. Use the default values except where indicated.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you want the help file to appear in the New and Updated pane?</td>
<td>Yes</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Title</td>
<td>The name of the UPK topic.</td>
</tr>
<tr>
<td>Source Type</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/UPKcontent/PlayerPackage</td>
</tr>
<tr>
<td>Document ID</td>
<td>The document ID of the UPK topic to add to the help window in the Activity Stream region. You can copy and paste this ID from the Microsoft Excel file that you generated earlier.</td>
</tr>
<tr>
<td>Help Type</td>
<td>Demo</td>
</tr>
<tr>
<td>Help Security Group</td>
<td>Unsecured</td>
</tr>
<tr>
<td>Keywords</td>
<td>Terms relevant to the demo.</td>
</tr>
<tr>
<td>Description</td>
<td>Summary of the demo.</td>
</tr>
<tr>
<td>Include in New and Updated pane</td>
<td>Selected</td>
</tr>
</tbody>
</table>

The Help Location section contains values for the help window in the Activity Stream region. This help file will also appear in the Search by Business Process navigator under this predefined hierarchy.

5. Click **Save and Close**.

6. On the Manage Custom Help page, open the help locations for the help file that you just created.

7. Add a help location with the Welcome hierarchy type and select **Collaboration Features** as the level 1 node.

8. Click **Save and Close**.

**Customizing Embedded Help: Highlights**

You can customize help that is embedded in the application, for example hints, for all users of Oracle Fusion Applications. There are different types of embedded help.
Creating, Editing, or Deleting 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. Usually, the value resolves to a key in a resource bundle.

- Edit, create, or delete other types of embedded help. Refer to the Customizing or Adding Static Instructions, In-Field Notes, and Terminology Definitions section.

See: Oracle Fusion Applications Extensibility Guide for Developers

FAQs for Help File Customization

How can I add a Youtube video to custom help?

On the Create Help page, select URL as the Source Type. Find the video in Youtube and click the Share button then click the Embed button. Copy the URL within the embed code, for example http://www.youtube.com/embed/<unique ID>, and paste it in the URL field of the Create Help page.

How can I restrict help content to specific user roles?

When you create or edit help, select a help security group that represents the set of roles that you want to have access to the help. If you do not see the Security Group field, then your administrator has not selected the Custom Help Security feature choice. The Unsecured group has no associated roles, so anyone can view the help. The predefined Secured group includes all internal employees and contingent workers, unless this group has been edited. You can create security groups and associate roles using the Manage Help Security Groups page, which you can access by starting in the Setup and Maintenance Overview page and searching for the Manage Help Security Groups task. Your new security groups are immediately available for use to secure new or edited help files.

Why can't I select and add help to a location?

You must specify a page or section to add the existing help to. To ensure that help is added to the correct help window, go to the page or section in the application, click the Help icon, and click the Manage Custom Help link in the help window. Alternatively, in the Manage Custom Help page, search for at least a page or section and a level 1 value for the Business Processes hierarchy type before selecting the Select and Add option.

You cannot select and add help to a particular hierarchy, on the Manage Custom Help page, without a page or section. To add just a hierarchy, search for the help file, add a new help location, and specify only the hierarchy information.

What happens to custom help when a help patch is applied?

Oracle Fusion Applications Help patches update all help files, both active and inactive, except custom help. Custom help files are not affected by patches.
Consider reviewing inactive files to see if you want to activate the updated version, or to make similar edits to the custom versions of those files, if any.
Application Composer: Using Sandboxes

Using Sandboxes: Overview

Read this chapter to learn about proper and recommended sandbox usage when customizing and extending your Oracle Sales Cloud applications. When doing customization work within Oracle Fusion, you must always use a sandbox as a safety precaution, so that you can fully test your changes before rolling them out to your end users.

In this chapter, you will learn about:

- 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 manage and publish sandboxes

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.

Using Sandboxes: Points to Consider

In the customization run time workflow, you use sandboxes to isolate the changes from the mainline code for testing and validating. After you are satisfied with the changes, you can publish the changes back to the mainline code.

You can 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 code. The following figure illustrates the two types of sandboxes and their relationship to the mainline code.

![Diagram of sandbox relationships](image.png)

**Working with a Single Sandbox**

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

Conflicts may also arise when users are editing a shared artifact, such as when a user performs an operation that adds or edits a translatable string. For example, a user edits a field's display label or help text, or a validation rule's error message, while another user performs an operation around the same time that similarly affects translatable strings. Another example of a shared artifact conflict is when two or more users are working in navigator menus that are shared across applications. Whenever there is a customization conflict with another user, 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 should not update object1. Remember that if both modifications involve
changes to translatable strings, then saving changes to separate objects around the same time may still cause a conflict in the resource bundle that stores the translatable strings.

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

Working with Multiple Sandboxes

Multiple sandboxes are used when customizations are stored in testing as well as production sandboxes.

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

Tip

If you encounter a message showing a conflict on oracle/apps/menu/fnd/applcore/dataSecurity/dataSecurityService/mds/DSMO.xml when you publish your sandbox, it indicates that the security changes that you made in your sandbox conflict with other security changes in the mainline code. 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:

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

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

• If an artifact is updated in both the mainline code and in one sandbox or in two different sandboxes, when the sandbox is published, such conflicts are detected and an error is displayed. 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 to avoid any inconsistencies between the run time caches and the ADF Business Components definitions.
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 CloudApplications 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 out-of-the-box functionality to help isolate and control customization efforts without impacting other users’ configuration environments, and/or the production environment. You can read all about sandboxes in the Oracle Public Cloud Services Extensibility Guide for Business Analysts. Or, review this document to learn how to customize Oracle Sales CloudApplications, 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 back to the mainline code. When making changes in a sandbox, your customizations will not be available to any other sandbox or to application in the mainline code until you have published your sandbox. When publishing a sandbox, the included application customizations overwrite the mainline code application’s existing configuration.

Within this section, 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 applications by creating completely new business objects and artifacts.

- **Page Composer**: Customize pages.

For more information on customizing pages using Page Composer, see “Editing a Page in Page Composer” in Oracle Fusion Applications.
Customization Impact Areas

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.

Customization Types

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 here should 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.

Sandbox Manager

You can maintain sandboxes using the Sandbox Manager:

- Create a sandbox
- Activate a sandbox
- Delete a sandbox
- Publish a sandbox
- View available or published sandboxes

Managing Customizations Using Sandboxes: Explained

Different types of customizations can be applied to an application such as 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.

Environment

To customize an Oracle Fusion application in run time, you must first create a sandbox and then use Page Composer or Application Composer to make the
customizations. These changes remain within the sandbox and do not affect the mainline code. You can test and validate the changes by publishing the sandbox to a full test environment. After the application has been tested, it can then be moved to the production environment. The customizations created in the sandbox will be migrated to the production environment and will be available to the users of the system.

It is recommended that customizations are never made directly to the mainline code and have to be first done in the sandbox. You make changes to an application at run time in a sandbox so that the changes are isolated from the mainline code. The mainline code is a branch of data that serves as a single source. After the changes in the sandbox are complete and verified and you want to commit them, you can publish the metadata or security-enabled sandbox to the mainline code. If customizations existed in the mainline code before the sandbox was created, you will see the customization information in the sandbox. To view the customizations that were newly added to the mainline code, you will need to exit, publish, or delete the sandbox and create a new sandbox.

Flexfield sandboxes are for testing only and cannot be published. You make flexfield configurations that are stored in a database, and then deploy those configurations to a sandbox to see the resulting deployment artifacts in a sandbox environment. Flexfields are deployed directly to the mainline code using the flexfield UI.

**Tools**

You can use several run time tools to customize the application. The sandbox manager works with Application Composer and Page Composer to customize objects and pages. The Oracle Business Process Composer and Oracle SOA Composer are also run time customization tools, but they do not use the sandbox manager. They have their own mechanisms for handling customization changes.

For on-premise implementations, a metadata sandbox that you create using the sandbox manager is available in JDeveloper when you are creating and deploying customizations intended for a deployed Oracle Fusion Application in Oracle WebLogic Server. The available sandboxes except security sandboxes appear in a selection list in JDeveloper during deployment.

**Note**

For cloud implementations, you can use Application Composer to work with sandboxes. For more information on using sandboxes in Application Composer, refer to the Oracle Sales Extensibility Guide.

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

**Sandboxes: 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.

Sandbox Development Lifecycle Components: Explained

The development lifecycle and planning may look like the following for each customization cycle, usually one week:
This figure shows the development lifecycle and planning for each customization cycle.

Stage

This figure shows the different stages in the sandbox development lifecycle.
Publishing a Sandbox

Publishing a sandbox involves submitting a Service Request to Oracle. This figure shows the flow for publishing a sandbox.

Invalid Sandbox

In the following figure, sandbox A, B, and C are now invalid with the new label, as these sandboxes were derived from the 4/19/2012 mainline code.
Example of a Multi-week Cycle

This example shows a multi-week cycle.

This figure shows an example of a multi-week cycle.

Configuration Cycle

This figure shows the configuration cycle for week 2.
Setting Up Sandboxes: Procedures

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 you examine the changes and are certain about them, you can publish the sandbox, or deploy the flexfield, and the changes are merged into the mainline code. Eventually, the sandbox is archived.

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

To create a new sandbox and set it up, perform the following steps:

1. In the global area of the Oracle Fusion Applications, select Administration - Manage Sandboxes.
2. On the Manage Sandboxes dialog box, create a new sandbox.
3. To make it a security-enabled sandbox, select the Create Data Security Sandbox check box on the Create Sandbox dialog box. If it is not required, skip this step.

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

4. Click Save and Close.

5. On the Manage Sandboxes dialog box, select the newly created sandbox or an existing sandbox and click Set as Active. The sandbox is designated as the active sandbox.

6. Perform the following optional tasks, if required:
   • To export the sandbox to a file so that it can be transported or shared, click the sandbox. On the Sandbox Details dialog box, click Download All.
   • To import a sandbox from a file, click Import, upload the file and click OK.

7. Close the Manage Sandboxes dialog box.

Using the Sandbox Manager: 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
   • Review a list of available or published sandboxes
   • 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.

Tip

To customize data security policies without affecting the mainline code, create a separate data security sandbox by selecting the Create data security sandbox check box when creating a sandbox. When working within Oracle Sales Cloud, however, do not create data security sandboxes.

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-key) 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 code. 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 Code**

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 code. 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.

---

**Important**

Once back in the application in the mainline code, avoid making customizations using Application Composer. To start making customizations again, use the Sandbox Manager to set a new sandbox as active.

---

**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 code. Always publish customizations from the integration sandbox only.

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

**Warning**

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.

**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.

**Multiple Sandbox User Conflicts: Explained**

Customizations are stored as XML files in the Metadata Services (MDS) repository, segregated by sandbox. 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 Exist**

A user might overwrite changes performed by other users using the same sandbox:

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

When more than one user is working in the same sandbox, and the same object within the sandbox is saved at the same time, only one user's customization is saved. The other user's customization (customization B) may not be lost, but will overwrite customization A if the other user again tries to save customization B.

If customizations on the same object are saved at different times, then the last saved customization will overwrite the other user's changes.

**Multiple Integration Sandboxes Exist**

When multiple sandboxes exist for publication, one user might overwrite changes made by another user during sandbox publication into the application in the mainline code. The resulting mainline code configuration is always from the last published sandbox.

**Avoiding Conflicts**

To avoid such conflicts, comply with these guidelines:
• 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 username. Do not share users per administrator 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.

Users should never 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.)

Publishing Sandboxes: Procedures

After you have made the customizations in the sandbox, you need to publish them to make them available in the application. 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 code from another source and you publish your sandbox data, then the mainline code is not overwritten. However, if there are conflicts you are notified, and you need to fix the conflicts before publishing.

To publish a sandbox:

1. In the global area of the Oracle Fusion Applications, select Administration - Manage Sandboxes.
2. On the Mange 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 code.
4. Close the Manage Sandboxes dialog box.

FAQs for Using Sandboxes

When do I publish a sandbox?

You can publish when configurations have been thoroughly tested and are ready to be moved to the application in the mainline code.

You need to test the following configurations, which must be tested 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.

Note

Once you publish an integration sandbox, all private sandboxes are invalid because the label in the mainline code 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 a sandbox.

Note

When you delete a sandbox, you should first confirm that the sandbox is not active. Only non-published sandboxes can be deleted.

Once 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 entirely brand new sandboxes, including a new integration sandbox.

**What's a data security service error?**

When publishing a data security enabled sandbox, data security changes made in the sandbox Metadata Services (MDS) repository must not conflict with data security changes that have been made in the mainline code MDS repository that the sandbox was created from. A conflict causes a data security service error. If you encounter a message showing a conflict when you publish your sandbox, do the following:

1. Exit the sandbox
2. Create a new sandbox, activate it, and resume your work
3. Apply or create your security changes in the new sandbox
4. Delete the original sandbox where you encountered the error.
Application Composer: Using the Application Composer

Using Application Composer: Overview

Read this chapter to learn about the primary task flows that are available in Application Composer.

In this chapter, you will learn about:
- How application composer works, at a summary level
- 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 create, or modify, the user interface pages to display your custom objects and custom fields on various pages
- How to edit saved searches that were created using Application Composer
- How to secure custom objects, both in terms of what users can do on the object's work area, as well as the data that users can see

Other chapters in this guide also describe additional tasks flows that are available in Application Composer, such as creating object workflows and custom subject areas, writing Groovy scripts, and importing and exporting your customizations. Refer to the table of contents for these other chapters.

To navigate to Application Composer to make your application changes, select Application Composer from the Navigator, under the Tools > Customization category. Remember to always select the desired application from the Application choice list first, before making any changes. To test your changes in the actual application, use the Navigator to switch to the desired application to view your changes at run time.

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 desktop 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. Data model changes, such as the creation of custom fields, do require that you reauthenticate before you can see those changes.

**Pattern-Based Application Design**

Application Composer hides the complexity of customization from business analysts 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.

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

From Application Composer’s Overview page, you can make application changes such as:

- Customize objects by adding new fields, or create entirely new objects.
- Create foreign key-based relationships between two objects.
- Customize desktop pages by exposing your newly created fields for an object, or create an entirely new work area for your newly created objects.
Expose object relationships on desktop pages in the form of subtabs or tree nodes.

- Write application logic, such as triggers, validation rules, and workflows, for an object or for use across multiple objects.
- Implement functional and data-level security for custom objects.
- Enable objects for custom reporting.

**Getting Started: Application Composer's Overview Page**

To access Application Composer, log in with the **Customer Relationship Management Administrator** job role. Then, select **Application Composer** under the Tools > Customization category in the Navigator menu to navigate to the main Overview page.

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.

- 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.

Change the selected application in the **Application** choice list at any time to customize another Sales Cloud application.
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, such as Sales or Marketing. This is a critical selection because your customizations reside within that application only. Additionally, this concept of Web application “boundaries” is important because it directly impacts what you can do when making changes using Application Composer. Specifically, some customization tasks that you might want to complete across Web applications are available to you only between certain applications, not all.

This topic introduces you to the concept of Web applications, and also describes the customization tasks that cross Web application boundaries which you can complete only in selected Sales Cloud applications:

- Customer Center
- Marketing
- Sales

Selecting a Web Application

To complete a customization task, such as create a new custom object, you first select an application (such as Sales or Marketing) 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

What is a cross-application customization task? This refers to any customization 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 (Oracle Fusion Sales) that displays marketing campaign records (Oracle Fusion Marketing).

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, provided that the object you are customizing exists in one of three Sales Cloud applications:

- **Customer Center**
- **Marketing**
- **Sales**

**Note**
Crossing Web application boundaries is constrained to only the above three applications. In other words, if you are in Customer Center, Marketing, or Sales, then you can reference all objects (both custom and standard) from across those same three applications. Additionally, you can also reference standard objects (but not custom) from the Common application.

**Object Model Customizations**
When customizing an object's model in Customer Center, Marketing, or Sales, you can reference standard and custom objects from across those same three applications, plus standard objects from the Common application. 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 sales lead records (Marketing).
This lets customers see which sales lead an opportunity is converted from.

- Or, create a dynamic choice list field for the sales account object (Customer Center) which is populated with competitor records (Sales). This lets customers see which competitor they need to compete against for a given customer.

- 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 sales lead object is the target.

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

This lets customers see if more than one sales lead converted to a single opportunity.

- Or, create a relationship where the sales account object is the source and the competitor object is the target.

This enables the creation of the competitor object tree node on the Customer 360 tree. See the "User Interface Customizations" section in this topic.

---

**Tip**

Create a one-to-many relationship instead of a dynamic choice list, if there are usually multiple competitors for a given customer.

---

**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.

---

**Tip**
Object model customizations include the creation of custom objects, as well. When creating a new object, consider whether you eventually plan to share that object between Customer Center, Marketing, and Sales. If so, create that object as a top-level custom object in Customer Center. On the other hand, if you need to share that object with any objects that exist under the Common Web application, then create that object as a top-level custom object in the Common Web application.

**User Interface Customizations**

When customizing an object’s user interface in Customer Center, Marketing, or Sales, you can reference standard and custom objects from across those same three applications, plus standard objects from the Common application. 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’s 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 Sales Leads 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 in Customer Center, Marketing, or Sales, you can reference standard and custom objects from across those same three applications, plus standard objects from the Common application, in your scripts. The types of changes you can make include:

- When writing a script, access data from any standard object in any Sales Cloud Web application, as well as from any custom top-level object created in Customer Center, Marketing, or Sales, using the newView() built-in function. The newView() built-in function is described in "Accessing the View Object for Programmatic Access to Business Objects" in Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support.

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 leads that involve this opportunity. At run time, when a user saves an opportunity record, the trigger populates a custom field on the opportunity with the number of related leads.

- Or, write a script in a "before insert" trigger on the sales account object that retrieves information about the opportunity associated with the organization that the sales account is tied to.
Common Objects: Exceptions
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 (CPUI) objects. These Common objects are available to you when customizing an object in Customer Center, Marketing, or Sales.

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 outside the Common Web application. Your scripts will work only when triggered within the context of the other Web application.

For example, when creating a note, you can write a script to insert details about an opportunity into a custom field on the note. This script will work only if a user creates the note from within Sales. If, however, the user creates the note from within Marketing, then the script will not work.

Customizing Oracle Sales Cloud Applications Using Application Composer: Explained

Application Composer provides a series of task flows which let you customize and extend an Oracle Sales Cloud application according to the needs of your users. For example, you can create fields for an existing standard object, and expose those new fields on the object's work area. Or, create a brand new custom object and related fields, then create a work area where that object and its fields are exposed to users. The task flows available to you are dependent upon the application that you are customizing.

Available Customization Task Flows
Different sets of customization task flows are available to you, depending on the application that you are customizing. See Customizing Oracle Sales Cloud Applications Using Application Composer (Doc ID 1516151.1) on My Oracle Support at https://support.oracle.com. This document provides a list of which task flows are available for use in these Sales Cloud applications:

- Common
  This includes Master Data Management (MDM) and Common Party User Interface (CPUI) objects.
- Customer Center
- Marketing
- Sales
- Oracle Fusion Sales Catalog

You can also refer to the product-specific implementation guides to learn more about how a particular application works with Application Composer.

Defining Objects: Explained

One of the primary customization options available to you when using Application Composer is the ability to extend an Oracle Sales Cloud...
application’s object model. Customize Sales Cloud objects by adding 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).

Review these aspects of the object model extension process in Application Composer before you begin to customize or extend your Sales Cloud application’s object model:

- Browsing the object tree
- Creating a custom object
- Using the Object Overview page
- Editing an object’s attributes
- Viewing child and related objects
- Deleting a custom object

**Application Composer’s Object Tree**

Access Application Composer from a Sales Cloud application at run time by using 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 > Customization 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, such as an object's fields and UI pages where the object is exposed.

![Application Composer](image)

**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.

- **Saved searches**
  - Edit saved searches for an object.

For custom objects, you can also view and edit details for:
• Security

Implement functional and data-level security for an object and its records.

Creating a Custom Object

To create a new custom object, you first select an application on the main Overview page of Application Composer. The new custom object will belong to the application that you select. After you select the application:

1. Select the **Custom Objects** node or link in either the object tree or local area of the main Overview page.

   On the resulting summary table, click the New icon.

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

3. 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**

      Use the **Record Name** field to specify the display label for the object’s RecordName field. The RecordName field stores the "name" of the record. For example, for an opportunity object, this RecordName field stores the opportunity’s name. Accordingly, if you were creating this object as a custom object, then you would set the **Record Name** field to **Opportunity Name**.

      Typically, this field is the object’s primary user-recognizable identifier for the object, and as such, is usually the identifier that run time users drill down on, from the overview page to the detail page.

   d. **Object Name**

      The object name is the internal name for the object.

   e. **Description**

---

**Tip**

To create a custom child object, click the Create Child Object button in the standard or custom objects summary table, or from an object’s Object Overview page.

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. Select a standard or custom object in the object tree.
3. Or, select the Standard Objects or Custom Objects node or link in either the object tree or local area of the main Overview page, choose an object from the resulting summary table, and select the Edit icon.

From the Object Overview page, you can:
- Edit the object’s primary attributes, described in the previous section.
- 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. Select an application on the main Overview page.
2. Select the Standard Objects or Custom Objects node or link in either the object tree or local area of the main Overview page.
3. From the resulting summary table, select an object and then select the Edit icon to navigate to its Object Overview page.
4. On the Object Overview page, click Edit:
   - Change the object’s display label, description, and record name at any time.
• You cannot change the Object Name and API Name after the object has been created.

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

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, then all its children 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 code, optionally enter a note in the description that the object is no longer used.

Object Relationships: Explained

A relationship is a foreign key association between two objects, and indicates a connection between two objects’ data. You can expose this connection on desktop pages through the use of child or related object subtabs or tree nodes. Using the Oracle Fusion CRM 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 or tree node 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. You can 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.

Review these aspects of the relationship creation process in Application Composer before you begin to create relationships between objects:

• Relationship types
• Creating reference relationships
• Adding subtabs or tree nodes
• Many-to-many relationships

Relationship Types

Four types of one-to-many relationships exist:

• Parent child relationship

  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 on the parent object’s Object Overview page. 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.

  You can also view the parent child relationship from the child object’s Object overview page. If a parent child relationship exists, then the parent object is listed on the child’s Object Overview page in the Object Information region. A child object can have only one parent object.
Relationships that are implicitly created from parent child relationships are also viewable on the Relationships page. The relationship name is automatically generated for you.

- Choice list relationship

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

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

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.

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 from the system, 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.

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 viewable 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 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.

- Reference relationship

Instead of using a dynamic choice list to implicitly create a relationship between two objects, you can also manually create this relationship as a reference relationship.
Reference relationships are explicitly created between two top-level objects using the Create Relationships page. View reference relationships on an object’s Object Overview page. If a reference relationship exists, then the related object is listed on the object’s Object Overview page in the Related Objects region.

Using our previous example, perhaps you don’t need to display an HR Representative choice list on a department desktop page, but you still want to add a department subtab to an employee’s details page. In this case, manually create a reference relationship between the employee and department objects where the employee is the source object and the department is the target object. This enables the creation of the department subtab. Such a reference relationship, however, does not automatically create a corresponding HR Representative choice list for use on the department desktop page. In fact, once you manually create a relationship, you cannot reuse the relationship to create a choice list. This means that you should carefully consider the need for a choice list before you create a reference relationship.

- Standard relationship
  Standard relationships are relationships that are already created between two standard objects by the Oracle Sales Cloud application. You can also view standard relationships on an object’s Object Overview page. If a standard relationship exists, then the related object is listed on the object’s Object Overview page in the Related Objects region.

### Creating Reference Relationships

Create a foreign key-based, one-to-many relationship between two top-level objects explicitly using the Create Relationship page. Explicitly created relationships are also known as reference relationships. You can also create a foreign key-based, one-to-many relationship by creating child objects and dynamic choice lists. These implicit relationships are discussed in related topics.

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. Common components, such as notes, interactions, or tasks, are not available for selection as either source objects or target objects.

In general, you create a relationship between two objects within the same application. You can, however, select common objects as target objects. Common objects include:
- Account
- Address
- Customer Contact Profile
- Resource

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 in the system.

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, or as a tree node.

**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.

**Adding Subtabs or Tree Nodes**

After you create relationships between objects, you can then expose the “many” objects on a subtab or tree node that is displayed on the “one” object’s details page or tree.
When adding a subtab or a tree node to an object’s details page or object, you select to add a Child or Related Objects subtab from the object’s Pages Overview page. Application Composer lets you add a subtab or tree node based on any target object that has a relationship with the current object as the source object. Subtabs and tree nodes are discussed in related topics.

**Many-to-Many Relationships**

Objects can also have a many-to-many relationship. 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 employee objects, where the related records from both objects store their primary identifiers in an intersection table.

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

1. Create a child object of one object, and use this child object to represent the intersection table that stores the record identifiers of both objects.
   
   For example, create a service request member object as a child of the service request object. The service request member object’s table records the service request identifier as a foreign key.

2. Add a dynamic choice list for the new child object whose related object is the other object in the many-to-many relationship.
   
   For example, create a dynamic choice list, Support Representative, for the service request member object where the choice list’s related object is employee. Application Composer automatically creates the underlying relationship for you, where the employee is the source object and the service request member is the target object. The service request member object’s table records the employee identifier as a foreign key.

Now, the service request member object’s table records two foreign keys: one for the service request object and the other for the employee object. This enables the many-to-many relationship. You can now do the following:

- Create a child subtab on a service request’s details page. The subtab displays all employees that are working on a specific service request.
- Create a related object 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.

**Defining 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. A standard object has a set of standard fields. Standard fields are the fields that are delivered for a standard object in a Sales Cloud application. 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. 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.

Review these aspects of editing fields in Application Composer before you begin to customize or extend your Sales Cloud application’s object model:

• Adding fields to objects
• Deleting fields

Adding Fields to Objects

Use the Fields page to review the list of standard and custom fields for an object, and create custom fields. A Sales Cloud object can have a maximum of 625 fields.

To view the Fields page for an object:

1. Select an application from the Application choice list on the main Overview page.
2. Select either the Standard Objects or Custom Objects node in the object tree to expand the list of objects.
3. Select the object itself to further expand the tree hierarchy.
4. Select the Fields node to navigate to the Fields page.

On the Fields page:

• Standard Fields
  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 specific to an object, as well as system fields, which could include:
  • CreatedBy
  • CreationDate
  • Id
  • LastUpdateDate
  • LastUpdatedBy
  • RecordName
  The custom objects that you create also contain these same system fields, among others.

• Custom Fields
  Review the list of custom fields that were created specifically for your Sales Cloud implementation for either standard or custom objects, and create new custom fields.
  To create a custom field, select the New icon from the custom fields table on the Fields page. 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

Deleting Fields
Application Composer does not support the deletion of either standard or custom fields from objects. If you no longer need a field that was already published to the mainline code, optionally enter a note in the field description that the field is no longer used.

Field Types and Field Properties: Explained

When you create a custom field in Application Composer, you select from a set of standard field types. Each field type has a corresponding set of standard properties. Some properties are unique to a specific field type, whereas other properties are common across field types. For example, for all field types, you must specify a display label for the field to indicate how you want the field to appear on a desktop page.

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 must specify for a field
• How field types work with other components

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

The types of fields that you can create are listed below.

• Check box
  Users can select a check box, indicating a true or false attribute of a record.
• Currency
  Users can enter a currency amount.

• Date
  Users can enter a date, or select a date from a calendar.

• Datetime
  Users can 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,
or both.

• Dynamic choice list
  Users can select from a list of values, populated from another object’s set
of records.

• Fixed choice list
  Users can select from a list of static values, populated from an FND
lookup type.

• Formula
  A formula field is a field that is 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
  Users can enter a combination of letters, numbers, or symbols. This field
type supports 32,000 characters.

• Number
  Users can enter a number in this field.

• Percentage
  Users can enter a percentage. The system automatically adds the percent
sign.

• Record Type
  Users can select from a list of static values, populated from an FND
lookup type. This particular type of field is useful, because you can
associate each choice list value with a role or a page layout.

• Text
  Users can enter a combination of letters, numbers, or symbols. This field
type is limited to 254 characters.

Common Field Properties
When you create a custom field, you first select the field type. For example, are
you creating a check box field, a formula field, or a long text field? 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, whereas 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></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Help Text</td>
<td>Set this property for all field types.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
<td>Set this property for all field types except for check box, date, and datetime fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**
When setting the display width, consider the resolution in use where this field will be displayed on a desktop page. A display width that is too wide will stretch beyond the resolution of the display and result in scroll bars.

Generally, enter a display width of no more than 20 to 25 characters.

**Note**
For fixed choice lists, note that the display width is determined by the length of the longest string in the choice list.
<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
<th>Related Field Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
<td>Set this property for all field types.</td>
</tr>
<tr>
<td>Enter a unique field name, which is for internal use only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The field name is automatically populated based on the field label you enter, but without spaces.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field names can contain only underscores and alphanumeric characters. They must begin with a letter, not include spaces, not end with an underscore, and not contain consecutive underscores. Field names are limited to 28 characters.</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>Note</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>It is possible to create custom fields with different names, but the same display label. Avoid this scenario, however, so that you do not see two fields with the same display label when configuring a desktop page.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The API name is also automatically generated for a field, by taking the logical name and appending _c. The API name is used in your Groovy scripts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Name</td>
<td>Set this property for all field types.</td>
</tr>
<tr>
<td>Enter a unique field description, which is for internal use only.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Property</td>
<td>Field Property Region</td>
<td>Related Field Types</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Required</td>
<td>Constraints</td>
<td>Set this property for all field types except for formula fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note</td>
<td>Default values are not required for required fields. However you must expose all</td>
</tr>
<tr>
<td></td>
<td></td>
<td>required fields on the object's creation and details (update/edit) pages wherever</td>
</tr>
<tr>
<td></td>
<td></td>
<td>those pages appear (such as on the desktop, simplified, mobile, or Outlook UI). This</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lets your users populate the field at run time. The object's Web services also</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reflect the required fields when your sandbox is published to the mainline.</td>
</tr>
<tr>
<td></td>
<td>Note</td>
<td>If you write an expression to control whether a field is required, then you must</td>
</tr>
<tr>
<td></td>
<td></td>
<td>also configure the Depends On choice list. This choice list includes fields from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>current object, and is used in the evaluation of your expression at run time.</td>
</tr>
<tr>
<td>Field Property</td>
<td>Field Property Region</td>
<td>Related Field Types</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td><strong>Updateable</strong></td>
<td>Constraints</td>
<td>Set this property for all field types except for formula fields.</td>
</tr>
<tr>
<td>Indicate if the field is an updateable field. You can also optionally use the expression builder to write an expression that describes the conditions required for this field to be updateable. This includes being updateable on a desktop page, via Web services, through the import and export functionality, and by server scripts.</td>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>If you write an expression to control whether a field is updateable, 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.</td>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td><strong>Searchable</strong></td>
<td>Constraints</td>
<td>Set this property for all field types except for long text and formula fields.</td>
</tr>
<tr>
<td>Indicate if you want this field to be made available for selection as an additional search criteria from the Add Fields choice list in the Advanced Search mode.</td>
<td>Constraints</td>
<td></td>
</tr>
<tr>
<td><strong>Indexed</strong></td>
<td>Constraints</td>
<td>Set this property for text, number, date, datetime, currency, and percentage field types.</td>
</tr>
<tr>
<td>Enable faster searching by indexing this column.</td>
<td>Constraints</td>
<td>You cannot index long text, formula, and check box fields, or fixed and dynamic choice lists.</td>
</tr>
<tr>
<td>Only a limited number of columns are indexed. For standard objects, 2 varchar extension columns and 3 number extension columns are indexed. For custom objects, 10 varchar columns and 10 number columns are indexed. Accordingly, use this property only on the most frequently searched fields. You cannot change this property after the field is created.</td>
<td>Constraints</td>
<td>You cannot index long text fields. Instead your users can use the Oracle Fusion Applications search capability to search these field types. &lt;&lt;Need confirmation of accuracy of this statement. Pulled from Sean's Fields.doc, where reference was made to &quot;full text search&quot; - what is that?&gt;&gt; [PD] For #2, should be removed for now. This feature will be implemented in Rel9 and we can add in Rel9 document. &lt;&lt;removed 1.17.14.&gt;&gt;</td>
</tr>
<tr>
<td>Field Property</td>
<td>Field Property Region</td>
<td>Related Field Types</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
<td>Set this property for all field types except for formula fields and dynamic choice lists.</td>
</tr>
<tr>
<td>Note</td>
<td></td>
<td>Do not assign a literal default value to fields that are both required and intended to be unique, as a run time error could occur.</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
<td>Set this property for all field types except for check box and formula fields, and fixed and dynamic choice lists. To set default values for these types of fields, write server scripts.</td>
</tr>
</tbody>
</table>

**How Fields 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 most flexibility possible 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, check box, and formula fields.

**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 field type that you can add to a custom or standard object is a check box field. A check box field is a field where users in the run time application can select a check box, indicating a true or false attribute of a record.

**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:
**Field Property** | **Field Property Region**
---|---
Label | Appearance
Help Text | Appearance
Name | Name
Description | Name
Required | Constraints
Updateable | Constraints
Searchable | Constraints
Fixed Value | Default Value

### Additional Check Box Field Specifications
Additional specifications for this field type include the following details:
- Data type is VARCHAR2.
- 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.

### 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 field type that you can add to a custom or standard object is a currency field. A currency field is a field where users in the run time application can enter a currency amount.

**Currency Field Properties**
Create a currency field by specifying values for the common set of field properties, such as display label and field name. You also set properties for this field that are specific to the currency field type.
The following properties are common across multiple field types:
### Field Property Region Table

<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>Updateable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Value</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indicate the minimum numerical value that a user can enter into this field.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Value</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indicate the maximum numerical value that a user can enter into this field.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Exchange Date</td>
<td>Exchange Date</td>
</tr>
<tr>
<td>Optionally specify the exchange date to use to calculate the currency conversion rate.</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td>To use the system date when the record was created as the exchange date, specify the field's creation date as the exchange date.</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Currency 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 those 625 fields, 200 fields are reserved for number, currency, and percentage fields.

**Note**

Each currency field uses two number type columns: one column stores the amount itself and the other column 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, provided that the object’s application allows the creation of currency fields, and are derived from the application’s corporate currency setup.
  - Currency code
    This is the currency code for all currency fields for an object.
  - Corporate currency code
  - 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.

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 field type that you can add to a custom or standard object is a date field. A date field is a field where users in the run time application can enter a date, or select a date 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>
</tbody>
</table>
### 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:

<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>
</tbody>
</table>

### Additional Date 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 those 625 fields, 50 fields are reserved for date and datetime fields.
- When you create a custom subject area to be used 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 field type that you can add to a custom or standard object is a datetime field. A datetime field is a field where users in the run time application can enter a date, or select a date from a calendar, and enter a time of day. You can show the date or time, or both.
<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<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>Updateable</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>
</tbody>
</table>

**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 those 625 fields, 50 fields are reserved for date and datetime fields.
- When you create a custom subject area to be used 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**
• When you are ready to add this dynamic choice list to a page, note that you cannot add dynamic choice lists to the local search region of a custom work area.
• 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>Updateable</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 run time.

- **List Data Source region**

![List Data Source](image)

- **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 related object’s field that you want to expose as a field for your own object. 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**

  The **List Selection Display Value** choice list is the related object’s field that is displayed 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**.
• Data Filter

You can further refine the set of data that appears within the dynamic choice list at run time by using data filters.

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

• 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 from the system, 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 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.

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.

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 field type that you can add to a custom or standard object is a fixed choice list. A fixed choice list is a field that contains a list of static values which are populated from FND lookup types. At run time, your users can select one or more values from this field, depending on the field's definition.

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

The following properties are common across multiple field types:
<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Appearance</td>
</tr>
<tr>
<td>Help Text</td>
<td>Appearance</td>
</tr>
<tr>
<td>Display Width</td>
<td>Appearance</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td>The size of the field depends</td>
<td></td>
</tr>
<tr>
<td>on the longest value of the</td>
<td></td>
</tr>
<tr>
<td>strings in the choice list.</td>
<td></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>Updateable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td>You cannot set a default value</td>
<td></td>
</tr>
<tr>
<td>for any fixed choice list that</td>
<td></td>
</tr>
<tr>
<td>is constrained by another fixed</td>
<td></td>
</tr>
<tr>
<td>choice list.</td>
<td></td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
</tr>
<tr>
<td>If this choice list allows</td>
<td></td>
</tr>
<tr>
<td>multiple values, you can still</td>
<td></td>
</tr>
<tr>
<td>write an expression to</td>
<td></td>
</tr>
<tr>
<td>preselect multiple values by</td>
<td></td>
</tr>
<tr>
<td>default. For example, if the</td>
<td></td>
</tr>
<tr>
<td>field is comprised of three</td>
<td></td>
</tr>
<tr>
<td>lookup codes with (Code, Label)</td>
<td></td>
</tr>
<tr>
<td>of (S, Small), (M, Medium),</td>
<td></td>
</tr>
<tr>
<td>(L, Large), (XL, Extra Large),</td>
<td></td>
</tr>
<tr>
<td>then to preselect the Small and</td>
<td></td>
</tr>
<tr>
<td>Extra Large selections by</td>
<td></td>
</tr>
<tr>
<td>default, set the default value</td>
<td></td>
</tr>
<tr>
<td>to the literal string (without</td>
<td></td>
</tr>
<tr>
<td>quotes): S,XL.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Field Property Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Type</td>
<td>Appearance</td>
</tr>
<tr>
<td>Indicate if your users can select</td>
<td></td>
</tr>
<tr>
<td>only one value, or multiple values,</td>
<td></td>
</tr>
<tr>
<td>from the choice list at run time.</td>
<td></td>
</tr>
<tr>
<td>Selecting the display type is</td>
<td></td>
</tr>
<tr>
<td>possible only during field</td>
<td></td>
</tr>
<tr>
<td>creation.</td>
<td></td>
</tr>
<tr>
<td>Lookup Type</td>
<td>List of Values</td>
</tr>
<tr>
<td>Constrain List by Parent Field</td>
<td>List of Values</td>
</tr>
<tr>
<td>Value Selection</td>
<td></td>
</tr>
</tbody>
</table>
Using the List of Values Region

The values in a fixed choice list are populated from FND lookup types. Select the lookup type whose values you want to display in this choice list. Selecting the lookup type is possible only during field creation.

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

The set of FND lookup types that are available for selection is constrained to those lookup types that are 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.

You can constrain the actual values that display in this fixed choice list at run time by relating this 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 choice list for specifying the trouble ticket type and one choice list for specifying the trouble ticket area.

If a user selects Hardware from the Type choice list, then the Area choice list should contain a list of only hardware options against which the trouble ticket can be logged.

To do this, select the Constrain List by Parent Field Value Selection check box, select the parent field, and then map the values between the parent lookup type and this field’s lookup type.
Note

The Constrain List by Parent Field Value Selection check box is available for selection only during field creation, and only if at least one other fixed choice list, which is a single-select choice list, has been defined.

After field creation, however, you can update the mapping between lookup values.

To implement the previous example:

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

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).
- 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.

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 field type that you can add to a custom or standard object is a formula field. A formula field is a field that is calculated in the run time Sales Cloud application using the Groovy-based expression included in the formula field’s definition. For example, write an expression to calculate an employee’s annual bonus amount.

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 only 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>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Display Type</td>
<td>Appearance</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Depends On</td>
<td>Constraints</td>
</tr>
</tbody>
</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.

For example, if your expression calculates the value of an employee’s annual bonus amount, then you could set the expression to automatically recalculate that amount if the employee’s salary changes. Whenever the salary changes, the bonus field immediately reflects the new bonus amount without your users having to refresh the employee’s record.

In another example, if your expression determines the right customer phone number to use for an opportunity, then you could set the expression to automatically reset the phone number if the opportunity’s customer account changes. Whenever the customer account changes, the phone number field immediately reflects the new phone number without your users having to refresh the opportunity record.

**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. This means that 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 which does not persist in the database as a table column. Hence, there is no maximum number of formula fields for an object.
Joins and Join Fields: Explained

A join is a relationship between an object and its related object. Joins let you display a related object’s fields on an object’s work area. Before you can add related object fields to an object’s work area, however, you must register those fields, either custom or standard, by creating join fields. (Join fields are not provided automatically for you.) For example, the Sales Account object and the Household object are related objects by default and are already delivered with a join. You can display Household object fields in the Sales Account user interface, but only if you first register those fields as join fields for the Sales Account object.

Joins

Joins are view links to related top-level objects in Application Composer. Joins are delivered by default for these objects:

- Opportunity
- Partner
- Product Group
- Sales Account
- Sales Lead Contact

To view the joins available for an object, expand an object and click the Joins node. For example, select the Customer Center application, expand the Sales Account object, and click the Joins node.

Tip

- Click the join name to navigate to the Join Fields page, where you can register join fields.
- Select the join row and click Edit to navigate to the read-only Join Specification page, where you can review details about the join.

Note

You cannot create joins or edit existing joins.

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.

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.

Note

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.

Refer to the worked example related topic for a detailed procedure that describes how to register join fields for the Sales Account object and add those fields to the Customer Overview.
Note
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.

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.

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 Customer Center 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.

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 field type that you can add to a custom or standard object is a long text field. A long text field is a 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>Updateable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including 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 if you want this text field to render in the run time application as a simple text box. Or, indicate if the field should allow 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 those 625 fields, 25 fields are reserved for long text fields.
- The long text field type is not supported by custom subject areas. This means that you cannot add long text fields to a custom report.

**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 field type that you can add to a custom or standard object is a number field.
A number field is a 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>Updateable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

The following properties are unique to only certain field types, including 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>

Specify how many numbers can be entered and displayed to the right of the decimal point. If at run time, a user enters more numbers 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.
### Field Property Region

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Maximum Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify how many numbers a user can enter in the field. This number should be greater than or equal to one and less than or equal to 38. During field creation, consider how this property interacts with these other field properties:</td>
<td>Constraints</td>
</tr>
<tr>
<td>Display Width</td>
<td>If you set a maximum length that is longer than the display width, then users have to scroll inside the field at run time to see the number in this field.</td>
</tr>
<tr>
<td>Decimal Places</td>
<td>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.</td>
</tr>
<tr>
<td>Minimum Value</td>
<td>Indicate the minimum numerical value that a user can enter into this field.</td>
</tr>
<tr>
<td>Constraints</td>
<td>Constraints</td>
</tr>
<tr>
<td>Maximum Value</td>
<td>Indicate the maximum numerical value that a user can enter into this field.</td>
</tr>
</tbody>
</table>

### Additional Number 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 those 625 fields, 200 fields are reserved for number, currency, and percentage 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 field type that you can add to a custom or standard object is a percentage field. A percentage field is a 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>Updateable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Searchable</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indexed</td>
<td>Constraints</td>
</tr>
<tr>
<td>Fixed Value</td>
<td>Default Value</td>
</tr>
<tr>
<td>Expression</td>
<td>Default Value</td>
</tr>
</tbody>
</table>

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

<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>

Specify how many numbers can be entered and displayed to the right of the decimal point. If at run time, a user enters more numbers 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.
### Field Property Region

<table>
<thead>
<tr>
<th>Field Property</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Length</td>
<td></td>
</tr>
<tr>
<td>Specify how many numbers a user can enter in the field.</td>
<td></td>
</tr>
<tr>
<td>During field creation, consider how this property interacts with these other field properties:</td>
<td></td>
</tr>
<tr>
<td>Display Width</td>
<td></td>
</tr>
<tr>
<td>If you set a maximum length that is longer than the display width, then users will have to scroll inside the field at run time to see the amount in this field.</td>
<td></td>
</tr>
<tr>
<td>Decimal Places</td>
<td></td>
</tr>
<tr>
<td>Maximum Length - Decimal Places = the number of digits that can appear to the left of the decimal point.</td>
<td></td>
</tr>
<tr>
<td>Do not set a maximum length that is shorter than the number of decimal places.</td>
<td></td>
</tr>
</tbody>
</table>

#### 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 those 625 fields, 200 fields 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 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 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 particular type of field is useful, because you can associate each choice list value with a role or a page layout.

### Why Use a Record Type Field?

Create a record type field, so that 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.

#### Note

You can create only one record type field per object.

You can:

- Associate each choice list value with a role. You do this while you are creating the field.
  At run time, the field’s available list of values are constrained according to the user’s role.
• Associate each choice list value with a page layout. You do this by adding the field to a simplified page layout, after you have created the field.

At run time, the system assesses the record type value selected by your user to determine which layout to display.

**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.

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

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

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

**Available Record Types**

Indicate the record type values that each enterprise duty role has access to.

For example, perhaps the sales representative can see only selected record type values, but the sales manager can see all the record type values.
### Using the Application Composer

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 text field. A text field is a 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>
</tbody>
</table>

### 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. Selecting the lookup type is possible only during field creation.

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

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

### Additional Record Type Field Specifications

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500).
- 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. A record type field is not required for an object, however.
- You can create a record type field, only if the object already has a work area. If the object's work area has not yet been created, then you must create the work area first, before you can create the record type field.
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>Indicate if you want this text field to render in the run time application as a simple text box. Or, indicate if the field should allow multiple lines where text can wrap, or where the user can enter carriage returns.</td>
<td></td>
</tr>
<tr>
<td>Maximum Length</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indicate 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 characters against the total.</td>
<td></td>
</tr>
<tr>
<td>Minimum Length</td>
<td>Constraints</td>
</tr>
<tr>
<td>Indicate the minimum number of characters that a user can enter into the field.</td>
<td></td>
</tr>
</tbody>
</table>

**Additional Text Field Specifications**

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500 char).
- A object can have a total of 625 fields. Out of those 625 fields, 350 fields are reserved for text and check box fields, and fixed and dynamic choice lists.
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 also expose new objects and fields in mobile applications using the Mobile Pages tab. And, you can expose new fields in an alternative set of simplified user interface pages, if available for a standard object, using the Simplified Pages tab.

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.

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.

Note
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.

**Tip**
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 common components (note, interaction, task) 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.
3. Select the Pages node.
Note

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

**Note**

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 local search is formatted as a region with two 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 first column again, and so on.

**Note**

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.

2. 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
1. Select the fields that you want to display as columns in the summary table.

2. Add custom actions or links to the summary table, if you previously created them.

- Creation Page

a. Select the fields that you want to display on the object's creation page.

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](image)

a. Select the fields that you want to display on the object's details page, including both the default summary and detailed summary regions.

Tip

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. Select the Allow Attachments check box to enable the attachments feature on the run time details page, in the collapsible detailed summary.

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. 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.

Note

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 following figure shows the Search and Select picker dialog box at run time.

![Search and Select picker dialog box](image)

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.
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.

**Note**

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 Application Composer duty 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 cannot use Page Composer to edit those pages again.

**Note**

The exception is the customer profile in Customer Center. You can create and add new fields to the Sales Account region on the customer profile using Page Composer.

**Creating a Work Area: Explained**

When first created, custom objects do not 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 desktop pages, also known as a work area, for that object. Every Sales Cloud object can be displayed on a work area, which consists of an overview page, a creation page, and a details page. Application Composer employs a wizard approach to walk you through the creation of that object’s work area. After you create the initial work area, you can always navigate to the object’s Pages Overview page to continue to customize those desktop pages using work area configuration pages. You do not create a work area for child objects. 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.

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

**Using 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 has not yet been created for an object, then hyperlinks to the work area configuration pages are not displayed. Instead, the Pages Overview page displays only a single hyperlink to launch 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.

**Note**

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.

4. Select the hyperlink to launch 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.

**Configuring 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.
On this page, you can also do the following:

- 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 your having to reauthenticate.

---

**Note**

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.

---

**Extending 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: Set a field as required on the object search region. This field must be populated.

- At Least one is required: Set at least one field as a required field or a collection of required fields. You must populate one field from this group.

- Default Operator: Set an operator field that appears in object search as the default.

**Configuring the Local Search Region**

The local search region appears above the summary table on an object's overview page. 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 to you in a single column, although the local search region is formatted as a region with two columns.

**Note**

You cannot control the order in which the chosen fields are displayed in the local search region.

**Note**

During field creation, consider indexing any fields that you plan to display as search criteria for your custom objects.
Configuring the Overview and Creation Pages

Select the fields that you want to display in the work area’s overview page, and in the object’s creation 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 drilldown column for the summary table.
   The drilldown column is the column in the summary table that users can click to drill down to an object record’s details page. You cannot change a summary table’s drilldown column after the work area is created.

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 should 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

Select the fields that you want to display on the object’s details page, also known as the edit page.
Note
A details page can have subtabs, which include information that is related to the object record. For example, the details page for an opportunity could include a subtab that lists customer contacts or previous orders. 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 page is discussed in a related topic.

1. Select the fields that you want to display on the object’s details page, including both the default summary and detailed summary regions.

Tip
Include the primary object fields in the default summary, since the detailed summary could 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 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.

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

3. Select the Allow Attachments check box to enable the attachments feature on the run time details page, in the collapsible detailed summary.

Subtabs: Explained

Every top-level Oracle Sales Cloud object has a details page as part of its work area. When configuring the details page, you can optionally 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. Add subtabs to a standard or custom object’s details page from that object’s Pages Overview page in Application Composer. This functionality is available for Simplified Pages also.

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
- Subtab types:
  - Child or related object subtabs
  - Context link subtabs
  - Common component subtabs
• Web content subtabs

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

For custom objects, only subtabs are supported.

For standard objects that are already using tree nodes, such as the Sales Account Profile and Partner objects, 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 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 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.

Adding Subtabs

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.

To add a subtab to an existing details page:

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.

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

4. On the Pages Overview page, click the Create Subtab icon in the Details Page region to create one or more subtabs to display on the object’s details page.
5. Select the type of subtab you want to add.
To hide subtabs that you previously added using Application Composer, use Page Composer.

Adding Subtabs to Simplified Pages

Add a subtab for a standard object user interface directly from the Simplified Pages tab, Details Page Layout region.

To add a subtab, navigate to the Create Subtab page and enter the details:

1. Select an application in Application Composer.
2. Select a standard object in the object tree.

Note

You cannot add subtabs to a custom object for Simplified pages.

3. Select the Pages node.
4. Click the Simplified Pages tab in the local area.
5. Click the Edit icon in the Details Page Layout region.
6. In the Edit Simplified Details page, click the Create icon at the bottom of the vertical tabs tray. The Create Subtab page appears.

This figure shows the Create Subtab page.
Child or Related Object Subtabs

A relationship is a foreign key association between two objects. Using Application Composer, you can create a one-to-many relationship between two objects within the same application. Once relationships are created, you can expose the "many" objects on a subtab that is displayed on the "one" object’s details page. 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. In this example, the account is the source object and the service request is the target object. This relationship adds the account identifier to the service request object's table.

Application Composer lets you add a subtab to an object’s details page for either a child object or for three types of related objects. These objects exist in four types of one-to-many relationships:

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

  For example, to enable the creation of a subaccount subtab on an account’s details page, you would create the subaccount object as a child of the account object. This relationship adds the account identifier to the subaccount object’s table.

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

  For example, to enable the creation of a department subtab on an employee’s details page, you would create a dynamic choice list, HR Representative, for the department object where the choice list’s related object is employee. Application Composer automatically creates the underlying relationship for you, where the employee is the source object and the department is the target object. This relationship adds the employee identifier to the department object’s table, thus enabling the creation of a department subtab on an employee’s details page. The subtab displays all departments that an HR representative can manage, since each HR representative can be in charge of multiple departments of a company.

- **Reference relationship**
  Reference relationships are explicitly created between two top-level objects using the Create Relationships page.
Using our previous example, perhaps you don’t need to display an HR Representative choice list on a department desktop page, but you still want to add a department subtab to an employee’s details page. In this case, manually create a reference relationship between the employee and department objects where the employee is the source object and the department is the target object. This enables the creation of the department subtab. Such a reference relationship, however, does not automatically create a corresponding HR Representative choice list for use on the department desktop page. In fact, once you manually create a relationship, you cannot reuse the relationship to create a choice list. This means that you should carefully consider the need for a choice list before you create a reference relationship.

- Standard relationship

Standard relationships are relationships that are already created between two standard objects by the Oracle Sales Cloud application.

To add a child or related object subtab to an existing details page:

1. On the Pages Overview page, click the **Create Subtab** icon.

2. Select **Child or Related Object**.

3. On the Child or Related Object subtab configuration page:
   a. Select the related object from the list of all related objects that is to be exposed on the subtab, and choose the subtab display label.
   
   b. Optionally hide the New and Delete buttons that appear on the subtab at run time.

   For child object subtabs, you can also optionally hide the Edit button.
c. Select which fields and links you want to display on the subtab 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.

d. Select which buttons you want to display on the subtab at run time.

Note

This region appears only if you previously created buttons for this object.

e. Select which fields you want to display on the subtab detail form at run time.

You can configure fields for the detail form that appears under the summary table. If the subtab'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 subtab's object is a related object, then users can associate an existing record of the subtab 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 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. On the Pages Overview page, click the Create Subtab icon.
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.

   **Tip**
   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. 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.

**Common Component Subtabs**
A common component subtab adds a Notes, Tasks, Interactions or Appointments subtab to show a list of the selected components related to a custom, top-level
object. Each component has a standard user interface (UI) that is shared across all Oracle Sales Cloud applications. To customize such a UI for all common components (other than Appointments), select the appropriate 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, users can access these subtabs and create a common component record that is tied to the object record. For example, a user can record a customer interaction on a service request record.

- Notes
- Tasks
- Interactions
- Appointments

To add a common component subtab to an existing details page:

1. On the Pages Overview page, click the Create Subtab icon.
2. Select Common Component.
3. On the Common Component subtab configuration page, select the type of common component you want to add to the details page as a subtab.

**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 Google map that shows the location of the contact. The Google 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 Google Maps API.

To add a Web content subtab to an existing details page:

1. On the Pages Overview page, click the Create Subtab icon to create one or more subtabs to display on the object’s details page.
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.

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:
```
def myURL1 = adf.util.GlobalEncodeField(ContactAddress_c)
def myURL2 = adf.util.GlobalEncodeField(ContactCity_c)
def myURL3 = adf.util.GlobalEncodeField(ContactState_c)
def myfinalURL = "http://maps.google.com/maps?hl=en&q=" + myURL1 + "+" +
  myURL2 + "+" + myURL3
return(myfinalURL)
```

Tree Nodes: Explained

Some Oracle Sales Cloud standard objects, such as the Sales Account and Partner
objects, use a tree to display its related 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.

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 or related object tree nodes
  • Context link tree nodes
  • Web content tree nodes

---

**Note**

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

For custom objects, only subtabs are supported.

For standard objects that are already using tree nodes, such as the Sales Account and Partner objects, 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, either the Sales Account or Partner object, in the object tree.
3. Select the **Pages** node.

---

**Note**

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

---

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 between two objects within the same application. 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 expose a list of contacts associated with a specific partner as a tree node on the partner’s tree, you must first create a one-to-many relationship between the partner and contact objects. In this example, the partner is the source object and the contact is the target object. This relationship adds the partner identifier to the contact object’s table.

Application Composer lets you add a tree node to an object’s tree for either a child object or for three types of related objects. These objects exist in four types of one-to-many relationships, which are described in detail in the related topic about object relationships:

- Parent child relationship
- Choice list relationship
- Reference relationship
- Standard relationship

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 **Child or Related Object**.
3. On the Child or Related Object tree node configuration page:

   a. Select the tree node category and enter the tree node label.
   b. Select the related object from the list of all related objects 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.

   g. Select which fields you want to display on the tree node page's detail form at run time.

   **Note**

   This region appears only if you previously created buttons for this object.

   You cannot add buttons to a tree node page for the Sales Account object.
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. 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, perhaps you want to add a Google map that shows the location of the contact. The Google 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 Google 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 myURL1 = adf.util.GlobalEncodeField(ContactAddress_c)
def myURL2 = adf.util.GlobalEncodeField(ContactCity_c)
def myURL3 = adf.util.GlobalEncodeField(ContactState_c)
def myfinalURL = "http://maps.google.com/maps?hl=en&q=" + myURL1 + "+" + myURL2 + "+" + myURL3
return(myfinalURL)
```

**Actions and Links: Explained**

In Oracle Sales Cloud applications 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.

**What are 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. You 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.
Adding Actions or Links

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.

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.

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.

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:

- 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

## Saved Searches for Oracle Sales Cloud Objects: Explained

A saved search is a predefined set of search criteria which users can apply at run time to a standard or custom object’s overview page. The overview page provides a list of records for an object in a summary table, which is the starting point in an Oracle Sales Cloud application for users to view and manage data. At run time, users can select a saved search from the Saved Search choice list to constrain the list of records that appear in the summary table.

### Important

In previous releases, Application Composer provided you with the ability to create and edit saved searches, available to your users at the site level. In the current release, you can only edit existing saved searches, since Page Composer also lets you create and edit saved searches at the site level.

Review these aspects of the saved search definition process in Application Composer before you begin to view or edit saved searches for Sales Cloud objects:

- Saved searches at run time
- Editing a saved search

### Saved Searches at Run Time

The list of saved searches is available from the **Saved Search** choice list above the overview page’s summary table.

Searches are displayed in alphabetic order, followed by the Personalize option.

The saved searches that you edit for an object are available to all users and roles with functional security access to the object’s overview page.

### Editing a Saved Search

You edit a saved search for an object on the object’s Edit Saved Search page. You can edit saved searches only for top-level objects, because only top-level objects
have overview pages in the run time Sales Cloud application. To create a new saved search, use Page Composer.

To edit a saved search 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 Saved Searches node.
4. Select the Saved Search that you want to edit from the summary table in the Saved Searches page and select Actions > Edit
5. On the Edit Saved Search page for the object, you can modify the display label for the saved search.
   The display label is the label that appears in the Saved Search choice list.
6. Modify the description, if required.
   You cannot edit the Name field.
   The Name is automatically generated based on the specified display label at the time of creating the saved search. Spaces in the display label are converted to underscores in the Name. For example, if the display label is Top Ten Opportunities, then the Name is automatically generated as Top_Ten_Opportunities at the time of creating the saved search.
7. Modify the search criteria.
   You can enter up to four search conditions using the following criteria:
   • Specify the matching criteria, All or Any.
     • All appends multiple conditions with an AND.
     • Any appends two conditions with an OR.
     If more than two conditions exist, then you cannot select Any.
   • Field Name
     Specify the object field who value you want to include as a search criterion.
     Long text and formula fields are not available for use in saved searches.

Tip
For better performance, the fields you include in a saved search should be indexed. You index a custom field when it’s first created.

• Operator
  Operator values available for selection are dependent upon the type of field selected.
This table lists, for each field type, the operators that are available for selection, as well as the values that you can enter.

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Available Operators</th>
<th>Available Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Equal to</td>
<td>Enter a literal value.</td>
</tr>
<tr>
<td></td>
<td>Greater than</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greater than or equal to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is blank</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is not blank</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Less than</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Less than or equal to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not equal to</td>
<td></td>
</tr>
<tr>
<td>Check box</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Percentage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic choice list</td>
<td>Contains</td>
<td>Enter a literal value.</td>
</tr>
<tr>
<td>Text</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed choice list</td>
<td>Does not contain</td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ends with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Equal to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is blank</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Is not blank</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not equal to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Starts with</td>
<td></td>
</tr>
</tbody>
</table>

**Note**
For percentage fields, divide the percentage by 100 to calculate the literal value you should enter as a search criterion. For example, to correctly use 150 percent in your saved search, you must enter 1.5.

**Note**
For fixed choice list fields, you must enter the lookup code, not the lookup meaning.
<table>
<thead>
<tr>
<th>Field Type</th>
<th>Available Operators</th>
<th>Available Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>After</td>
<td>Enter:</td>
</tr>
<tr>
<td>Date time</td>
<td>Before</td>
<td>Literal value</td>
</tr>
<tr>
<td></td>
<td>Equal to</td>
<td>Date function</td>
</tr>
<tr>
<td></td>
<td>Is blank</td>
<td>Specify the number of days, months, or years after or before the current date.</td>
</tr>
<tr>
<td></td>
<td>Is not blank</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Not equal to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On or after</td>
<td></td>
</tr>
<tr>
<td></td>
<td>On or before</td>
<td></td>
</tr>
</tbody>
</table>

- **Value**
  
The values that you specify are applied as the search criteria against the object records at run time.
  
  See the previous table.

### Making Saved Searches Available to All Users: Procedures

You can use Page Composer to create saved searches that appear to all users at the site layer. In Oracle Sales Cloud simplified pages, saved searches are also known as saved lists.

**Note**
Create saved searches for all users at the site layer only. Do not use other layers, such as the role layer.

### Creating Saved Searches for All Users

To create site-level saved searches:

1. Navigate to the search component.
2. Under your Settings and Actions, click **Customize <Page Name> Pages** to open Page Composer.
3. Select the Site layer.
4. In Design view, enter the search criteria.
5. Click **Search**.

**Note**
You must perform the search before you save.

6. Save the search component.
7. Name your search and optionally select from these options:
   - **Set as Default** - the search criteria appear when users view the search component.
   - **Run Automatically** - the search criteria and results appear when users view the search component.

8. Close the dialog box.

9. Save and close Page Composer.

10. Sign out and sign in again to refresh the current list of saved searches.

**Editing Saved Searches for All Users**

To delete, rename, or change the search options for a site-level saved search, perform the following steps:

1. Navigate to the search component.

2. Under your Settings and Actions, click **Customize <Page Name> Pages** to open Page Composer.

3. Select the Site layer.

4. In the Design view, select Personalize from the Saved Search list of values.

5. Select a saved search, then you can:
   - Delete
   - Rename
   - Change search options

---

**Note**

You must perform the search before you save.

---

**Note**

You can’t change the search criteria of a saved search in Page Composer. You can delete and then re-create it if you need to change the search criteria.

6. Save the changes and close Page Composer.

7. Sign out and sign in again to refresh the current list of saved searches.

**Securing Custom Objects: Explained**

After you create custom objects and fields, you then expose them on desktop pages for your users. Your next step is to control which users can access that object’s data at run time. By default, the object and its records are visible and editable only to a default duty role specified by the application. Grant additional access manually using Application Composer’s security policy configuration pages. A security policy specifies which users are authorized to access an object’s
data, and what type of access they have. For example, does a user have view only access, or can the user create and update an object's record, as well? Define security policies for an object by authorizing the roles whose users can access that object's data. Or, define security policies for a role by specifying access levels across multiple custom objects.

Review these aspects of the custom object security process in Application Composer before you begin to define your security policies:

- Securing objects
- Securing roles
- Enabling function security and data security
- Application Composer and the Oracle Authorization Policy Manager (APM)
- Default security settings

Securing Objects

The object-centric Define Policies page displays a list of the enterprise-level duty roles which map to an Oracle Sales Cloud job role. Use this page to manage access to either a top-level or child custom object by specifying a security policy for one or more duty roles. When you do this, users with the corresponding 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. Select an application on the main Overview page.
2. Select a custom object in the object tree.
3. Select the Security node.

Or, from the role-centric Define Policies page, select a custom object.

![Roles Table](image)

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

- Enable function security for a role.
- Enable data security for a role.

Securing Roles

The Role Security page displays a list of the enterprise-level duty roles, which map to a Sales Cloud job role. Select a 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 role by specifying a security policy for
one or more top-level or child custom objects. When you do this, users with the corresponding 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. Select an application on the main Overview page.
2. Select the **Role Security** node from the Common Setup pane.
   Or, select the **Role Security** hyperlink in the local area of the main Overview page.
   Or, from the object-centric Define Policies page, select a role.

3. Click the **Define Policies** button.

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

- Enable function security for a custom object.
- Enable data security for a custom object.
- 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.

**Enabling Function Security and Data Security**

A security policy specifies the type of access to an object and its records that users with the corresponding roles have. Access includes both function security as well as data security. Security settings are the same whether you are defining a security policy for an object or a role.

On the Define Policies 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 two 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.

Tip

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

Note

To let users view or update 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.

**Application Composer and the Oracle Authorization Policy Manager (APM)**

Oracle Authorization Policy Manager (APM) 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 APM.

Since you define the security policies outside APM, you cannot later modify the security policies within APM. Instead, modify all security policies for custom objects using only Application Composer.

**Default Security Settings**

By default, top-level custom objects are visible and editable only to users with a default duty role specified by a Sales Cloud application. You must manually grant additional access to other duty roles, if desired. For example, if you create a custom object in the Sales application, then only users with the default duty role
specified by Sales are automatically granted access to that object. This lets you first access an object and its pages for testing, before you officially grant access to your customizations to users in a production environment.

This table lists the default duty roles that are provided by Oracle Sales Cloud:

<table>
<thead>
<tr>
<th>Application</th>
<th>Default Duty Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Center</td>
<td>Sales Administrator Duty</td>
</tr>
<tr>
<td></td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td>Marketing</td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td>Sales</td>
<td>Sales Administrator Duty</td>
</tr>
</tbody>
</table>

Child objects do not inherit security settings from parent objects. Rather, if you create a custom child object, then a default set of duty roles are granted access to the child object. In other words, a child object is visible and editable only to users with these default duty roles, as follows:

<table>
<thead>
<tr>
<th>Application</th>
<th>Child Objects of This Parent Object</th>
<th>Access Granted to These Duty Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Center</td>
<td>Sales Account</td>
<td>Sales Administrator Duty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td>Marketing</td>
<td>Sales Lead</td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sales Administrator Duty</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Channel Operations Manager Duty</td>
</tr>
<tr>
<td>Marketing</td>
<td>Marketing Campaign</td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td></td>
<td>Marketing Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing List</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing Treatment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing Event Activity</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing Advertising Activity</td>
<td></td>
</tr>
<tr>
<td>Marketing</td>
<td>Marketing Budget</td>
<td>Marketing Operations Manager Duty</td>
</tr>
<tr>
<td></td>
<td>Marketing Claim</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing Budget Entry</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing Budget Fund Request</td>
<td></td>
</tr>
</tbody>
</table>

Child operations
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.

**Oracle Sales Cloud Import and Export Processes**

In Oracle Sales Cloud, two processes exist to enable the importing and exporting of object data: 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.

Bulk export lets you 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. The system creates comma or tab-delimited files 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 object model extensions that you make using Application Composer do not create the artifacts required by these import and export processes. For example, file-based import leverages Oracle Data Integrator (ODI).

Accordingly, after completing your object model extensions, generate the required artifacts to register your extensions and make them available for importing and exporting.

**Note**
The creation of import and export artifacts occurs only in the Oracle Metadata Services (MDS) mainline code, and is not supported within the MDS sandbox.

To enable the import and export of custom object data:

1. Select an application on the main Overview page.

2. Select the Import and Export link in the Common Setup pane, or in the local area of the main Overview page.
3. On the Import and Export page, click the **Generate** button.
After you enable your object model extensions for importing and exporting, you can then schedule your file-based import and bulk export processes using the following tasks, available by selecting **Setup and Maintenance** from the Tools menu and searching on the task name.

- To schedule your custom object imports, select the **Manage File Import Activities** task.
  To initially set up file-based import for importing custom object data, select the **Manage File Import Objects** and **Manage File Import Mappings** tasks.

**Note**
Custom child objects are imported as part of the parent object.

- To schedule your custom object exports, select the **Schedule Export Processes** task.

**Note**
Both top-level and child custom objects are available as independent exportable objects.

**Important**
Refer to Oracle Sales Cloud product-specific documentation for additional details on the import and export of custom object data (custom fields) for standard objects.

---

**FAQs for Using Application Composer**

**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>
<tr>
<td>Make field read-only or updateable</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>
</tbody>
</table>
Use Cases | Design View - Standard mode | Design View - Direct Selection mode | Page Type
--- | --- | --- | ---
Reorder table columns | Yes | Yes | Transactional pages (all non-dashboard pages)
Hide or show table columns | Yes | Yes | Transactional pages (all non-dashboard pages)
Set table column width with the mouse | Yes | No | Transactional pages (all non-dashboard pages)
Set table column width and min width in percent or pixels | No | Yes | Transactional pages (all non-dashboard pages)
Make column sortable or not | No | Yes | Transactional pages (all non-dashboard pages)

Application Composer also lets you make UI changes at run time. However, the types of UI changes that you can make using Application Composer are quite different. Specifically, your primary focus when using Application Composer is to make actual object model changes. For example, you can create a new business object and related fields, and then create new application pages where that object and its fields are exposed to users.

The following table describes some of the primary differences between Page Composer and Application Composer. For example, using Application Composer, you cannot access the Resource Catalog to add new content to a page. With Application Composer, administrators can make customizations at the site level only.

| Customization Task | Available in Page Composer (site, job role, external or internal level)? | Available in Application Composer (site level only)?
--- | --- | ---
Make object model extensions and expose your customizations by creating or modifying work area pages | No | Yes
Reorder subtabs | No | Yes
Customize dashboard pages | Yes | No
Add content from the Resource Catalog | Yes | No
Simple field customizations (show, hide, make read only, make required) | Yes (WYSIWYG - what you see is what you get) | Yes (non-WYSIWYG)
<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>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>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>
Application Composer: Extending Simplified Pages

Extending Simplified Pages: Overview

Read this chapter to learn about the set of simplified pages that are available with Oracle Sales Cloud. Certain standard objects offer a set of simplified pages, some of which you can customize using Application Composer.

In this chapter, you will learn about:
- Which simplified pages are extensible, what you can do, and how to customize simplified pages using Application Composer
- What you can extend on the simplified pages for contacts, customers, leads, assets, and opportunities
- 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
- How you can enable or disable drill down fields

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 a standard 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.

Oracle Sales Cloud Simplified Pages: Explained

Oracle Sales Cloud provides an alternative set of simplified user interfaces for some select standard objects, such as opportunities and leads. These simplified pages are ideal for busy salespeople who want a very streamlined and fast-loading experience when accessing core salesforce automation tasks using their laptops or tablets. For example, with a minimum of mouse clicks or finger strokes, simplified page users can quickly set up appointments and tasks, and
create, view, and edit contacts, accounts/customers, opportunities, and leads. Just as with standard desktop pages, 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. Objects whose simplified pages are extensible have a new tab called Simplified Pages when the Pages node is selected. Business users can customize the pages listed under this Simplified Pages tab.

**Note**

Use of the simplified pages is optional. Sales administrators or other power users who require the full set of application features can continue to use the standard set of Oracle Sales Cloud desktop pages.

In this topic, you will learn which pages are extensible, what you can do, and how to customize simplified pages using Application Composer.

**What is Extensible?**

Certain standard objects offer a set of simplified pages, some of which you can customize.

This table lists the objects that have a set of simplified pages, and which pages are extensible using Application Composer and Page Composer.

<table>
<thead>
<tr>
<th>Object</th>
<th>Available Simplified Pages</th>
<th>Extensible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>Contacts landing page</td>
<td>Yes</td>
</tr>
<tr>
<td>Account</td>
<td>Accounts landing page</td>
<td>Yes</td>
</tr>
</tbody>
</table>
| Forecast Territory Details | Sales Forecasts landing page  
Forecast Items subtab  
Analytics subtab  
Unforecasted Pipeline subtab | The Sales Forecasts landing page and Forecast Items subtab are extensible.  
The remaining subtabs are not extensible. |
| Household               | Households landing page                                         | Yes             |
| Lead                    | Leads landing page  
Lead Contact subtab  
Lead Team subtab         | Yes             |
| Opportunity             | Opportunities landing page  
Opportunity Contact subtab  
Opportunity Team subtab   | Yes             |
| Appointments            | Appointments page                                               | No              |
| Interactions            | Interactions page                                               | Yes             |
What Can You Do?
Use either Application Composer or Page Composer to customize a set of simplified pages, if they are extensible.

Note
With simplified pages, the ability to customize the search region is not available. Simplified pages are available only for a select group of standard Oracle Sales Cloud objects; you cannot create a set of simplified pages for custom objects.

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 a standard 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 are not automatically replicated on the object's corresponding desktop page.

Tip
You can also modify simplified pages using Page Composer. Refer to the Page Composer chapter in this guide to learn more.

Customizing the Springboard for Simplified Pages: Points to Consider

The springboard is the area on the simplified user interface home page, and above all simplified pages, that contains a set of functional area icon buttons. You can define which functional areas and pages within these areas are available to users.

Click Settings in the springboard, and then select the Structure tab. The Structure page displays all the functional areas and pages that are currently selected to appear on the springboard.

Displaying or Hiding Functional Areas

To display functional areas on the springboard, click Add Item and select the functional areas to include.
To hide a functional area, hover over that functional area and click Hide Item.

Displaying or Hiding Pages

To display a page in a functional area, hover over that functional area and click Add Tab, if available.
To hide a page, hover over that page and click Hide Tab.

Changing the Order of Functional Areas and Pages

Drag and drop functional areas and pages to set the order.
You can also use the Organize menu and select:
- Site: To reorder functional areas.
- Any functional area: To reorder pages within that area.

Renaming Functional Areas and Pages

Click the name of a functional area or page to modify its name.

Customizing Simplified Pages Using Page Composer: Points to Consider

On a simplified page, you can click your name and select Customize User Interface to customize the UI using Page Composer. When customizing a
simplified page, consider the customization layer to choose, the types of customizations you can make, and labels for your saved changes.

**Customization Layers**

The customization layer that you select before making changes to the page determines the scope of users impacted by your customizations. If you are not presented with customization layers to choose from after you select **Customize User Interface**, then your changes are made to the site layer. For more information to understand customization layers, see the Oracle Fusion Applications Extensibility Guide for Business Analysts.

**Types of Customizations**

In simplified pages, customization using Page Composer is limited to what you can change with component properties. For example, you can show or hide fields or make a check box required, but you cannot add new components or change the layout of the page.

After you select a customization layer, if available, you can click:

- **Design** to navigate around and get to the components you want to customize. You cannot make any customizations in this mode.
- **Select** to select a component on the page and open its properties.

Each component has its own set of properties, which may include some of the properties in this table.

<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 Required</td>
<td>Whether an asterisk is displayed to indicate that the component is required</td>
</tr>
</tbody>
</table>
When you access component properties on a workstation page using Page Composer, more properties are available.

**Save and Label**

When you save, you can label your changes so that you can later revert to your saved customizations. Labels are stored with a prefix of `composer_`. For example, if you enter `myLabel`, then the label is `composer_myLabel`.

As needed later, you can click your name in the global area and select Manage Customizations. Click Promote for the desired component and select the label to revert to.

**Changing the Look and Feel of Simplified Pages: Points to Consider**

Use the Appearance page to apply predefined themes to all simplified pages, or customize these themes to create your own themes. To open the Appearance page, click Settings in the springboard and then select the Appearance tab.

Use themes to determine:

- The shape of buttons, menus, tabs, and pages
- The style of icon buttons in the springboard
- The branding above every page
- The background of every page

**Branding**

You can define the branding logo and application name above all pages.

---

**Tip**

For the logo, use an image that is as close to 400 by 100 pixels as possible. In general, an image that is wider than it is tall works best.

---

**Background**

The base color and watermark appear in the background of all simplified pages.

---

**Tip**

For the watermark, use an image that is as close to 1024 by 768 pixels as possible.

---

**Creating and Adding Custom Action Links to Simplified Pages: Worked Example**

This worked example demonstrates how to create and add custom action links to simplified pages.
In this example, we will create a custom link that launches the Google homepage and add this link to the Edit Lead page.

Creating a Custom Link
In this example, we will create an action link named Test and add it to the simplified pages for editing a lead.

1. Select a sandbox to work in and make it active.
2. Select the Marketing application in Application Composer.
3. Expand standard objects and click Sales Lead.
4. Click the Actions and Links link.
5. Click the Simplified Pages tab.
6. In the Sales Lead: Actions and Links page, click the Create icon or select Actions - Create.
7. In the Sales Lead: Create Action or Link page enter the following information:
   - Display Label: Test
   - Name: Test
   - Description: Optional
   - Type: Link
   - Source: URL
   This figure shows the Create Action or Link simplified page for the Sales Leads object.

8. In the script region, enter 'http://www.google.com'

Note
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 URL definition region of the Edit Action or Link page.

9. Click the Validate icon in the Show Palette region. Note the Information message that appears.

This figure shows the information message that appears when you click the Validate icon.

10. Click the Save button to save the new link.

11. Next, you must add this new link called Test to the Leads page. Click the Pages link under Sales Lead.

12. Select the Simplified Pages tab

13. Select the Edit Details Page link.

14. In the Edits Details Page select the vertical tab and the region to add to custom link to. In this example, select the Summary tab.

15. Click the pencil Edit icon in the Summary region.

This figure shows the Edit Simplified Details page, Summary tab.
16. In the Edit Lead page, Configure Detail Form region, look for Test link that we created in the Available Fields column.

This figure shows the Edit Lead Page, Configure Detail Items region.

17. Select Test - Link from the Available Fields column and click on the right arrow to move the link to the Selected Fields column at the bottom of the list.

18. Click the Save and Close button to save changes.

**Testing the Link in the Runtime Application**

You must test whether the Test link you created appears in the Simplified Pages runtime application.

1. Open the Simplified Pages for Leads Application

   This figure shows the Simplified Pages home page.
2. Click a lead to edit.

This figure shows the Simplified Pages for Leads.

3. Note the newly created Test link in the Edit Lead page

4. Click the Test link and check whether the Google home page appears.

This figure shows the Google home page that appears when you click the Test link.
Creating and Reordering Subtabs in Simplified Details Pages: Worked Example

This worked example demonstrates how to add subtabs for the standard object User Interface directly from the Simplified Details Page. In this example, we will add a subtab to the Leads Simplified page and also reorder the location of the subtab that we created.

**Creating a Subtab in the Leads Simplified Page**

1. Select a sandbox to work in and make it active.
2. From the Navigator menu, select **Customization - Application Composer**.
3. Select the **Marketing** application.
4. Under the Standard Objects tree, expand the **Sales Lead** object.
5. Click the **Pages** link.
6. Click the **Simplified Pages** tab in the local area.
   - This figure shows the Sales Lead landing page for simplified pages.
8. Click the **Edit** icon.
   - This figure shows the Details Page Layouts region with the Edit Layout option selected.
9. In the **Edit Simplified Details** page, click the + (add) icon at the bottom of the vertical tabs tray.

   This figure shows the Edit Simplified Details page with the Create Icon at the bottom of the subtabs panel.

10. In the **Create Subtab** page, select **Child or related object**.

    This figure shows the Create Subtab page.

11. Click Next.

12. Enter details of the new subtab that you want to create. In this example, we will enter the following:

    - Data Object: Sales Lead
• Display Label: Infodev Subtab
• Drill Down Column: Use default
• Icon: Select the desired icon

13. Select the fields that you want in the subtab from the Available Fields column and move it to the Selected Fields column using arrow buttons. This figure shows the Create Subtab page.

14. Click Next

15. In the Create Subtab: Child or Related Object page, you can optionally double click a layout in the Available layouts column to move it to the Selected Layouts column. In this example, we will not make any changes. Click Save and Close. This figure shows the Select Additional Layouts region in the Child or Related Object page.

The new sub tab(s) appear at the bottom of the vertically stacked tabs in the Edit Simplified Details Page. They should also appear in the Detail tab table for the object.

Reordering Subtabs

You can reorder subtabs from the Edit Simplified Details page.

1. In the Edit Simplified Details Page click the Layout button. This brings the layout editor in Edit Layout Mode and enables the Reorder icon for the sub tabs.

This figure shows the Reorder icon in the Subtabs region
2. Click on the Reorder icon above the sub tabs.

3. The Reorder Subtabs popup window opens. Use the up and down arrows to change the sub tabs order as desired.

   This figure shows the Reorder Subtabs window with the original location of the new subtab we created.

   ![Reorder Subtabs window with original location](image)

4. In this example, we will move our new subtab to the top of the list. Click the Move to Top arrow.

   This figure shows the Reorder Subtabs window with the new location of the subtab we created.
5. Click OK to save changes.


   This figure shows the Edit Simplified Details page, Subtabs region with the new subtab at the top of the subtabs panel.

**Testing the New Subtab in the Runtime Application**

We must check whether the subtab appears correctly in the runtime application.

1. Go to run time Simplified Pages application, and select the Leads application.

2. In the Leads Overview page, select a lead and click on the Infodev Subtab that we created in the previous section.

   This figure shows the runtime Simplified Pages Leads application.
3. Check whether the subtab appears at the top of the list. Click the new subtab icon in the subtabs panel.

4. Verify whether the subtab lists the fields that you had selected while creating the subtab.
   This figure shows the Edit Lead page showing the new subtab details.

Extending Simplified Pages for Assets: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for assets. The pages that are available for extensibility using Application Composer are listed below:
• Assets landing page
• 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 with the Customer Center application, to extend the simplified set of pages that are available for assets.
Note that you cannot add custom buttons and actions to any simplified page for assets.

Assets Landing Page
You can extend the following items in the list that appears on the Assets landing page:
• Hide or show existing fields that belong to the Asset object.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).

Create Asset Page
You can extend the following items on the Create Asset page:
• Hide or show existing fields that belong to the Asset object.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).

Edit Asset Page
You can extend the following items on the Edit Asset page:
• Hide or show existing fields that belong to the Asset object.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).

Extending Simplified Pages for Assets Using Application Composer: 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 Customer Center application on the main Overview page.

2. In the object tree, select the standard object, Asset, which includes a set of simplified pages.
   This figure shows the object tree with the Asset object selected.

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.
   This figure shows the configuration pages that are available to customer the set of simplified pages for assets.

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.

**Extending Simplified Pages for Contacts: Explained**

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for contacts.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Create Contact page
- Contact detail page (Profile page)
Note

Use the Contact object in Application Composer, available with the Common application, to extend the simplified set of pages that are available for contacts.

Note that these pages are not extensible:
- Customer Contacts overview table
- Contact Overview page

Create Contact Page

You can extend the following items on the Create Contact page:
- Hide or show existing fields that belong to the Contact object.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
  Custom fields that you add to the Contact object appear on the Create Contact page, before the addresses set of fields.

Note that you cannot add custom buttons and actions to this page.

Contact Detail Page (Profile Page)

You can extend the following items on the Contact detail page (Profile page):
- Hide or show existing fields that belong to the Contact object.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).
  Custom fields that you add to the Contact object appear on the Contact detail page (Profile page), before the Contact Information region.
- Extend the Interactions subtab.
In Application Composer, the Interaction object is available under the Common application.

Note that you cannot make changes to these components on the page:

- Subtab regions
- Notes overview page
- Tasks
- Appointments
- Actions button
- Custom buttons and actions

You cannot create new subtabs.

**Extending Simplified Pages for Customers: Explained**

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for customers.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.
- Customers overview table
- Create Customer page
- Edit Customer page (Profile page)
- Create Consumer page
- Edit Consumer page (Profile page)

**Note**

To extend the simplified set of pages that are available for customers, use the Contact and Account objects in Application Composer, available with the Common application. You will also use the Sales Account object, which is available with the Customer Center application.

Note that these pages are not extensible:
- Customer Overview page
- Consumer Overview page

**Customers Overview Table**

You can extend the following items in the Customers overview table:
- Hide or show existing fields that belong to the Sales Account object.
- Reorder columns.
- Change column labels.
- Add custom fields (all types), using the Sales Account object.

Note that you cannot make changes to these components on the page:
- Show filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
  - Custom font size for Name
• Drilldown
• Custom buttons and actions

**Create Customer Page**

You can extend the following items on the Create Customer page:

• Hide or show existing fields that belong to the Account and Sales Account objects.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).
  • Custom fields that you add to the Account object appear on the Create Customer page, before the addresses set of fields.
  • Custom fields that you add to the Sales Account object appears at the bottom of the page.

Note that you cannot add custom buttons and actions to this page.

**Edit Customer Page (Profile Page)**

These regions on the Edit Customer page (Profile page) are extensible:

• Basic Information region
  Use the Account object in Application Composer to extend this region.

• Sales Account region
  Use the Sales Account object in Application Composer to extend this region.

• Sales Account Team subtab
  Use the Sales Account Resource object in Application Composer to extend this region. Then, navigate to the Pages node for the Sales Account object, and edit the Sales Account Team subtab in the Subtabs region.

You can extend the following items on the Edit Customer page (Profile page):
• Hide or show existing fields.
• Reorder fields.
• Change display labels.
• Add custom fields (all types).

Create Consumer Page
You can extend the following items on the Create Consumer page:
• Hide or show existing fields that belong to the Contact and Sales Account objects.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).
  • Custom fields that you add to the Contact object appear on the Create Consumer page, before the addresses set of fields.
  • Custom fields that you add to the Sales Account object appears at the bottom of the page.
Note that you cannot add custom buttons and actions to this page.

**Edit Consumer Page (Profile Page)**

These regions on the Edit Consumer page (Profile page) are extensible:

- Basic Information region
  Use the Contact object in Application Composer to extend this region.
- Sales Account region
  Use the Sales Account object in Application Composer to extend this region.
- Sales Account Team subtab
  Use the Sales Account Resource object in Application Composer to extend this region. Then, navigate to the Pages node for the Sales Account object, and edit the Sales Account Team subtab in the Subtabs region.

You can extend the following items on the Edit Consumer page (Profile page):

- Hide or show existing fields.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

Note that you cannot make changes to these components on the page:

- Subtab regions
  - Opportunities
  - Leads
  - Actions button
  - Custom buttons and actions

You cannot create new subtabs.
Extending Simplified Pages for Leads: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for leads.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Leads landing page
- Create Lead page
- Edit Lead page (Summary page)

**Note**
Use the Sales Lead Version 2 object in Application Composer, available with the Marketing application, to extend the simplified set of pages that are available for leads.

**Leads Landing Page**
You can extend the following items in the list that appears on the Leads landing page:

- Hide or show existing fields.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

Note that you cannot make changes to these components on the page:

- Show filter
- Leads: Date filter
- Search
- Actions menu (cannot add or hide and show)
• Custom column widgets:
  • Rank
  • Custom font size for Name
• Drilldown
• Embedded analytics
• Custom buttons and actions

Create Lead Page
You can extend the following items on the Create Lead page:
• Hide or show existing fields.
• Reorder fields.
• Change field labels.
• Add custom fields (all types).

Note that you cannot add custom buttons and actions to this page.

Edit Lead Page (Summary Page)
These regions on the Edit Lead page (Summary page) are extensible:
• Edit Lead region
• Basic Qualification region
• Contacts subtab
  In Application Composer, add custom fields to the Sales Lead Contacts object. Then, navigate to the Pages node for the Sales Lead Version 2 object, and edit the Contacts subtab in the Leads tabs component region.
• Sales Team subtab
  In Application Composer, add custom fields to the Sales Lead Resources object. Then, navigate to the Pages node for the Sales Lead Version 2 object, and edit the Sales Team subtab in the Leads tabs component region.
• Notes subtab
  In Application Composer, the Note object is available under the Common application.
• Interactions subtab
  In Application Composer, the Interaction object is available under the Common application.
You can extend the following items on the Edit Lead page (Summary page):

- Hide or show existing fields.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

Note that you cannot make changes to these components on the page:

- Products table region
- Subtab regions
  - Full Qualification
  - Notes overview page
- Tasks
- Appointments
- Actions button
- Custom buttons and actions

You cannot create new subtabs.

**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.

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 **Marketing** from the **Application** list. Select **Lead** from the **Objects** panel. This figure shows the Sales Lead Overview page in Application Composer.

2. Under the Lead object, select the **Fields** link.

3. Click the **New** button and, in the Select Field Type window, select **Checkbox**. This figure shows the Select Field Type window with the **Checkbox** option selected.

4. In the Create Checkbox Field page, enter "Strategic Deal" as the **Display Name**. Leave the rest of the fields with their default values.
This figure shows the Create Checkbox Field page.

5. Click **Save and Close**.

6. Now augment this 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.

   This figure shows the Server Scripts object selected for a lead.
8. In the Server Scripts Lead page, select the Validation Rules tab and click the New button to create a new object rule.

This figure shows the Validation Rules tab in the Server Scripts Lead page.

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.

This figure shows the Create Object Validation Rule page.
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.

   This figure shows the Simplified Pages tab.

   ![Simplified Pages Tab](image)

12. In the Details Page Layout region, click the Edit Layout icon or click Actions - Edit Layout

13. In the Edit Lead page, select the Strategic Deal field in the Available Fields list and double click on this field to move it to the Selected Fields list.

   This figure shows the Edit Lead page.

   ![Edit Lead Page](image)

14. 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 the Edit Lead Summary page with the Strategic Deal check box selected.

   ![Edit Lead Summary Page](image)
17. Enter a value less than 1000 in the **Deal Size** field.

18. Select the **Strategic Deal** check box and click **Save and Close**.

19. You should receive the error message that you entered in Step 9 because the deal size violates the validation rule.

20. Reenter a value greater than 1000 in the **Deal Size** field and save.

You have successfully extended a simplified page and added Groovy logic.

**Extending Simplified Pages for Opportunities: Explained**

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for opportunities.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Opportunity landing page
- Create Opportunity page
- Edit Opportunity page

**Opportunity Landing Page**

You can extend the following items in the list that appears on the Opportunity overview table:

- Hide or show existing fields.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

This figure shows the Opportunities landing page.
Note that you cannot make changes to these components on the page:

- Show filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
  - Win Probability
  - Custom font size for Name
- Drilldown
- Embedded analytics
- Custom buttons and actions

**Create Opportunity Page**

You can extend the following items on the Create Opportunity page:

- Hide or show existing fields.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
- Add custom fields to the Revenue Items table.

In Application Composer, add custom fields to the Opportunity Revenue object, and then use the Edit Revenue Table configuration page available for the Opportunity object, under the Pages node.

Note that you cannot add custom buttons and actions to this page.

**Edit Opportunity Page**

These regions on the Edit Opportunity page are extensible:

- Opportunity region
- Revenue Items table

  In Application Composer, add custom fields to the Opportunity Revenue object, and then use the Edit Revenue Table configuration page available for the Opportunity object, under the Pages node.

- Opportunity Contacts subtab

  In Application Composer, add custom fields to the Opportunity Contact object. Then, navigate to the Pages node for the Opportunity object, and edit the Contacts subtab in the Opportunity Detail Tab region.

- Opportunity Team subtab
In Application Composer, add custom fields to the Opportunity Team object. Then, navigate to the Pages node for the Opportunity object, and edit the Team subtab in the Opportunity Detail Tab region.

- **Notes subtab**

  In Application Composer, the Note object is available under the Common application.

- **Interactions subtab**

  In Application Composer, the Interaction object is available under the Common application.

This figure shows the Edit Opportunity page.

You can extend the following items on the Edit Opportunity page:

- Hide or show existing fields.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

Note that you cannot make changes to these components on the page:

- Subtab regions
  - Notes overview page
  - Tasks
  - Appointments
- Appointments/Task region on Profile page
- Actions button
- Custom buttons and actions
- You cannot create new subtabs.
Extending Simplified Pages for Opportunities Using Application Composer: 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.

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.

   This figure shows the simplified page for Opportunities

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 **Fuse Opportunity Create** region.

8. In the Fuse Opportunity Create page, select **Budget Amount** from the Available Fields list in the Configure Detail Form region. This figure shows the Configure Detail Form in the Fuse Opportunity Create page.

9. Click the First arrow button to move the Budget Amount 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.

10. Click Save and Close.

**Extending Simplified Pages for Opportunities using Page Composer: Worked Example**

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 it for adding or removing predefined content from the Resource Library. All changes are done and stored in the UI layer. You can extend simplified pages using Page Composer. Perform the changes in sandbox to test it first.

This example demonstrates how you can hide the **Include in Forecast** field for a sales representative in the Opportunities page and leave it available for the rest of the roles.

**Extending Simplified Pages for Opportunities Using Page Composer**

The steps to hide the **Include in Forecast** field for a sales representative are:

1. Within a sandbox, sign in as a Sales Administrator in the Oracle Sales Cloud Sales Application to perform your configuration.
2. Click the role choice list and select Customize Pages. In this example, click Matt Hooper and select Customize Pages. This figure shows the Simplified Page for the Opportunity page with the **Customize User Interface** option selected.
3. In the Customize Application Pages, select the layer that you want to edit.

   You can customize three layers:
   
   • Global
   • External/Internal
   • Job Role

4. Click **Job Role** and select Sales Representative.

   This figure shows the Customize User Interface window.

5. After you invoke the Page Composer in a particular layer (in this case, the Job Role layer) two buttons are displayed.
The two buttons are:

- Design: Represents design mode used to extend transactional pages.
- Select: Allows you to select components in a page to edit.

This figure shows the Page Composer view in the Job Role layer.

6. Drill into an opportunity and click Select.

7. In the Edit Opportunity page that appears, select the Include in Forecast field that you want to hide.

This figure shows the Edit Opportunity page.

8. After you select the field, two options appear: Edit Component and Edit Parent Component.


10. In the Component Properties window, notice that the Show Component check box is selected, which means that this field will be visible in the Opportunity page.

This figure shows the Component Properties window.
11. Deselect **Show Component** to hide this field in the Opportunity page. Click **Apply** and click **OK**.

12. Notice that the field no longer appears in the Edit Opportunities page. Click **Close** to close Page Composer.

13. Note that in the Edit Opportunity Page, the field is still present because you are in a sales administrator role.

14. Sign out of the application and sign in as a sales representative.

15. On the Opportunity landing page, drill into an opportunity. In the Edit Opportunity page, notice that the **Include in Forecast** field is hidden or not visible.

   This figure shows the Edit Opportunity page for the sales representative role.

---

![Edit Opportunity page](image)

---

Application Composer: Extending Simplified Pages  6-41
Extending Simplified Pages for Sales Forecasts: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for sales forecasts.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Sales Forecasts landing page
- Forecast Items subtab

Note

The Analytics subtab and Unforecasted Pipeline subtab are also available from the Sales Forecasts functional area, but are not extensible.

Sales Forecasts Landing Page

On the Sales Forecasts landing page, you can extend the region that appears at the bottom of the page. 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, with the extensible region highlighted at the bottom of the page.

Note that you cannot make changes to these components on the page:

- Show filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
• Rank
• Custom font size for Name
• Drilldown
• Embedded analytics
• Custom buttons and actions

**Forecast Items Subtab**

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:

• Show filter
• Search
• Actions menu (cannot add or hide and show)
• Custom column widgets:
  • Rank
  • Custom font size for Name
• Drilldown
• Embedded analytics
• Custom buttons and actions

**Using Application Composer to Customize the Sales Forecasts Simplified Pages**

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 for the selected object.

<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 Sales Forecasts Summary</td>
</tr>
<tr>
<td>Forecast Items subtab</td>
<td>Edit the default layout in the Details Page Layouts section, or create a new page layout.</td>
</tr>
</tbody>
</table>

---

**Adding Drill Down Fields to a Subtab: Example**

This example covers the steps to add drill down fields to a subtab. In this example, you will first add a child or related object subtab in which the **Opportunity Name** column will contain the drill down fields. You will then verify the addition of the subtab.

**Creating a Subtab**

In this step, you will create a subtab of type child or related object using the Sales Lead object. This subtab will display the fields, including drill down fields, from the Opportunity object.

For more information on creating subtabs, see related link Subtabs: Explained.

1. Sign in to Oracle Fusion Applications.
2. Navigate to Application Composer.
3. Select the **Marketing** application.
4. Under the Standard Objects tree, expand the **Sales Lead** object.
5. Click the **Pages** node.
6. On the Sales Lead: Pages page, select the Simplified Pages tab.
7. Under the Details Page Layouts region, select **Default Layout**.

8. Click **Actions - Edit**.

9. In the Edit Layout page, select the plus icon in the subtabs region.

10. Select the **Child or related object** subtab.
11. Click Next.

12. On the Create Subtab page, specify the following:

   a. **Data Object**: The object from which the fields will be populated. Select *Opportunity*.
   
   b. **Display Label**: Name of the subtab. Enter *Opportunity Name*.
   
   c. **Drill Down Column**: The column which enables you to drill down or edit the object. Select *Opportunity Name*.
   
   d. **Display Icon**: This appears on the simplified page as a subtab icon.
   
   e. Specify the fields you want to appear on the subtab. This may include the drill down fields that appear on the page.

13. Save the Opportunity Name subtab.

   Using the Sales Lead object, you have created a subtab to display opportunities on a simplified page. You will now sign in to the Simplified Pages UI to view the new subtab.

**Verifying the Addition of the Subtab**

In this step, you will verify whether the Opportunity Name subtab has been added to the (Sales) Leads page.

1. Sign in to Simplified UI.
2. Select **Leads** to view a list of Leads.

3. Click on a lead to open the **Edit Lead** page.
On the Edit Lead page, the Opportunity Name subtab will appear as an icon on the left.
Select the subtab to view details.

Enabling or Disabling Drill Down Fields in Simplified Pages: 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 only for those standard objects whose task flow has been registered for extensibility. Common components and custom objects do not support drill down.
You can configure drill down for the following pages:
- 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 laptops or tablets. This figure shows the simplified pages tab.

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 example, the **Sales Account** field that is available on the Opportunity page is grayed out.

For workarea 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** drop down, 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.

  ![Image of Child or related object subtab configuration](image)

  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** drop down, 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.

  ![Image of Context link subtab configuration](image)

  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.

Making Custom Dynamic Choice List Fields Drillable: Worked Example

This worked example demonstrates how to making custom dynamic choice list fields drillable in simplified pages.

In this example, we will add make a dynamic choice list drillable.

Making Custom Dynamic Choice List Fields Drillable

1. Select a sandbox to work in and make it active.
2. Select the Marketing application in Application Composer.
3. Expand standard objects and click Sales Lead.
4. Click the Actions and Links link.
5. Click the Simplified Pages tab.
6. Select the default layout in the Details Page Layouts region and click the Edit Pencil icon or select Actions - Edit Layout.

7. Select the Summary tab and click the Edit Pencil icon.

This figure shows the Subtabs region with the Summary tab selected.
8. Select a dynamic choice list item from the Available Fields column. In this example, select LeadNilDynamicCL.

This figure shows the drill down status column for the selected dynamic choice list fields.

---

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?
• How to control the display of page layouts dynamically, based on:
  • Role/privileges of the user
  • Groovy expression
  • Type of record
• How to work with page layouts:
  • Editing
  • Duplicating
  • Inactivating

Where Can You Use Dynamic Page Layouts?

The page layouts you create are restricted to certain objects’ simplified pages, specifically the creation pages and details pages (edit pages).

You can create layouts that are displayed dynamically, for the objects listed below:
  • Account
  • Contact
  • Household
  • Opportunity
  • Sales Lead
  • Territory Forecast
  • Forecast Item
  • Notes

Note

Page layouts are not available for desktop pages, nor are they available for custom objects, although you can add custom object data as subtabs to other objects’ layouts.

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
  • Role
  • Advanced expression

Tip
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.

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 default layout is always the last layout in the table, and it cannot be removed.

### Working With Page Layouts

The objects listed above are delivered with default layouts. You can edit the existing default layouts, or you can duplicate and then edit a page layout to create a new layout. When a layout is no longer of use, you can inactivate, or deprecate, that layout.

You can:

- **Edit page layouts**
  Edit existing page layouts, including the default simplified page layout for creation pages and details pages that all supported objects are delivered with.

**Note**

When you edit the default layout for either the creation page or details page, note that the Type, Role, and Advanced Expression conditions are disabled. You cannot add conditions to the default layout.

The changes that you make to default layouts are retained, even after an upgrade.

- **Duplicate page layouts**
  To create a new page layout, duplicate an existing layout and then make your edits.

**Note**

When you duplicate the default layout, you make a new layout which exists independently from the default layout. During upgrades, default layouts are automatically refreshed, whereas duplicated layouts are not.

- **Inactivate, or deprecate, page layouts**
  You cannot delete page layouts, but you can inactivate them by deselecting the Active check box for a page layout on the Simplified Pages tab. You cannot inactivate the default layouts.

To create or edit 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. In the Creation Page Layouts region or Details Page Layouts region, edit the default layout, or click the **Duplicate Layout** icon to duplicate and edit an existing 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 default layout is always the last layout in the table, and it cannot be removed.
5. When you edit a page 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.

6. After you edit the page layout, you can assign one or more conditions that control when the layout is displayed:
   - Type
     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 page 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.
   - Role
     Select the enterprise duty role who can see this page layout. For example, perhaps only the sales representative can see this page layout at run time.
     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
     Enter a Groovy expression that controls when this page layout is displayed.

**Page Layouts per Record Type: 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. For example, if an opportunity is open, then certain fields might display. However, if the opportunity is closed, then other fields might display. To control the display of layouts based on the type of record, you must first create a record type field for an object.

Read this topic to learn about creating page layouts for a set of record types:
   - Creating a record type field for an object
   - Creating page layouts per record type
   - Examples of page layouts per record type

**Creating a Record Type Field**

To control the display of page layouts based on the type of record, you must first create a record type field for an object. A record type field is actually a choice list field with a list of values that you specify. After you create the record type
field, you 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). Also, you assign each record type value to a layout. At run time, the system assesses the value of the record type field to determine which layout to display to your users.

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 updateable.
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 enterprise duty roles have which access to particular record type 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 record type values, but the sales manager can see all the record type values.

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 record type values to one or more page
layouts. In this way, you control which layout displays at run time, depending on the type of record.

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. In the Creation Page Layouts region or Details Page Layouts region, click the Duplicate Layout icon to duplicate and edit a layout to create a new layout.

Note
Remember that when you edit the default layout for either the creation page or details page, the Type, Role, and Advanced Expression conditions are disabled. You cannot add conditions to the default layout.

5. Add the record type field to the selected layout.
6. After editing and saving the layout, select a record type field value under the Type column. At run time, if the value is selected, 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 set of record types 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.

Page Layouts per 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 enterprise duty role. For example, a sales representative might see one particular layout, while the sales manager might see a different layout.

Read this topic to learn about creating page layouts for enterprise duty roles:
- Creating page layouts per role
- Examples of page layouts per role

Creating Page Layouts per Role

Assign an enterprise duty 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. In the Creation Page Layouts region or Details Page Layouts region, click the Duplicate Layout icon to duplicate and edit a layout to create a new layout.

Note

Remember that when you edit the default layout for either the creation page or details page, the Type, Role, and Advanced Expression conditions are disabled. You cannot add conditions to the default layout.

5. After editing and saving the layout, select an enterprise duty role under the Role column. At run time, if the user has the specified role, then this 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.

Page Layouts per 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. Read this topic to learn about writing expressions for displaying page layouts:

- Writing expressions for displaying page layouts
- Examples of page layouts that display based on expressions

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.
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. In the Creation Page Layouts region or Details Page Layouts region, click the **Duplicate Layout** icon to duplicate and edit a layout to create a new layout.

**Note**

Remember that when you edit the default layout for either the creation page or details page, the Type, Role, and Advanced Expression conditions are disabled. You cannot add conditions to the default layout.

5. After editing and saving the layout, under the Advanced Expression column, access the expression builder by clicking the calculator icon. Use the expression builder to write an expression that describes the conditions required for this layout to display at run time.

- If your script references a field, then you must select those fields using the Depends On choice list. This selection enables your script to work at run time.

For example, let’s say you write this script:
Your script references two fields. Thus, you must select those fields, Note Text and Type, in the Depends On choice list, which appears at the top of the expression builder.

- 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.

### How can I post announcements on the simplified home page?

On any simplified page, click Settings in the springboard, and then select the Announcements tab. Use the Announcements page to create, edit, and delete announcements.

Only the content (not the subject) of announcements appears on the home page.
Using Groovy Scripts: Overview

Read this chapter to learn about how and where you can use Groovy scripting in Application Composer.

**Important**

To fully understand all the scripting features available to you in Application Composer, you should also review the Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

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.

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 will generally 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 Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

**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 scripts, see Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

---

**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, we strongly suggest that the Groovy code you write uses only the classes and methods shown there to avoid the unpleasant and possibly time-consuming task of having to rewrite your code in the future.

---

**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 CommissionPercentage field’s value does not exceed 40%, you might use a one-line script like:

```
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:

```
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:

```
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, however 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.

**Where You will Use Groovy 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 updateable, 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 Fusion CRM Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com, see "Groovy Scripting: Examples" for a concrete example of each of these usages. The Oracle Fusion CRM Application Composer Scripting Guide also includes a section on "Groovy Tips and Techniques" for additional tips and techniques on getting the most out of Groovy in your application.
**Groovy Scripting: Examples**

The following examples show how you can use Groovy in all of the different supported contexts in your application.

**Providing an Expression to Calculate a Custom Formula Field's Value**

When you need a calculated field or a transient value-holder field with an optional initial, calculated value, use a formula field.

1. For read-only calculated fields:

A formula field defaults to being a read-only, calculated value. It displays the value resulting from the run time evaluation of the calculation expression you supply. By using the **Depends On** multi-select list in the field create or edit page, you can configure the names of fields on which your expression depends. By doing this, its calculated value will update dynamically when any of those **Depends On** fields' value changes. The expected return type of the formula field's expression must be compatible with the formula field type you specified (Number, Date, or Text).

For example, consider a custom **TroubleTicket** object. If you add a formula field named **DaysOpen**, you can provide its calculated value with the expression:

```
(today() - CreationDate) as Integer /* truncate to whole number of days */
```

2. For transient value holder fields with optional calculated initial value:

If you want to allow the end user to override the calculated value, then mark your formula to be updateable. An updateable formula field is a "transient value holder" whose expression provides the value of the field until the user overrides it. If the user overrides the value, the object remembers this user-entered value for the duration of the current transaction so that your validation rules and triggers can reference it. If you have configured one or more **Depends On** fields for your updateable formula field, then note that the value of the formula will revert back to the calculated value should any of the dependent fields’ value change. If you want a transient field whose initial value is null until the user fills it in, simply provide no formula expression for your updateable formula field to achieve this.

**Providing an Expression to Calculate a Custom Field's Default Value**

When a new row is created for an object, the value of a custom field defaults to **null** unless you configure a default value for it. You can supply a literal default value of appropriate type or supply an expression to calculate the default value for new rows. The default value expression is evaluated at the time the new row is created. The expected return type of your field’s default value expression must be compatible with the field’s type (Number, Date, Text, and so on).

For example, consider a custom **CallbackDate** field in a **TroubleTicket** object. If you want the callback back for a new Help Request to default to 3 days after it was created, then you can provide a default expression of:

```
CreationDate + 3
```
Providing an Expression to Make a Custom Field Conditionally Updateable

1. To provide an expression to make a custom field conditionally updateable:

A custom field can be updateable or read-only. By default, any non-formula field is updateable. Alternatively, you can configure a conditionally updateable expression. If you do this, it is evaluated each time a page displaying the field is rendered or refreshed. The expected return type of the expression is boolean. If you define one for a field, you must also configure the Depends On list to indicate the names of any fields on which your conditionally updateable expression depends. By doing this, your conditionally updateable field will interactively enable or disable as appropriate when the user changes the values of fields on which the conditionally updateable expression depends.

For example, consider a custom TroubleTicket object with Status and Justification fields. Assume you want to prevent a user from editing the justification of a closed Help Request. To achieve this, configure the conditionally updateable expression for the Justification field as follows:

```
Status_c != 'Closed'
```

After configuring this expression, you must then indicate that the Justification field depends on the Status field as described in "Understanding When to Configure Field Dependencies", in Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at https://support.oracle.com. This ensures that if a Help Request is closed during the current transaction, or if a closed Help Request is reopened, that the Justification field becomes enabled or disabled as appropriate.

Tip

A field configured with a conditionally updateable expression only enforces the conditional updateability through the Web user interface. For more information on how to ensure it gets enforced for Web service access as well, see "Enforcing Conditional Updateability of Custom Fields for Web Service Access" in Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at http://www.oracle.com/technetwork/indexes/documentation.

2. To provide an expression to make a custom field conditionally required:

A custom field can be optional or required. By default it is optional. Alternatively, you can configure a conditionally required expression. If you do this, it is evaluated each time a page displaying the field is rendered or refreshed, as well as when the object is validated. The expected return type of the expression is boolean. If you define one for a field, you must also configure the Depends On list to indicate the names of any fields on which your conditionally required expression depends. By doing this, your conditionally required field will interactively show or hide the visual indicator of the field’s being required as appropriate when the user changes the values of fields on which the conditionally required expression depends.

For example, consider a custom TroubleTicket object with Priority and Justification fields. Assume that priority is an integer from 1 to 5 with priority 1 being the most critical kind of problem to resolve. To enforce that a justification is required for Help Requests whose priority is 1 or 2, configure the conditionally required expression for the Justification field as follows:

```
Priority_c <= 2
```
After configuring this expression, you must then indicate that the **Justification** field depends on the **Priority** field as described in "Understanding When to Configure Field Dependencies", in Groovy Scripting Reference for Application Composer (ID 1354807.1). This ensures that if a Help Request is created with priority 2, or an existing Help Request is updated to increase the priority from 3 to 2, the **Justification** field becomes mandatory.

### Defining a Field-Level Validation Rule

1. To define a field-level validation rule:

A field-level validation rule is a constraint you can define on any standard or custom field. It is evaluated 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. 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. Your script can use the **newValue** keyword to reference the new value that will be assigned if validation passes. To reference the existing field value, use the **oldValue** keyword. A field-level rule is appropriate when the rule to enforce only depends on the new value being set. You can use the Keywords tab of the Expression Palette to insert the **newValue** and **oldValue** keywords.

For example, consider a custom **TroubleTicket** object with a **Priority** field. To validate that the number entered is between 1 and 5, your field-level validation rule would look like this:

<table>
<thead>
<tr>
<th>Field-Level Validation Rule Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Name</td>
<td>Priority</td>
</tr>
<tr>
<td>Rule Name</td>
<td>Validate_Priority_Range</td>
</tr>
<tr>
<td>Rule Body</td>
<td></td>
</tr>
<tr>
<td>Error Message</td>
<td>The priority must be in the range from 1 to 5.</td>
</tr>
</tbody>
</table>

**Tip**

If a validation rule for field A depends on the values of one or more other fields (For example, Y and Z), then create an object-level rule and programmatically signal which field or fields should be highlighted as invalid to the user, as explained in "Setting Invalid Fields for the UI in an Object-Level Validation Rule" in Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

2. To define an object-level validation rule:
An object-level validation rule is a constraint you can define on any standard or custom object. It is evaluated whenever the framework attempts to validate the object. This can occur upon submitting changes in a web form, when navigating from one row to another, as well as when changes to an object are saved. 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 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.

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. To validate that a Help Request of priority 1 or 2 cannot be saved without being assigned to a staff member, your object-level rule would look like this:

<table>
<thead>
<tr>
<th>Object-Level Validation Rule Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Name</td>
<td>Validate_High_Priority_Ticket_Has_Owner</td>
</tr>
<tr>
<td>Rule Body</td>
<td></td>
</tr>
<tr>
<td>Error Message</td>
<td>A Help Request of priority 1 or 2 must have a staff member assigned to it.</td>
</tr>
</tbody>
</table>

### Defining Utility Code in a Global Function

Global functions are useful for code that multiple objects want to share. 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. The most common types for function return values and parameters are the following:

- **String**: a text value
- **Boolean**: a logical true or false value
- **Long**: an integer value
- **BigInteger**: an integer of arbitrary precision
- **Double**: a floating-point decimal value
- **BigDecimal**: a decimal number of arbitrary precision
- **Date**: a date value with optional time component
- **List**: an ordered collection of objects
- **Map**: an unordered collection of name/value pairs
- **Object**: any object

In addition, a function can define a void return type which indicates that it returns no value.
Note

A global function has no current object context. To write global functions that work on a particular object, refer to “Passing the Current Object to a Global Function,” in Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at http://www.oracle.com/technetwork/indexes/documentation.

For example, you can create the following two global functions to define standard helper routines to log the start of a block of Groovy script and to log a diagnostic message. The examples in Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at http://www.oracle.com/technetwork/indexes/documentation, make use of them.

This table describes how to set up a global function to log the start of a block of Groovy script.

<table>
<thead>
<tr>
<th>Global Function Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Name</td>
<td>logStart</td>
</tr>
<tr>
<td>Return Type</td>
<td>void</td>
</tr>
</tbody>
</table>
| Parameters                | Name: scriptName  
Type: String |
| Function Definition       | |

This table describes how to set up a global function to log a diagnostic message.

<table>
<thead>
<tr>
<th>Global Function Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Name</td>
<td>log</td>
</tr>
<tr>
<td>Return Type</td>
<td>void</td>
</tr>
</tbody>
</table>
| Parameters                | Name: message  
Type: String |
| Function Definition       | |

Defining Reusable Behavior with an Object Function

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 code 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 (described above).
For example, you might define the following `updateOpenTroubleTicketCount()` object function on a `Contact` custom object. It begins by calling the `logStart()` global function above to log a diagnostic message in a standard format to signal the beginning of a block of custom Groovy script. It calls the `newView()` built-in function (described in "Accessing the View Object for Programmatic Access to Business Objects" in Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support) to access the view object for programmatic access of Help Requests, then calls another global function `applyFilter()` (described in “Simplifying Most Common View Criteria Creation with a Global Function” in Groovy Scripting Reference for Application Composer (ID 1354807.1)) to apply a filter to find Help Requests related to the current contact’s id and having either `Working` or `Waiting` as their current status. Finally, it calls `getEstimatedRowCount()` to retrieve the count of Help Requests that qualify for the filter criteria.

This table describes the `updateOpenTroubleTicketCount()` object function on a `Contact` custom object:

<table>
<thead>
<tr>
<th>Object Function Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Name</td>
<td><code>updateOpenTroubleTicketCount</code></td>
</tr>
<tr>
<td>Return Type</td>
<td><code>void</code></td>
</tr>
<tr>
<td>Parameters</td>
<td><code>None</code></td>
</tr>
<tr>
<td>Function Definition</td>
<td></td>
</tr>
</tbody>
</table>

### Defining an Object Level Trigger to Complement Default Processing

Triggers are scripts that you can write to complement the default processing logic for a standard or custom object. You can define triggers both at the object-level and the field-level. The following object level triggers are available:

- **After Create**: Fires when a new instance of an object is created. Use to assign programmatic default values to one or more fields in the object.
- **Before Modify**: Fires when the end-user first modifies a persistent field in an existing, queried row.
- **Before Invalidate**: Fires on an existing object when its first persistent field is modified. Fires on a valid parent object when a child row is created, removed, or modified.
- **Before Remove**: Fires when an attempt is made to delete an object. Returning false stops the row from being deleted and displays the optional trigger error message.
- **Before Insert in Database**: Fires before a new object is inserted into the database.
  
  See tip below.
- **After Insert in Database**: Fires after a new object is inserted into the database.
See tip below.

- **Before Update in Database**: Fires before an existing object is modified in the database.
  See tip below.

- **After Update in Database**: Fires after an existing object is modified in the database.
  See tip below.

- **Before Delete in Database**: Fires before an existing object is deleted from the database.

- **After Delete in Database**: Fires after an existing object is deleted from the database.

- **After Changes Posted to Database**: Fires after all changes have been posted to the database, but before they are permanently committed. Can be used to make additional changes that will be saved as part of the current transaction.
  See tip below.

- **Before Commit in Database**: Fires before the change pending for the current object (insert, update, delete) is made permanent in the current transaction. Any changes made in this trigger will not be part of the current transaction. Use "After Changed Posted to Database" trigger if your trigger needs to make changes.

- **After Commit in Database**: Fires after the change pending for the current object (insert, update, delete) is made permanent in the current transaction.

- **Before Rollback in Database**: Fires before the change pending for the current object (insert, update, delete) is rolled back.

- **After Rollback in Database**: Fires after the change pending for the current object (insert, update, delete) is rolled back.

Consider a Contact object with a OpenTroubleTickets field that needs to be updated any time a Help Request is created or modified. You can create the following two triggers on the TroubleTicket object which both invoke the updateOpenTroubleTicketCount() object function described above.

This table describes how to set up a trigger on the TroubleTicket object.

<table>
<thead>
<tr>
<th>Trigger Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Object</td>
<td>TroubleTicket</td>
</tr>
<tr>
<td>Trigger</td>
<td>After Insert In Database</td>
</tr>
<tr>
<td>Trigger Name</td>
<td>After_Insert_Set_Open_Trouble_Tickets</td>
</tr>
<tr>
<td>Trigger Definition</td>
<td></td>
</tr>
</tbody>
</table>
This table describes how to set up a second trigger on the **TroubleTicket** object.

<table>
<thead>
<tr>
<th>Trigger Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Object</td>
<td>TroubleTicket</td>
</tr>
<tr>
<td>Trigger</td>
<td>After Update In Database</td>
</tr>
<tr>
<td>Trigger Name</td>
<td>After_Update_Set_Open_Trouble_Tickets</td>
</tr>
<tr>
<td>Trigger Definition</td>
<td></td>
</tr>
</tbody>
</table>

**Tip**

When setting the value of an attribute using the triggers listed below, check to see if the attribute being set already has the value being assigned, and if so, skip performing the assignment.

- Before Insert in Database
- After Insert in Database
- Before Update in Database
- After Update in Database
- After Changes Posted to Database

For example:

```groovy
if (Text_c != 'AfterCommit') {
    setAttribute('Text_c', "AfterCommit")
}
```

**Defining a Field-Level Trigger to React to Value Changes**

Field-level triggers are scripts that you can write to complement the default processing logic for a standard or custom field.

The After Field Changed trigger fires when the value of the related field has changed (implying that it has passed any field-level validation rules that might be present). Use the After Field Changed trigger to calculate other derived field values when another field changes value. 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.

**Using Groovy Scripting: Example**

The 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 Oracle Fusion CRM Application Composer provides deep support. This topic provides simple examples of using Groovy in various contexts.

This example demonstrates:

- Using the println function in your Groovy scripts
- Making a field conditionally required
- Making a field conditionally updateable
- Adding a default value to a field
- Adding validation to a field

**Using The Println Function In Your Groovy Scripts**

Write your first Groovy script to understand how the println function can be used to debug your scripts.

In this example, you will add a println function to a trigger to view the opportunity’s Win Probability. Whenever the opportunity’s Win Probability field is updated, the println function inside the trigger will fire on update, and its output will be displayed for you in the Run Time Messages UI, within the Application Composer.

1. Sign in to Oracle Fusion CRM.
2. Click the Navigator link.
3. Click the more... >> link.
4. Click the Application Composer link.
5. Click the Application list.
6. Click the Sales list item.
7. Click the Expand link next to the Standard Objects item.
8. Click the Expand link next to the Opportunity item.
9. Click the Server Scripts link.
10. Click the Triggers tab.
11. Click the Add a new Trigger... button.
12. Click the Trigger list.
13. Click the Before Update in Database list item.
14. Enter the desired following information into the Trigger Name field. Enter: TestPrintln_XY where XY is your user name initials.
15. Click in the Expression field.
16. Enter the following information into the Expression field. Enter:
   ```groovy
   if(startsWith(Name, 'XY')) { println("XY - Before Update Trigger. The new value of the Win Probability is" + WinProb) }
   ```
17. Click the Validate button.
18. Question: How can you correct the warning that appears? Why is it popping up? Answer: WinProb could be null. Next, we will ensure that
our variables are null-aware by using the nvl() function to define a value in the case of a null. Click the OK button.

19. Change the println script as follows (changes are shown in bold):
   
   ```groovy
   if(startsWith(Name, 'XY')) { println("XY - Before Update Trigger.
   The new value of the Win Probability is" + nvl(WinProb, "Win
   Probability was null")) }
   ```
   
   20. Click the Validate button.
   
   21. Notice the message that informs you that the script has parsed successfully.
   
   22. Click the Save and Close button.
   
   23. Click the Navigator link.
   
   24. Click the Opportunities link.
   
   25. Click the Create Opportunity link.
   
   26. Enter the XY Opportunity Trigger Test into the Name field, where XY is the initials of your user name.
   
   27. Enter the desired information into the Win Probability (%) field. Enter "50".
   
   28. Click the Save and Close button.
   
   29. Click the XY Opportunity Trigger Test link.
   
   30. Click the Edit button.
   
   31. Enter the desired information into the Win Probability (%) field. Enter "25".
   
   32. Click the Save and Close button.
   
   33. Click the Navigator link.
   
   34. Click the more... >> link.
   
   35. Click the Application Composer link.
   
   36. Click the Application list.
   
   37. Click the Sales list item.
   
   38. Click the Run Time Messages tree item in the Common Setup area.
   
   39. Click the Get Latest Log Messages button.
   
   40. Click the Sort Descending button.
   
   41. Ensure that the message is as you designated in your println function.
   
   42. You have successfully added a println function to a trigger to view an opportunity’s Win Probability.

**Making a Field Conditionally Required**

To learn how to make a field conditionally required, you will add two custom fields to your Help Request object: a Justification field and a Priority field. You can make the completion of the Justification field conditionally required, based on whether or not the Priority field was updated to Urgent.

- If the Priority field was updated to Urgent, then the Justification field is required.
• If the Priority field was updated to anything other than Urgent, then completing the Justification field is optional.

1. Sign in to Oracle Fusion CRM.
2. Click the Navigator link.
3. Click the more... >> link.
4. Click the Application Composer link.
5. Click the Application list.
6. Click the Sales list item.
7. Click the Expand button next to Custom Objects.
8. Click the Expand button next to the Help Request item.
9. Click the Fields tree item.
10. Create a text field called Justification.
11. Create a fixed choice list field called Priority, with values such as Important and Urgent.
12. Select the Justification field and click the Edit the selected field... button.
13. Click the Enter expression for the field’s Required property value button next to the Required field in the Constraints area.
14. Click in the Expression field.
15. Enter the desired information into the Expression field. Enter: 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") } 
16. Click the OK button.
17. Click the Depends On list.
18. Click the Priority list item.
19. Click the Save and Close button.
20. Click the Pages tree item.
21. Click the Edit Summary Form link.
22. Double-click the Priority field to move it into the Selected Fields list.
23. Double-click the Justification field to move it into the Selected Fields list.
24. Notice that the Priority field is above the Justification field. If this is not the case, you should click Priority and then click the up arrow button to the right of the Selected Fields area to move it above Justification.
25. Click the Save and Close button.
26. Click the Navigator link.
27. Click the Help Request link.
28. Click any active Help Request in the list.
29. Click the Priority list.
30. Click the Important list item.
31. Click the Save button.
32. Notice that the ticket can be saved successfully.
33. Click the Priority list.
34. Click the Urgent list item.
   Notice that an asterisk appears next to the Justification field, indicating that is now a required field.
35. Click the Save button.
36. An error message is displayed because you did not enter a value in the Justification field before saving.
37. Enter the desired information into the Justification field. Enter "Laptop is on fire".
38. Click the Save and Close button.
39. The ticket is saved successfully.
40. You have successfully added two custom fields to your Help Request object, and made the completion of one of the fields conditionally required based on the value to which the other field was updated.

**Making a Field Conditionally Updateable**

To learn how to make a field conditionally updateable, you will add two custom fields to your Help Request object: an Executive Sponsor field and an Executive Sponsor Program check box. You can make the completion of the Executive Sponsor field conditionally updateable, based on whether or not the Executive Sponsor Program check box is updated to Yes.

- If the Executive Sponsor Program check box is updated to Yes, then the Executive Sponsor field can be updated.
- If the Executive Sponsor Program check box is updated to No, then completing the Executive Sponsor field is not possible.
1. Sign in to Oracle Fusion CRM.
2. Click the Navigator link.
3. Click the more... >> link.
4. Click the Application Composer link.
5. Click the Application list.
6. Click the Sales list item.
7. Click the Expand link next to the Custom Objects item.
8. Click the vertical scrollbar to scroll down to the Help Request item.
9. Click the Expand link next to the Help Request item.
10. Click the Fields link.
11. Create a text field called Executive Sponsor and a check box field called Executive Sponsor Program.
12. Select the Executive Sponsor field and click the Edit the selected field... button.
13. Click the Enter expression for the field’s Updateable property value button next to the Updateable field.

14. Click in the Expression field.

15. Enter the following into the Expression field:

```java
try {
    if(nvl(ExecutiveSponsorProgram_c,"") == "N") {
        return false
    } else {
        return true
    }
} catch(e) {
    println("Error with the Updateable property of Executive Sponsor field in the Help Request object")
}
```

16. Click the OK button.

17. Click the Depends On list.

18. Click the Executive Sponsor Program option to select it.

19. Click the Save and Close button.

20. Click the Pages tree item under Help Request in the Objects list.

21. Click the Edit Summary Form link.

22. Double-click in the Executive Sponsor field to move it to the Selected Fields list.

23. Double-click in the Executive Sponsor Program field to move it to the Selected Fields list.

24. The Executive Sponsor Program item needs to be above the Executive Sponsor item, so you need to move it up in the list. Click in the Executive Sponsor Program field.

25. Click the Move selected items up one in list link to the right of the Selected Fields list.

26. Click the Save and Close button.

27. Next, you will test the change you made. Click the Navigator link.

28. Click the Help Request link.

29. Click any active Help Request.

30. Click the Executive Sponsor Program option to select it.

31. Click the Save button. Notice that you can enter a name into the Executive Sponsor field.

32. Click the Executive Sponsor Program option to deselect it.

33. Click the Save button. Notice that you can no longer enter a name into the Executive Sponsor field.

34. You have successfully made a field conditionally updateable.

Adding a Default Value to a Field

In this section, you will add a default value to a field using an expression. To learn how to add a default value to a field using an expression, you will add a custom field to your Activity child object: Current Ticket Priority. When you create a new activity for a Help Request, automatically populate the activity’s Current Ticket Priority field with the value from the parent Help Request’s Priority field.
1. Sign in to Oracle Fusion CRM.
2. Click the Navigator link.
3. Click the more... >> link.
4. Click the Application Composer link.
5. Click the Application list.
6. Click the Sales list item.
7. Click the Expand button next to the Custom Objects item.
8. Click the Expand button next to the Activity item.
9. Click the Fields tree item.
10. Create a fixed choice list field called Current Ticket Priority.
11. For that new field, click the Expression option in the Default Value area to select it.
12. Click the Enter the expression used for the field's default value button.
13. Click in the Expression field.
14. Enter the desired information into the Expression field. Enter "TroubleTicket_c?.Priority_c".
15. Tip: Instead of typing the expression, try selecting the object in the expression builder, then selecting the field, then clicking Insert. The selected field’s value will populate the Current Ticket Priority field on new activities that you create.
16. Click the Save and Close button.
17. Click the vertical scrollbar in the Objects area to scroll down to Help Request.
18. Click the Expand button next to the Help Request item.
19. Click the Pages tree item.
20. Click the Activities object.
21. Click the Edit button.
22. Double-click in the Current Ticket Priority field in the Configure Summary Table section to move it into the Selected Fields area.
23. Double-click in the Current Ticket Priority field in the Configure Detail Form section to move it into the Selected Fields area.
24. Click the Save and Close button.
25. Click the Navigator link.
26. Click the Help Request link.
27. Click any active Help Request.
28. Click the Add Row button.
29. Notice how the value for Current Ticket Priority is defaulted from the parent record.
30. You have successfully added a default value to a field using an expression.

Adding Validation to a Field

In this section, you will add validation to a field using an expression. To learn how to add validation to a field using an expression, first add a custom text field to your Help Request object: Corporate E-Mail. You will add validation to this Corporate E-Mail field to check if the e-mail address that your users enter at runtime is valid.

1. Sign in to Oracle Fusion CRM.
2. Click the Navigator link.
3. Click the more... >> link.
4. Click the Application Composer link.
5. Click the Application list.
6. Click the Sales list item.
7. Click the Expand button next to the Custom Objects item.
8. Click the vertical scrollbar if necessary to scroll down to your Help Request.
9. Click the Expand button next to the Help Request item. Click the + button before the Expand tree item.
10. Click the Pages tree item.
11. Click the Edit Creation Page link.
12. Double-click the Corporate E-Mail list item to move it to the Selected Fields list.
13. Click the Save and Close button.
14. Click the Edit Summary Form link.
15. Double-click in the Corporate E-Mail field under the Configure Default Summary section to move it into the Selected Fields list.
16. Click the Save and Close button.
17. Click the vertical scrollbar.
18. Click the Server Scripts tree item.
19. Click the Validation Rules tab.
20. Click the Add a new validation rule... button.
21. Click the Field Name list.
22. Click the Corporate E-Mail list item.
23. Click in the Rule Name field.
24. Enter the desired information into the Rule Name field. Enter "CheckValid".
25. Click in the Expression field.
26. Enter the desired information into the Expression field. Enter: ```java import java.util.regex.Matcher; import java.util.regex.Pattern; //Will
return true if correct and false if not. def x= new String(newValue)
Pattern pattern; Matcher matcher; def EMAIL_PATTERN = new
String(".*@.*\..*") pattern = Pattern.compile(EMAIL_PATTERN);
matcher = pattern.matcher(x); return matcher.matches();

27. Click in the Error Message field.
28. Enter the desired information into the Error Message field. Enter "The Corporate E-Mail is not valid syntax."
29. Click the Save and Close button.
30. Now you verify that your expression is functioning correctly. Click the Navigator link.
31. Click the Help Request link.
32. Click any active Help Request.
33. Click in the Corporate E-Mail field.
34. First, you will enter an incorrectly formed e-mail address into the field. Enter the desired information into the Corporate E-Mail field. Enter "mhoope.oracle.com".
35. Click the Save button.
36. Notice that the error message you specified is displayed, because the e-mail address you entered was not in proper syntax.
37. Click in the Corporate E-Mail field.
38. Enter the desired information into the Corporate E-Mail field. Enter "mhoope@oracle.com".
39. Click the Save button.
40. Notice that you have now been allowed to save the Help Request with the e-mail address added.
41. You have successfully added validation to a field using an expression.

Supported Classes and Methods for Use in Groovy Scripts:
Explained

Groovy is a standard, dynamic scripting language for the Java platform for which Application Composer provides deep 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()`

<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> <code>getSecurityContext()</code> <code>getUserRoles()</code> <code>isUserInRole()</code></td>
<td>oracle.adf.share</td>
</tr>
<tr>
<td>Array</td>
<td>Any constructor Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>Array</td>
<td><code>getArray()</code> <code>getElemType()</code> <code>getList()</code></td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td>ArrayList</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Arrays</td>
<td>Any constructor Any method</td>
<td>java.util</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>AttributeDef</td>
<td>getAttributeKind()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getIndex()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getJavaType()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getPrecision()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getScale()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUIHelper()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getUpdateableFlag()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isMandatory()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>isQueriable()</td>
<td></td>
</tr>
<tr>
<td>AttributeHints</td>
<td>getControlType()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getDisplayHeight()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayHint()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayWidth()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormat()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormattedAttribute()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormatter()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getFormatterClassName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getHint()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLocaleName()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>parseFormattedAttribute()</td>
<td></td>
</tr>
<tr>
<td>AttributeList</td>
<td>getAttribute()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getAttributeIndexOf()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAttributeNames()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>setAttribute()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>BaseLobDomain</td>
<td>closeCharacterStream()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>closeInputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>closeOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getInputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLength()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getcharacterStream()</td>
<td></td>
</tr>
<tr>
<td>BigDecimal</td>
<td>Any constructor</td>
<td>java.math</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BigInteger</td>
<td>Any constructor</td>
<td>java.math</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BitSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Blob</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>BlobDomain</td>
<td>Any constructor</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>getBinaryOutputStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBinaryStream()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBufferSize()</td>
<td></td>
</tr>
<tr>
<td>Boolean</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Byte</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Calendar</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Char</td>
<td>Any constructor</td>
<td>oracle. jbo.domain</td>
</tr>
<tr>
<td></td>
<td>bigDecimalValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>bigIntegerValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>booleanValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>doubleValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>floatValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>intValue()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>longValue()</td>
<td></td>
</tr>
<tr>
<td>Clob</td>
<td>Any constructor</td>
<td>java.sql</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>ClobDomain</td>
<td>Any constructor</td>
<td>oracle. jbo.domain</td>
</tr>
<tr>
<td></td>
<td>toCharArray()</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>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>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------</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>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>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Exception</td>
<td>Any constructor Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>ExprValueErrorHandler</td>
<td>addAttribute() clearAttributes() raise() raiseLater() warn()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Float</td>
<td>Any constructor Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>Formattable</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>FormattableFlags</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Formatter</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>GregorianCalendar</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>HashMap</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>HashSet</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Hashtable</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>IdentityHashMap</td>
<td>Any constructor Any method</td>
<td>java.util</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>Integer</td>
<td>Any constructor&lt;br&gt;Any method</td>
<td>java.lang</td>
</tr>
<tr>
<td>Iterator</td>
<td>Any constructor&lt;br&gt;Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>JboException</td>
<td>getDetails()&lt;br&gt;getErrorCode()&lt;br&gt;getErrorParameters()&lt;br&gt;getLocalizedMessage()&lt;br&gt;getMessage()&lt;br&gt;getProductCode()&lt;br&gt;getProperty()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>JboWarning</td>
<td>Any constructor&lt;br&gt;getDetails()&lt;br&gt;getErrorCode()&lt;br&gt;getErrorParameters()&lt;br&gt;getLocalizedMessage()&lt;br&gt;getMessage()&lt;br&gt;getProductCode()&lt;br&gt;getProperty()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Key</td>
<td>toStringFormat()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>LinkedHashMap</td>
<td>Any constructor&lt;br&gt;Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>LinkedHashSet</td>
<td>Any constructor&lt;br&gt;Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>LinkedList</td>
<td>Any constructor&lt;br&gt;Any method</td>
<td>java.util</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>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>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>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Number</td>
<td>Any constructor</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td></td>
<td>decimalValue()</td>
<td></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>Random</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>RandomAccess</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Ref</td>
<td>Any constructor, Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>ResourceBundle</td>
<td>Any constructor, Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>Row</td>
<td>getAttribute(), getAttributeHints(), getKey(), getLookupDescription(), getOriginalAttributeValue(), getPrimaryRowState(), getSelectedListDisplayValue(), getSelectedListDisplayValues(), isAttributeChanged(), isAttributeUpdateable(), remove(), revertRow(), revertRowAndContainees(), setAttribute(), validate()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>RowId</td>
<td>Any constructor, Any method</td>
<td>java.sql</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>RowIterator</td>
<td>createAndInitRow()</td>
<td>oracle:jbo</td>
</tr>
<tr>
<td></td>
<td>createRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findByKey()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>findRowsMatchingCriteria()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>first()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getAllRowsInRange()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getCurrentRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getEstimatedRowCount()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasNext()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>hasPrevious()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>insertRow()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>last()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>next()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>previous()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>reset()</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>RowSet</td>
<td><code>avg()</code> <code>count()</code> <code>createAndInitRow()</code> <code>createRow()</code> <code>executeQuery()</code> <code>findByPrimaryKey()</code> <code>findRowsMatchingCriteria()</code> <code>first()</code> <code>getAllRowsInRange()</code> <code>getCurrentRow()</code> <code>getEstimatedRowCount()</code> <code>hasNext()</code> <code>hasPrevious()</code> <code>insertRow()</code> <code>last()</code> <code>max()</code> <code>min()</code> <code>next()</code> <code>previous()</code> <code>reset()</code> <code>sum()</code></td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Scanner</td>
<td>Any constructor Any method</td>
<td>java.util</td>
</tr>
<tr>
<td>SecurityContext</td>
<td><code>getUserProfile()</code> <code>getUserProfile()</code></td>
<td>oracle.adf.share.security</td>
</tr>
<tr>
<td>Session</td>
<td><code>getLocale()</code> <code>getLocaleContext()</code> <code>getUserData()</code></td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>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>String</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringBuffer</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringBuilder</td>
<td>Any constructor</td>
<td>java.lang</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>StringTokenizer</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------</td>
<td>-----------------</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(), getAttributeIndexOf()</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>Timestamp</td>
<td>Any constructor, Any method</td>
<td>java.sql</td>
</tr>
<tr>
<td>Timestamp</td>
<td>compareTo(), dateValue(), getValue(), stringValue(), timeValue(), timestampValue()</td>
<td>oracle.jbo.domain</td>
</tr>
<tr>
<td>TreeMap</td>
<td>Any constructor, Any method</td>
<td>java.util</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>TreeSet</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>UUID</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UserProfile</td>
<td>getBusinessCity()</td>
<td>oracle.adf.share.security.identitymanagement</td>
</tr>
<tr>
<td></td>
<td>getBusinessCountry()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessEmail()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessFax()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessMobile()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPOBox()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPager()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPhone()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPostalAddr()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessPostalCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessState()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getBusinessStreet()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDateofBirth()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDateofHire()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDefaultGroup()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDepartment()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDepartmentNumber()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDescription()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getDisplayName()</td>
<td></td>
</tr>
<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>
</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>ValidationException</td>
<td>getDetails()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td></td>
<td>getErrorCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getErrorParameters()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getLocalizedMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getMessage()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProductCode()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>getProperty()</td>
<td></td>
</tr>
<tr>
<td>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>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ViewCriteriaItem</td>
<td>getValue() makeCompound() setOperator() setUpperColumns() setValue()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>ViewCriteriaItemCompound</td>
<td>ensureItem() getValue() makeCompound() setOperator() setUpperColumns() setValue()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>ViewCriteriaRow</td>
<td>ensureCriteriaItem() getConjunction() isUpperColumns() setConjunction() setUpperColumns()</td>
<td>oracle.jbo</td>
</tr>
<tr>
<td>Class Name</td>
<td>Allowed Methods</td>
<td>Package</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>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>
<tr>
<td></td>
<td>setMaxFetchSize()</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sum()</td>
<td></td>
</tr>
<tr>
<td>WeakHashMap</td>
<td>Any constructor</td>
<td>java.util</td>
</tr>
<tr>
<td></td>
<td>Any method</td>
<td></td>
</tr>
</tbody>
</table>
Accessing View Objects in Scripts: 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, 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 you will do with this new view object instance is to query some data. You do this by calling the findByKey() function on the view object to find a row by key.

For more information on groovy scripts, see Groovy Scripting Reference for Application Composer (ID 1354807.1) on My Oracle Support at https://support.oracle.com. In particular, refer to "Accessing the View Object for Programmatic Access to Business Objects" for details on accessing view objects.

This topic:

- 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.

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.
2. In the expression builder palette, on the Functions tab, select the Other category and the newView() function.
3. Click Insert.

A window will display which lists the view objects you can call in your script.
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, you must refer to Oracle Enterprise Repository for Oracle Fusion Applications to view a list of attributes that you can script against.

2. Click Login as Guest.
3. Under Type, select ADF Service Data Object.
4. Under Fusion Apps: Logical Business Area, select the desired logical business area, such as Customer Data Management.
5. Click Search.
6. Select the desired object, and click the Details tab to view the attributes you can use in your scripts.

Tip

From OER, you can also review object model diagrams by querying for Data Model Diagram types of records, for a desired logical business area.

<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 does not 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 does not have a view link to this object. Access this object as a child of the Account or Contact objects.</td>
<td>Refer to the Trading Community Classification Code Assignment in OER ADF Service Data Object.</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>Standard View Object</td>
<td>Description</td>
<td>Typical Attributes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Contact</td>
<td>Use this object to access customer contact information in scripting, if the current object does not have a view link to this object. Access this Customer Contact Profile extensible object as a child of the Account, Contact, or Household objects.</td>
<td>Refer to the Customer Contact Profile object in Application Composer, and review the descriptions provided for all attributes.</td>
</tr>
<tr>
<td>FndTreeVersion</td>
<td>Use this object in scripting to access tree versions. The customer hierarchy and party hierarchy are modeled as trees.</td>
<td>TreeStructureCode, TreeCode, TreeVersionID, Status, EffectiveStartDate, EffectiveEndDate, TreeVersionName</td>
</tr>
<tr>
<td>FndTreeNode</td>
<td>Use this object to determine the parent/child relationships for a given hierarchy. The hierarchy for a given version is stored in this object.</td>
<td>TreeStructureCode, TreeCode, TreeVersionID, TreeNodeID, ParentTreeNodeID, Depth, ChildCount, ParentPk1Value</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. 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>Refer to the Trading Community Location SDO in OER ADF Service Data Object.</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 do not have a view link from the current object to the Account object.</td>
<td>Refer to the Trading Community Organization Details in OER ADF Service Data Object.</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>Standard View Object</td>
<td>Description</td>
<td>Typical Attributes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>--------------------</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 OER ADF Service Data Object.</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 do not have a view link from the current object to Account object.</td>
<td>Refer to the Trading Community Person Details in OER ADF Service Data Object.</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 does not 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 does not expose a view link to this object.</td>
<td>Refer to the Trading Community Resource Profile in OER ADF Service Data Object.</td>
</tr>
</tbody>
</table>

**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 Oracle Sales Cloud provide global functions ready for your use, or you can define new global functions.

For more information on global functions, see Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com. In particular, refer to “Defining Utility Code in a Global Function” for details on using global functions in your scripts, including examples.

This topic:

- Explains how to define new global functions.
- Provides a list of some global functions that are provided with the Oracle Fusion Common CRM application.
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.
3. On the Global Functions page, click the New icon.
4. When defining a function, specify the global function name and a return value.
5. Optionally specify one or more typed parameters that the caller will be required to pass in when invoked.
6. Specify the body of the function.
7. Validate and save your function.

Examples of Predefined Global Functions

This table lists the global functions that are provided with the Oracle Fusion Common CRM 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>
<tr>
<td>adf. util. getUserRootResourceOrgId()</td>
<td>Returns the organization_ ID for the logged-in user’s organization hierarchy root resource organization.</td>
</tr>
</tbody>
</table>

Calling Web Services from Groovy Scripts: Explained

You can call 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. 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 invoke 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 "Creating 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.
   
   The code that will be inserted is shown under *Function Signature*.
4. Click the **Insert** button to insert the code to invoke the Web service method.

As you can see in the figure, a Web service call from a Groovy script has the following syntax:

```
adf.webServices.YourServiceVariableName.MethodName(args)
```

The information under function signature includes the parameter types and also the return type to indicate the type of variable the result of the call should be assigned to. The possible return types are as follows:

<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>Return Value</td>
<td>Return Type</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Object</td>
<td>Map</td>
</tr>
<tr>
<td>Collection</td>
<td>List</td>
</tr>
</tbody>
</table>

For examples of Groovy code that calls a Web service, read "Web Service Calls in Groovy Scripts: Examples".

**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",
```
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:


For more information on maps and lists, see the section called Working with Maps in the Oracle Fusion CRM Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

Web Service References for Groovy Scripts: 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 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.
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.
After you create a Web service reference, the name of the Web service appears in the list available in the Web services tab in the Expression Builder. When you select a Web service from the list, you can then select any of the functions provided by the Web service for use in your Groovy scripts.

You can edit existing Web service references, for example, to change the security scheme used or the settings used for a particular security scheme.

**Tip**

When managing Web service references, click the **Refresh** icon in the Web Services page to make sure the list is up to date. Read "Refreshing the List of Web Service References" below for information about when you need to click **Refresh**.

**Specifying Variable Names**

When you create a Web service reference, you specify a variable name on the Create SOAP Web Service Connection page. This name is simply an identifier that is used in the list of Web services in the Expression Builder.

**Specifying WSDL URLs**

The WSDL file for a Web service provides information about a Web service that includes the following:

- **Service.** Defines one or more ports for the Web service.
- **Port.** Defines an address or connection endpoint to the Web service.

For each service and port there can be one or more associated security policies. To specify a WSDL URL:

1. On the Create SOAP Web Service Connection page, enter the WSDL file in URL format, for example:
   
   http://internal-hosted:7101/MathsWS-Model-context-root/
   
   UsernameTokenSecurity?wsdl

2. Click **Read WSDL.**
   
   The **Service**, **Port**, and **Security Scheme** fields are then populated based on what is found in the WSDL. When there are multiple services and ports defined, the **Service** and **Port** dropdowns 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 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:

- Using separate user name and password credentials over Secure Sockets Layer (SSL)
- Using separate user name and password credentials with message protection
• Using single sign-on through Security Assertion Markup Language (SAML).

You can also specify that no security scheme is used.

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 not only in new methods but changing of the signature of existing methods, or deletion of existing methods as well.

You might also want to click Refresh to display any new Web service references that have been created in a separate user session.

Web Service Calls in Groovy Scripts: 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. This topic provides examples of calling Web service methods from Groovy scripts.
A Web service call from a Groovy script has the following syntax:
```
adf.webServices.YourServiceVariableName.MethodName(args)
```
In the examples in this topic, methods of a Web service registered with the variable name EmployeeService are called.

**Note**
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, 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 a New Employee Including New Dependents
The following example shows how to use Groovy'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 New Child Dependent 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 emp object retrieved, updates the names of the existing dependents, and then adds a new dependent before calling the mergeEmployee() method of the Web service to save the changes.

```groovy
// 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 from Groovy when No Security Scheme is Required: Worked Example

This example shows how to create a connection to an external 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://externalhosted">http://externalhosted</a> 7101/ MathsWSModel context-root/NoSecuritywsdl</td>
</tr>
<tr>
<td></td>
<td>Note</td>
</tr>
<tr>
<td></td>
<td>The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,</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>
</tbody>
</table>

To call a Web service from a Groovy script when no security scheme is required, complete the following tasks:
1. Create the Web service connection.
2. Add the Web service call to the Groovy script, and verify that the call succeeds.

**Prerequisites**

Verify that you have completed the following prerequisite steps:
1. Get details of the WSDL URL to use from the Web service provider.
2. Create a custom field for an object that has a calculated default value.
3. Prepare the Groovy script for the expression used to calculate the field’s default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.
Creating the Web Service Connection

When you create a Web service connection, you specify a name for the Web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of Web services in the Expression Builder in Application Composer.

1. In Application Composer, select **Web Services** in the Common Setup pane.

2. On the Web Services page, click the New icon.

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 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 External Web Service from Groovy with Message Protection: Worked Example

This example shows how to create a connection to an external, secured 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>http: // externalhosted 7101/ MathsWSModel context- root/ Wss11UsernameWithMessageProtectionSecuritywsdl</td>
</tr>
</tbody>
</table>

*Note*  
The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
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.

<table>
<thead>
<tr>
<th>Decisions to Consider</th>
<th>In This Example</th>
</tr>
</thead>
<tbody>
<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 timestamp in the response from the Web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td></td>
<td>To ignore the timestamp, you select the Disable Timestamp Verification check box. This may be required to address interoperability issues.</td>
</tr>
</tbody>
</table>
2. On the Web Services page, click the New icon.

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 Invoke 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 timestamp verification so that the timestamp 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 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.
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 timestamp verification so that the timestamp 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 from Groovy with Separate User Credentials over SSL: Worked Example

This example shows how to create a connection to an external, secured 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://externalhosted7102/MathsWSModelcontext-root/UsernameTokenOverSSLSecuritywsdl">https://externalhosted7102/MathsWSModelcontext-root/UsernameTokenOverSSLSecuritywsdl</a></td>
</tr>
<tr>
<td>This WSDL file specifies the desired SSL security scheme.</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>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>This method returns the sum of two integer argument values.</td>
<td></td>
</tr>
<tr>
<td>Is it required to ignore the timestamp in the response from the Web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td>To ignore the timestamp, you select the Disable Timestamp Verification check box. This may be required to address interoperability issues.</td>
<td></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.
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](image)

   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 Invoke with separate user credentials over SSL radio button becomes enabled and selected and the Credential Key field appears.
8. Click the New Key icon next to the Credential Key field.

9. In the Create Key dialog box, enter a name in the Credential Key field, in this example, mylogin, enter the user name and password credentials supplied by the Web service provider, and click OK.

10. Select Disable timestamp verification so that the timestamp 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 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.

**Calling an Internal Web Service from Groovy with Separate User Credentials over SSL: Worked Example**

This example shows how to create a connection to a Fusion Applications 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>
</tbody>
</table>
### Decisions to Consider

<table>
<thead>
<tr>
<th>What is the URL of the Web Services Description Language (WSDL) file that you will use?</th>
<th><a href="https://internalhosted/7102/MathsWSModel/context-root/UsernameTokenOverSSLSecuritywsdl">https://internalhosted/7102/MathsWSModel/context-root/UsernameTokenOverSSLSecuritywsdl</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>This WSDL file specifies the desired SSL authentication scheme.</td>
<td></td>
</tr>
</tbody>
</table>

**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. |

<table>
<thead>
<tr>
<th>Which Web service method will be called from the Groovy script?</th>
<th>getSum</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method returns the sum of two integer argument values.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Is it required to ignore the timestamp in the response from the Web service?</th>
<th>Yes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>To ignore the timestamp, you select the Disable Timestamp Verification check box. This may be required to address interoperability issues.</td>
<td></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.

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 Invoke 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 timestamp verification so that the timestamp 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 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 Message Protection Security: Worked Example**

This example shows how to create a connection to a Fusion Applications 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>mathswns</td>
</tr>
<tr>
<td>Decisions to Consider</td>
<td>In This Example</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</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://internalhosted:7101/MathsWSModel">http://internalhosted:7101/MathsWSModel</a> context-root/Wss11UsernameWithMessageProtectionSecuritywsdl</td>
</tr>
<tr>
<td>This WSDL file specifies the desired message protection security scheme.</td>
<td></td>
</tr>
<tr>
<td>Note</td>
<td>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>This method returns the sum of two integer argument values.</td>
<td></td>
</tr>
<tr>
<td>Is it required to ignore the timestamp in the response from the Web service?</td>
<td>Yes.</td>
</tr>
<tr>
<td>To ignore the timestamp, you select the Disable Timestamp Verification check box. This may be required to address interoperability issues.</td>
<td></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 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.

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 **Invoke 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 timestamp verification** so that the timestamp 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 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 from Groovy using SAML for ID Propagation: Worked Example

This example shows how to create a connection to a Fusion Applications 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><code>mathsws</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? | https://internalhosted7102/MathsWSModel Context- root/SamlOrUsernameTokenWithMessageProtectionwsdl

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 timestamp in the response from the Web service? | Yes.
To ignore the timestamp, you select the Disable Timestamp 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.
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 **Invoke with current user credentials using SAML** radio button becomes enabled and selected.

This figure shows the Create SOAP Service Connection window.

5. Select **Disable timestamp verification** so that the timestamp 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 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.
Application Composer: Creating Object Workflows

Creating Object Workflows: Overview

Read this chapter to understand how you work with object workflows, which are automated workflows that you trigger based on your specified conditions. In this chapter, you will learn about:

- The concepts and terminology used in object workflows.
- How to 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

You configure object workflows to automate business processes. You set a trigger condition for these workflows and specify the actions that the workflow must perform when the trigger condition is met.

Familiarize yourself with the following terms before you start working with object workflows:

- Business Object: A business object or an object can be a standard object delivered with the Sales Cloud or a custom object that you define based on your business need. When configuring your workflow, the fields available for selection or update belong to the business object that you select.
- Active: The Active check box indicates whether the workflow is in use. To stop a workflow from triggering event actions, clear the Active check box for that workflow.
- 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.
- E-Mail Notification.
- Task Creation.
- Outbound Message.
- Business Process Flow.

This figure illustrates how you configure and trigger object workflow.

This figure illustrates the create object workflow page.
Application Composer

You can use the Application Composer to customize standard business objects delivered as is or create custom business objects based on business needs. You can use both standard objects and custom objects to define object workflows. How you create and configure these objects determines and enforces that only the data relevant to the object in context is available for selection when defining a workflow. 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.

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 a 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 via 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
```

In addition, keep these points in mind when using the expression builder to build raise conditions:

- Ensure that you return a valid boolean as part of your raise condition. Returning a non-boolean value could lead to run-time 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 run-time errors.
- For lookup values, use the lookup code instead of the display value.
- Use `<`, `>`, `==` for comparison.
- For the event point *When a record is updated*, avoid redundant calls of the actions by always specifying which field change should trigger the object workflow, using the function `isAttributeChanged`.
- Be aware of Groovy-type coercion, if you are not returning an explicit boolean value.
- Use the logging capability to debug your condition and review the generated log by selecting *Runtime Messages* in the Common Setup pane.

For more information on groovy scripting, see Oracle Sales Cloud Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

**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.

**Object Workflows and Application Composer: How They Work Together**

Application Composer is a browser-based configuration tool. It is part of Oracle Sales Cloud extensibility framework that provides a mechanism to
extend applications. You must have administrative privileges to work with and configure object workflows using the application composer.

This figure illustrates what you can do using Application Composer within the extensibility framework.

**Application Composer**

You can use the Application Composer to customize standard business objects delivered as is or create custom business objects based on business needs. You can use both standard objects and custom objects to define object workflows. How you create and configure these objects determines and enforces that only the data relevant to the object in context is available for selection when defining a workflow. 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.
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**

Using object workflows, you can configure automatic field updates by specifying what fields you want to update and the value to apply to those fields.

This figure illustrates how the field updates are configured and triggered.
To create field updates event 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.
4. Define the trigger condition using When a record is created or When a record is updated event point.

**Note**
If you have created a field in a sandbox and want to set a condition on that field or apply a new value, you must first publish the sandbox to make that field available to the mainline code application.

5. Select Field Updates as the event action.
   This opens the Create Action: Field Updates page.
6. Provide the Name and Description for the field updates action and optionally set the Execution Schedule.
7. Specify the fields you want to update when the workflow is triggered. See: Specifying Field Update Details in this topic.
8. Save the event action.

**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 Field Update Details**

In the Field Update Details region, select the fields and specify the values to which these fields must be set when the workflow is triggered. When you select a field to update, the options on the page dynamically change depending on the field type. For example, if you select a field of type Date, a date picker appears.
If you are updating a currency field, you must specify the value to which this field will be set when the field updates action is invoked.

In case of static choice list, the list of field values can either be in an alphabetical order or in the display sequence such as High, Medium, or Low. For updating static fields that have a display sequence, specify whether you want to populate the field with the next value in list or with the previous value in 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 field updates; however, 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.

Use **Update More Fields** option to select additional fields to update.

**Primary and Child Object Support**

You can configure field updates using either a parent or a child object. When you select a child object for defining an object workflow, you can update fields only in the records of that child object; however, you cannot update fields in the records of that child object's parent.

For example, if you are 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 in context is available for selection.

**Same-Object Field Updates**

The field-updates event action acts on the same object that raises the event. For example, when you create an **Opportunity** object and do not specify the **Close Date**, you can define a field update action that sets the **Close Date** automatically to 90 days after the creation date.

**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.

**Object Workflows and E-Mail Notification: How They Work Together**

You can configure object workflows to send e-mail communications to the intended recipients as an automated process.
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 that opportunity along with the required information.

This figure illustrates how e-mail notifications are configured and triggered.

You can configure e-mail notifications using either a parent or a child object.

To define an e-mail notification:

1. From the Application Composer main page, select **Object Workflows**.
2. Click **Create**.
3. Select an **Object** and provide a meaningful **Name** and **Description**.
4. Define the trigger condition using **When a record is created** or **When a record is updated** event point.
5. Select **E-Mail Notification** as the event action.
   
   This opens the Create Action: E-Mail Notification page.
6. Provide the **Name** and **Description** for the e-mail notification action and optionally set the **Execution Schedule** for triggering such notifications.
7. 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.
8. Specify the **Recipient Types**. See: Recipient Types in this topic.
9. Click **Save**.

**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.

**Recipient Types**

The recipients available for sending e-mail notifications are contextual. If you are defining a workflow using a parent object, then 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. Additionally, when inserting field tokens while configuring e-mail notifications, only those tokens belonging to the selected primary or child object are available for selection.

Select the recipients from the following options available under Recipient Type:

- **E-mail fields on record**: If the selected object contains fields that store e-mails, you can select those fields to send e-mail notifications. For example, Primary Contact E-Mail under Opportunity object.
- **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 direcets, or a team of resources working on an object, and so on.
- **Roles**: Contains users assigned to a particular resource role. Enterprise groups or users granted with that role could be the recipients of the e-mail notification. 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.
- **Specific users**: Contains details of individual users and their e-mails.
- **Specific e-mail addresses**: Enter individual e-mails separated by a comma.

**Object Workflows and Task Creation: How They Work Together**

Tasks in Oracle cloud services integrate with BPEL human tasks and are available in the BPEL worklist application. 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.

When you configure an object workflow using a child object, the selectable fields for Tasks Creation event action contain fields 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.

This figure illustrates how tasks are configured.
To define a tasks creation 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**.
4. Define the trigger condition using **When a record is created** or **When a record is updated** event point.
5. Select **Tasks Creation** event action.
   This opens the Create Action: Tasks Creation page.
6. Provide the **Name** and **Description** for the event action and optionally set the **Execution Schedule**.
7. Use the **Task Details** region to configure a task based on your requirements. Here are the things you can do using tasks details region:
   - Select a **Subject** and **Description**, as appropriate, for your task notification.
   - Select the **Start Date** and **Due Date** for a task. When you select a date, you can also provide logical conditions for these dates, for example, a **Due Date** must be 30 days after the **Start Date**.
   - Select the **Owner** and **Assignees** for a task. The values in these lists change depending on the object for which the task is being defined.
     For example, for an opportunity object, the **Assignees** list would include the owner, resource team, resource team with different access levels, member functions, and so on. When an opportunity reaches a **Close Date** and the status is still **Open**, you can use the task creation action to assign a follow up task to the owner of the opportunity.
   - Select the **Category** to which a task belongs, such as meeting, administration, and so on.
   - Assign a **Priority** to the task. The default priority is 3.
8. 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.

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.

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, 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. 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.

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.

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.

Navigate to Oracle Enterprise Repository (OER):

1. For a standard object, search for **ADF Service** in OER 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_NAMESPACE$` and `OBJECT_NAME$`. 
Examples of OutboundMessageService.wsdl and OutboundMessageService.xsd are provided here for reference.

**WSDL File Example**

This section includes an example of the OutboundMessageService.wsdl, for your reference.

```xml
<wsdl:definitions name="OutboundMessageService"
  targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/">
  <wsdl:import namespace="http://xmlns.oracle.com/adf/svc/errors/">
    <wsdl:location="ServiceException.wsdl"/>
  </wsdl:import>
  <wsdl:import namespace="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/">
    <wsdl:schemaLocation="OutboundMessageService.xsd"/>
  </wsdl:import>
  <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:operation name="processOutboundMessage">
    <wsdl:input message="tns:OutboundMessageService_processOutboundMessage"/>
    <wsdl:output message="tns:OutboundMessageService_processOutboundMessageResponse"/>
    <xsd:fault name="ServiceException" message="errors:ServiceException"/>
  </wsdl:operation>
  <wsdl:portType name="OutboundMessageService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <soap:operation name="processOutboundMessage" soapAction="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/processOutboundMessage"/>
    <xsd:input message="tns:OutboundMessageService_processOutboundMessage"/>
    <xsd:output message="tns:OutboundMessageService_processOutboundMessageResponse"/>
    <xsd:fault name="ServiceException"/>
  </wsdl:portType>
</wsdl:definitions>
```
<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 provides an example of the OutboundMessageService.xsd, for your reference. The parameters are marked in the .xsd file as parameters, $OBJECT_TARGET_NAMESPACES$ and $OBJECT_NAME$.

//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="$OBJECT_TARGET_NAMESPACES$"
xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/"
xmlns:tns="http://xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
xmlns="http://www.w3.org/2001/XMLSchema">
<import namespace="http://xmlns.oracle.com/adf/svc/types/
schemaLocation="BC4JService.xsd"/>
<import namespace="$OBJECT_TARGET_NAMESPACES$"
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/ws10_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/ws10_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.
E-Mail Templates: Explained

You can create, manage, and use e-mail templates for configuring e-mail notifications as part of 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 must have administrative privileges to define e-mail templates.

A template is created for a particular type of object and can be used only with that object when defining e-mail notifications. You can compose e-mail templates using Application Composer. You can also create using any e-mail editor and then upload the template using application composer. You can use e-mail templates, for example, for including your company letterhead in outbound e-mail communications.

You can search an e-mail template using either or all of the search parameters such as full or partial name of the template, the object to which the template is associated, and whether the template is active.

Managing E-Mail Templates

Managing e-mail templates include tasks that enable you to search, view, duplicate, create, edit, or delete a template.

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.

Note

You cannot change the object for which a template is defined.

- Creating and configuring a new template.
- Detaching the listed templates for a full-page view.
- Viewing which templates are active.

Note

Only an active template can be used for object workflow definition. Saving does not automatically make the template active.

Configuring E-Mail Templates

You can define a template using the formatting features as available in any HTML based e-mail client. Templates are automatically converted to plain text for users who cannot view HTML e-mails. Advanced HTML users can edit the templates using the source code editing mode where the tags are set to visible. These advanced users can copy and edit the source code in any HTML editor, and then paste the edited code back.

When defining an e-mail template, you specify the tokens or fields which are populated with values at run time.

Note
When specifying tokens, fields of type Dynamic Choice List will not be available for selection.

**To configure an e-mail template:**

- Specify the **Name** and **Description** of the template.
- Attach artifacts relevant to the e-mail template. **Attachments** are included in every e-mail that uses the template having an attachment. You can also send attachments as links to avoid crowding the Inbox of the recipients. An attachment can be a file on a local computer or a shared file in a repository.

**Note**

File attachments cannot exceed 10 MB.

- Specify whether an e-mail template is **Active**.

**Note**

You can use only **Active** templates when creating e-mail notifications.

- Insert **Fields** or **Functions**, as applicable. You can use **Functions** to insert **Date**, **Current date and time**, or a **Hyperlink** to record. You can also **Browse** and then insert the content of a local file in the body of the e-mail template.
- 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.

---

**Configuring Object Workflows: 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 will 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 Field Updates event action.
3. Configuring 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 will 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 more... >> link.
3. Click the Application Composer link.

You are on the Overview page of Application Composer. 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, Note, Interaction, and so on. In this activity, you are configuring a workflow for the Opportunity object; therefore, you will select Sales from the Application list.

4. In the left pane, click the Application list.
5. Click the Sales list item.
6. In the Overview region, click the Object Workflows link.

You are on the Object Workflows page. You can use this page to search for an existing workflow or create a workflow. In this activity, you will create a workflow.

7. Click the Actions menu below the Search region.
8. Click the Create menu item.
9. You are on the Create Object Workflow page. You must first select an object for which you are creating a workflow. Click the Object list.
10. Click the Opportunity list item.
11. Enter the desired information into the Name field. Enter "Budget Date Revised".
12. In the Event Point and Condition region, click the When a record is updated option.
13. You will 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.
14. You are on the Expression Builder popup. Before you proceed, ensure that the Fields tab is selected.
15. 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.
16. Insert Date Budget Available into the expression builder. Click the Insert button.
17. 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 will 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; }
   ```

18. Validate your script. Click the Validate button.

19. Click the OK button.

20. You have set the event point and trigger condition for your object workflow. You will now create a Field Updates event action. In this event action, you will 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 will 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.

2. You are on the Create Action: Field Updates page. Enter the desired information into the Name field. Enter "Update Close Date".

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 the desired information in the Days field. Enter "7".

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 are back to the Create Object Workflow page. You will now 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 will 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 the desired information into the **Name** field. Enter "Follow Up Call".

3. In the Task Details region, enter the desired information into the **Subject** field. Enter "Follow up with customer on budget available date".

4. You will 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.

8. Notice that the [$TargetPartyName$] token is inserted in the Subject where you placed your cursor. You will need to adjust trailing or leading text-spaces as required.

9. Enter the desired information into the **Description** field. Enter "Opportunity Budget Available Date is changed. Follow up with within three days of the new available date. Primary Contact Email ID: ".

10. 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 ([$PrimaryContactEmailAddress$])
11. Click after "Opportunity" in the Description text.
12. Click the field-token list on the right of the Description field.
13. Click the Opportunity Name list item
14. Click the <<Insert button
15. Click after "Follow up with" in the Description text.
16. Click the field-token list on the right of the Description field.
17. Click the Primary Contact list item.
18. Click the <<Insert button
19. Click after "Email ID:" in the Description text.
20. Click the Primary Contact E-Mail list item.
21. Click the <<Insert button
22. You will now set the task's Due Date to be 3 days after the new Budget Available Date. Click the Due Date list.
23. Click the Date Budget Available list item.
24. Make sure the '+' operator is selected, and use the increment button (up arrow) to set the value in the Days field to 3.
25. Click the Start Date list.
26. Click the Date Budget Available list item. Accept the default 0 (zero) in the Days field.
27. Click the Owner list.
28. Click the OwnerResourcePartyId list item.
29. Click the Assignees list
30. Click the OpportunityResource Team list item.
31. Click the Category list.
32. Click the Call, outbound list item.
33. Click the Priority list.
34. Click the 1 - Very high list item.
35. Save the Task Creation event action. In the upper-right region of the page, click the Save button.
36. 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 will now create the E-Mail Notification event action. Before you create this event action, you must first create an E-Mail Template that you will 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 will create an e-mail template, which you will use for sending e-mail notifications.
1. In the Common Setup pane on the left, click the **E-Mail Templates** link.

2. 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 will create a template.

3. In the Search Results region, click the **Actions** menu.

4. Click the **Create** menu item.

5. You are on the Create E-Mail Template page. Click the **Object** list.

6. Click the **Opportunity** list item.

7. Enter the desired information into the **Name** field. Enter "**Budget Available Date Update**".

8. Enter the desired information into the **E-Mail Subject** field. Enter "**Opportunity customer budget available date has changed**".

9. You will 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.

10. Click the field-token list on the right of the **E-Mail Subject** field.

11. Click the **Opportunity Name** list item.

12. Click the **Insert** button

13. In the E-Mail Body region, enter the desired information into the **E-Mail Body** field. Enter "**- budget available date regarding opportunity has moved to . The new opportunity close date is .**".

14. 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.

15. Click after "- " in the e-mail body.

16. In the E-Mail Body region, click the field-token list on the left of the **Insert** button.

17. Click the **Customer** list item.

18. Click the **Insert** button.

19. Click after "regarding opportunity" in the e-mail body.
20. Click the field-token list.
21. Click the **Opportunity Name** list item.
22. Click the **Insert** button.
23. Click after "has moved to" in the e-mail body.
24. Click the field-token list.
25. Click the **Date Budget Available** list item.
26. Click the **Insert** button.
27. Click after "close date is" in the e-mail body.
28. Click the field-token list.
29. Click the **Close Date** list item.
30. Click the **Insert** button.
31. Save the E-Mail Template. In the upper-right region of the page, click the **Save** button.
32. You are back to the E-Mail Templates page. In the Common Setup pane on the left, click the **Object Workflows** link.
   You will now create an E-mail Notification event action using the e-mail template you just created.
33. In the Search region, click the **Object** list.
34. Click the **Opportunity** list item.
35. Click the **Search** button.
36. The **Name** column in the search results lists the object workflows created for the opportunity object. From this list, identify and select the object workflow that you are creating. Click the **Budget_Date_Revised** cell.
37. Click the **Edit** button.
38. You are on the Edit Object Workflow page. You are editing an existing object workflow to configure and add an E-Mail Notification event action.
39. In the Actions region, click the **Create** button on the right of the E-Mail Notification event action.
40. You are on the Create Action: E-Mail Notification page. Enter the desired information into the **Name** field. Enter "Notify sales team about budget available date change".
41. Select the e-mail template that you created. In the E-Mail Details region, click the **E-Mail Template** list.
42. From the list of e-mail templates, identify and select the template that you created. Click the **Budget Available Date Update** template.
43. Click the **Recipient Type** list.
44. Click the **Specific e-mail addresses** list item.
45. Enter the desired information into the **E-Mail Addresses** field.
46. Save the E-Mail Notification event action. In the upper-right region of the page, click the **Save** button.
47. You are back to the Edit Object Workflow page. 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 will now create.

**5. Creating an Opportunity Record**

In this section, 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 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**.
3. You are on the Overview page of Opportunities. You will use this page to 
   create an opportunity record. In the Opportunities region, click the **Create** 
   button.
4. You are on the Create Opportunity page. 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.
8. You are on the Search and Select: Sales Account popup. In the popup, 
   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 the desired information into the **Name** field. Enter "DG 150 Green 
    Servers".
13. Enter the desired information into the **Quantity** field. Enter "50".
14. Enter the desired information into the **Estimated Price** field. Enter "2000".
15. In the upper-right region of the page, click the **Save and Edit** button.
16. You are on the Edit Opportunity page. 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.

22. Currently there are no Tasks with your sign in initials.

23. In the Additional Details region, click the **Opportunity Team** tab.

24. In the Team Members table, locate your sign-in username. You will now add another team member: **Mateo Lopez**.

25. In the Opportunity Team tab, click the **Add Team Members** button.

26. You are on the Select and Add: Team Members popup.

27. Find a team member and click the **Done** button.

28. Search and add a contact to the opportunity. This will be the primary contact for the opportunity.

29. In the Additional Details region, click the **Contacts** tab.

30. In the Contacts tab, click the **Add Row** button.

31. Enter the desired information into the **Name** field. As you type in the contact name, the contact name and e-mail ID appear.

32. Click the desired contact.

33. In the upper-right region of the page, click the **Save and Close** button.

34. You are back to the Overview page of Opportunities. You will now edit the opportunity record you just created and update the budget availability date to trigger the event actions.

### 6. Editing the Opportunity Record to Trigger the Workflow and Verifying the Invoked Event Actions

Continuing from the previous step, you are on the Overview page of Opportunities. You will trigger (launch) the object workflow that you created in the previous steps by entering the budget availability date for an opportunity.

You will also verify the following after the workflow is invoked:

• Field Updates: Verify that the opportunity Close Date is set to 7 days from the new Budget Available Date. Verify that the Level of Risk is lowered for the opportunity and the Strategic Value is set to Medium
• Task Creation: Verify that a task is created for the opportunity owner to follow up with the customer.
• E-mail notification: Verify that an e-mail notification is sent to the entire opportunity team about the change in the opportunity’s Budget Available Date.

1. From the table in the Opportunities region, identify and select the opportunity record that you created. Click the **50 Solar Green Servers** link.

2. You are on the Edit Opportunity page. Before you proceed, expand the Show More area in the Summary region if not already expanded.

3. Under the Show Less area in the Summary region, click the **Date Budget Available** (calendar) button.

4. Click 9 as the date, or click any future date.

5. In the upper-right region of the page, click the **Save and Close** button.

   **Note**
   You have just changed the budget availability date and saved the record, 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 **50 Solar Green Servers** link.

   You are on the Edit Opportunity page. The workflow is now triggered and the configured event actions have been invoked. You will now verify the invocation of Field Updates event action.

7. In the Summary region, **Close Date** is now set to 7 days after the **Date Budget Available** value.

8. Under the Show Less area in the Summary region, **Strategic Value** is now set to **Medium**.

9. Under the Show Less area in the Summary region, **Level of Risk** has now been populated with the next value in the list, from **High** to **Low**.

10. You have verified the invocation of the Field Updates event action.

11. You will now 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 will use the **All Tasks** filter option to search for your invoked task.

12. In the Activity Center region, click the **Due Date** list under the Tasks tab.

13. Click the **All Tasks** list item.

14. In the Tasks tab, click the arrow button adjacent to the **Category** list.
15. 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 popup. Note these details in the popup:

- **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.

16. Click the **Done** button.

17. In the upper-right region of the page, click the **Save and Close** button.

18. You have verified the invocation of the Task Creation event action. Next, verify the invocation of the E-Mail Notification event action.

19. In your e-mail client, locate the **Subject** of your e-mail. Click the **Subject** link.

20. 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.

### Object Workflows and Business Processes: How They Work Together

This topic uses an illustration to help you understand how object workflows and business processes work together to trigger approval flows.

#### Overview

You create and deploy an approval flow (also called a project) using Oracle Business Process Composer using the seeded template. You then configure an object workflow with a business process flow event action and configure this event action to call the deployed approval flow. When you trigger the object workflow, the approval process is also triggered.

The approval flow and the object workflow are associated by way of input parameters, which supply the required inputs from object workflows to the approval flow. When the object workflow is triggered, the approval flow is also triggered which starts the approval process that you configured using business process composer.

There are three services involved in the approval flow process:

- **Application Composer**

  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 services.
For more information on how you customize and extend your cloud services using application composer, see Extending Oracle Sales Cloud Services: How It Works topic.

- **Oracle Business Process Composer**

  You can use Oracle Business Process Composer to orchestrate predefined components such as human-workflow tasks, services, and BPEL flows.

- **Oracle Business Process Management (BPM) Worklist application**

  Oracle BPM Worklist application displays tasks or approvals that are assigned to a user or a group. The worklist tasks appear on the Home page of Oracle Sales Cloud application, where the assignee approves or rejects an approval request.

---

**Note**

You do not perform any configuration in the worklist application as part of configuring object workflows and business processes.

---

This figure illustrates how object workflows and business processes work together.
You can use these automated flows to send approval requests, for example, for the following business scenarios. You can use both standard and custom objects to trigger workflows.

- When changing the win probability of an opportunity above a threshold level.
- When a new order is created.
- When a high-priority help request is raised.
- When a custom object has been updated.

These steps provide an overview of how approval flows are triggered using custom fields in a typical scenario. These steps have been described later in this topic using examples.

1. Create and add custom fields to the desired pages of an object. This object can be a standard or a custom object. You use these custom fields to trigger approval flows and track the status of approvals.

2. Create an approval flow in Oracle Business Process Composer using the seeded template, and save it as a project. You select this project when configuring the business process flow event action in object workflow.

3. Create an object workflow using the object to which you added the custom fields:
   a. Set a trigger condition on one of those custom fields.
   b. Select the Business Process Flow event action.
   c. Select the project you created in business process composer and map the input fields that you want to supply to that project (or approval flow) in business process composer.

4. Trigger the object workflow.
   This triggers the business process flow event action, which in turn triggers the approval flow.

**Creating a Custom Object and Adding Fields to Submit and Track Approvals**

This section provides an illustration to help you understand how you can submit and track approvals. The fields in this illustration have been used in examples later in this topic when configuring the object workflow.

1. Create a custom object and its work area.
   For more information on creating custom objects and how you customize and extend your applications, see Defining Objects: Explained and its related topics.

2. Create the following (custom) fields and add these to the desired pages of the custom object you just created.

---

**Note**

If you are creating these fields in a sandbox, you must publish the sandbox to make these fields available for use in the mainline code application.
a. **Approvalstatus_c** field: This is a drop-down, fixed-choice list field that holds the status of the approval. Use the lookup type ZCX_HOLD_STATUS to configure these status values as per your business requirement. The default values provided in this lookup type are: APPROVALPENDING, APPROVED, and REJECTED. To capture the approval status, you can use only fields of type text or fixed choice list.

When you are using the lookup type ZCX_HOLD_STATUS, ensure that it contains the default values or the values that you have specified in this lookup type for tracking the approval status.

b. **Submitforapproval_c** field: This is a check box field that you will use to submit approvals. You will use this field to set the trigger condition for object workflows.

**Creating and Deploying a Project in Oracle Business Process Composer**

This section covers how you create and deploy an approval flow (also called project) in Oracle Business Process Composer.

**Note**

You must have at least one business process flow deployed for use before defining an object workflow to trigger an approval process.

To access Oracle Business Process Composer:

1. Navigate to Application Composer using the main menu of the Oracle Public Cloud services.
2. From the Overview page of the Application Composer:
   a. Select the Business Processes option.
   b. On the Business Processes page, click **Create** under the **Actions** menu.
   c. Enter a meaningful name in the popup.
   d. Click **OK**.

   If popup blocker is enabled on your browser, the business process composer may not open after you click **OK**, and your browser may get locked. As a workaround, use the browser back button to go back to the Business Processes option in the application composer, and click **Edit** for the business process that you were creating.

   This opens the Oracle Business Process Composer in a separate Window.

   ![Diagram](image.png)

   This process is based on seeded template **ExtnBusinessProcess**, which uses the basic **CrmCommonSerialApprovalProcess** flow.
3. In the main menu, click **Deploy Project**.

4. In the **Deploy Project** popup 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>

5. Click **Deploy**.

A confirmation message appears when your project has been deployed successfully.


7. Verify whether the project has been deployed successfully.

   a. On the overview page of the application composer, click the Business Processes option

   b. On the Business Processes page, verify that a green check-mark appears under the **Deployed** column for the project you created.

You can also open Oracle Business Process Composer from object workflows.

1. Select the Object Workflows option in the application composer.

2. Create an object workflow with a Business Process Flow event action.

3. From the Create Action: Business Process Flow page, click the search icon for **Projects**.
4. Search a project using a name or directly select the business process flow from the list.

5. Click **Edit**.

   This opens the business process composer.

After you have successfully deployed your project, you can now configure the object workflow which triggers this deployed approval process flow.

**Setting Conditions and Parameters in Object Workflow to Trigger the Deployed Process**

This section covers how you set conditions and parameters in object workflow to trigger the approval flow. This section refers to the field names defined in "Creating a Custom Object and Adding Fields to Submit and Track Approvals" illustration in this topic.

For more information about object workflows, see related topic Object Workflows: Explained.

1. From the overview page of the application composer, select **Object Workflows**.

2. On the Create Object Workflows Page, select the object to which you added the custom fields.

3. Enter a name for your object workflow.

4. Define a trigger condition using When a record is updated event point.
   You set the trigger condition on the field that you created and have added to the required pages for submitting an approval.
   In the illustration, you added a **Submitforapproval_c** field of type check box, for which the trigger condition will be as follows:
   
   ```
   isAttributeChanged('Submitforapproval_c') && Submitforapproval=='Y'
   ```

5. Select the Business Process Flow event action.
   a. On the Create Action: Business Process Flow page, provide a meaningful name and description for the business process workflow.
   b. Specify an execution schedule that governs when the event action should be triggered.
   c. Search and select the **Project** that you deployed.

   This figure shows how you search and select a deployed business process. You can also create or edit a business process using this popup.

   ![Business Process Flow Details](image)

   d. Specify the **Input** parameters that you want to pass to the deployed approval flow.
To pass actual or static values, you must define the parameters as **Literal**. To do this, either select **Literal** to insert values when using the **Select Default Value** dialog, or type-in the parameters while enclosing them in single quotes, for example, 'user1'.

You can also obtain approvals serially from multiple approvers. For example, to obtain approvals first from user1 and then from user2 for static values, enter 'user1, user2'. Notice that you use single quotes for the entire string without any space in between the values. Similarly, for runtime fields, enter `user1, user2`.

To pass runtime values, do not define the parameters as **Literal**. In this case, you do not select **Literal** in the **Select Default Value** dialog, or you type-in the parameters without quotes. For example, to pass a runtime field for approver, enter `user1`.

This figure illustrates how you select the fields to map from object workflows.

You can map only **approvers** and **heldEntityStatusField** values.

In the illustration, you created a custom **Approvalstatus_c** field for capturing approval status; therefore, you must map the 'Approvalstatus_c' field with the reserved parameter **heldEntityStatusField**. Additionally, if the approvers are user1 and user2, specify these users as 'user1, user2'. Notice the use of quotes when supplying multiple values.

This figure illustrates the mapped fields under the inputs region.

You can also view these input parameters under **Properties** of the **Start** node in Oracle Business Process Composer. See related topic Object Workflows and Business Processes: Critical Choices.

6. Save the object workflow.
You have now created and deployed an approval flow, and configured the object workflow that triggers the approval flow.

7. Trigger the workflow.
   a. Navigate to the page where you added the `SubmitforApproval_c` check box field.
   b. Select the check box
   c. Save the page.

This satisfies the condition you set for object workflow. The business process flow event action is run, and it triggers the approval flow in the deployed project.

**Object Workflows and Business Processes: Critical Choices**

This topic covers critical choices and key points to consider when you work with object workflows to trigger a business process, which is specifically an approval flow.

**Overview**

You create and deploy an approval flow (also called a project) using Oracle Business Process Composer using the seeded template. You then configure an object workflow with a business process flow event action and configure this event action to call the deployed approval flow. When you trigger the object workflow, the approval process is also triggered.

The approval flow and the object workflow are associated by way of input parameters, which supply the required inputs from object workflows to the approval flow. When the object workflow is triggered, the approval flow is also triggered which starts the approval process that you configured using business process composer.

There are three services involved in the approval flow process:

- **Application Composer**
  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 services.

  For more information on how you customize and extend your cloud services using application composer, see Extending Oracle Sales Cloud Services: How It Works topic.

- **Oracle Business Process Composer**
  You can use Oracle Business Process Composer to orchestrate predefined components such as human-workflow tasks, services, and BPEL flows.

  For more information on Oracle Business Process Composer, refer Oracle Fusion Middleware Business Process Composer User’s Guide for Oracle
Oracle BPM Worklist application

Oracle BPM Worklist application displays tasks or approvals that are assigned to a user or a group. The worklist tasks appear on the Home page of Oracle Sales Cloud application, where the assignee approves or rejects an approval request.

Note

You do not perform any configuration in the worklist application as part of configuring object workflows and business processes.

Object Workflows with Approval Flows: Points to Consider

Following are some of the key considerations involved when working with approval flows and object workflows together:

1. **Deploy at least one project.**

   Ensure that a project has been deployed using the seeded template in Oracle Business Process Composer. You deploy a project to make it available for use for configuring object workflows. For information on deployment, see related topic Object Workflows and Business Processes: How They Work Together.

2. **Editing Projects.**

   You can edit a project only if the type of project is DEV or DEPLOYED_DEV. You cannot edit a project that does not exist in the Development partition of the Oracle Metadata Services (MDS) Repository, or if the type of project is DEPLOYED.

3. **Specifying trigger conditions.**

   Ensure that the trigger condition for the object workflow contains the field that you will be updating to trigger the workflow. For example, if you have defined a custom field `Submit_c` of type check box, the trigger condition would look like: `isAttributeChanged('Submit_c') && Submit_c=='Y'.`

   A trigger condition is a combination of event point and an expression that supports logical, math operations, or field-value lookups. 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 Object Workflows: Explained topic.

4. **Specifying input parameters.**

   Carefully specify the parameters that you want to map from object workflows to the approval flow in business process composer. The input parameters supply the fields or values to the business process flow when the object workflow is triggered.

   You can map only the following parameters for approval flows:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>Specifies the owner or the one who should receive the approval request.</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>Specifies the field that displays the status of the approval request.</td>
</tr>
</tbody>
</table>

To pass actual or static values, you must define the parameters as **Literal**. To do this, either select **Literal** to insert values when using the **Select Default Value** dialog, or type-in the parameters while enclosing them in single quotes, for example, 'user1'.

This figure illustrates how you select and mark the input parameters as literal.

To pass runtime values, do not define the parameters as **Literal**. In this case, you do not select **Literal** in the **Select Default Value** dialog, or you type-in the parameters without quotes. For example, to pass a runtime field for approver, enter `user1`.

You can also obtain approvals serially from multiple approvers. For example, to obtain approvals first from user1 and then from user2 for static values, enter `user1,user2`. Notice that you use single quotes for the entire string without any space in between the values. Similarly, for runtime fields, enter `user1,user2`.

To see where these input parameters reside in Oracle Business Process Composer, right click on the **Start** node and select **Properties**.

This figure illustrates the input parameters in business process composer.
Notice the `heldEntityStatus` and `approvers` parameters. These are the parameters you are mapping to from the object workflows.

5. **Using the seeded template.**

The seeded template `ExtnBusinessProcess` has the basic `CrmCommonSerialApprovalProcess` process flow. This template contains an approval process covering a typical Oracle Sales Cloud use case.

Any project you create with the seeded template will have `CrmCommonSerialApprovalProcess` process.

The seeded template contains services, business rules, and tasks which you can use for configuring approval flows. Select the Project Home tab to see the services, tasks, and rules available.
You can change or add a node 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.


### Sales Lead Approval Flow: Worked Example

This worked example shows you how to use object workflows and business processes to create and trigger an approval process. In this example, you are a sales administrator and your management has asked you to create a workflow to automate the approval process as follows:

1. The approval process should be triggered when a sales representative creates a sales lead with a deal amount greater than $100000.

2. When the approval process is triggered, the sales manager of that representative should automatically receive a request for approving the deal.

As a sales administrator, you will also test this approval process.
Step 1: Create a Field to Track the Approval Status

In this step, you will create a field to track the status of the approval and add this field to the desired pages.

1. Sign in to Oracle Sales Cloud Services as an administrator.
2. Create a sandbox and activate it.
   Sandboxes in Oracle Public Cloud Services provide robust out-of-the-box functionality to help isolate and control customization efforts without impacting other users’ configuration or production environment, or both. For more information on sandboxes, see Sandboxes: Explained topic.
3. Navigate to Application Composer.
   The Sales Lead object belongs to marketing service; therefore, select Marketing from the Application drop down.
4. In the left pane, open the Standard Objects tree.
5. Select the Sales Lead object.
6. Select the Fields node.
7. In the Custom tab, click Create under the Action menu.
   A Select Field Type popup appears.
8. Select Text in the popup.
9. Click OK.
   This opens a Create Text Field page.
10. In the Appearance region, enter the Display Label as Large Deal Approval.
11. Click Save and Close.
12. In the left pane, select the Pages node of the Sales Lead object.
13. Under Enterprise Pages tab, click Create Lead.
14. In the Create Lead page, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
15. Click Save and Close.
   You have now added the custom approval field to the create lead page.
16. Under Enterprise Pages tab, click Show Details.
17. In the Show Details page, move the Large Deal Approval field from the Available Fields box to the Selected Fields box.
18. Click Save and Close.
   You have now added the custom approval field to the show details page.

At this point, you are working within your sandbox. You have created a custom field and have added that field to two pages.

Note
Configurations done within a sandbox for object workflows and business processes are not available to the mainline code service unless published.

19. Publish your sandbox to make these changes available for use in the mainline code service.

Step 2: Publish the Sandbox and Verify the Added Field

In this step, you will publish your sandbox and verify that the Large Deal Approval field has been added to the Create Lead and Show Details pages. When publishing a sandbox, the included services customizations overwrite the mainline code service’s existing configuration. 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, expand the Settings and Actions menu which is available next to your user name. Then select Manage Sandboxes, under the Administration subheading.
2. Select your sandbox and click Publish.
3. Click the Navigator menu at the top.
4. Click Lead Qualification under Sales.
5. Click Create Lead.
6. Verify that Large Deal Approval field has been added to the Create Lead page.
7. Click Cancel. You will be creating a deal later in this example.

Step 3: Create and Deploy a Business Process

In this step, you will create a business process (also called project) in Oracle Business Process Composer using the seeded template and deploy it.

To create and deploy a process using Oracle Business Process Composer:

1. Navigate to Application Composer using the main menu of Oracle Sales Cloud Services.
2. From the Overview page of the Application Composer:
   a. Select the Business Processes option.
   b. On the Business Processes page, click Create under the Actions menu.
   c. Enter the Name as Sales Lead Approval in the popup.
   d. Click OK.

   If a popup blocker is enabled on your browser, the business process composer may not open after you click OK, and your browser may get locked. As a workaround, use the browser back button to go back to the Business Processes option in the application composer, and click Edit for the business process that you were creating.

   This opens the Oracle Business Process Composer in a separate Window.
This process is based on seeded template `ExtnBusinessProcess`, which uses the basic `CrmCommonSerialApprovalProcess` flow.

3. Click **Deploy Project** using the main menu.

4. In the **Deploy Project** popup 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>

5. Click **Deploy**.

A confirmation message appears when your project has been deployed successfully.


7. Verify whether the project has been deployed successfully.
a. On the overview page of the application composer, select the Business Processes option.

b. On the Business Processes page, verify that a green check-mark appears under the Deployed column for the project you created.

You have now deployed a project with the name Sales Lead Approval. You will now configure an object workflow to call this process.

**Step 4: Configure Object Workflow to Trigger Approval Process**

In this step, you will configure an object workflow with a desired trigger condition, and define the input parameters to pass to the Sales Lead Approval process when the object workflow is triggered.

To configure an object workflow:

1. From the Overview page of the application composer, select Object Workflows.
2. Select Create from the Actions menu.
3. On the Create Object Workflows Page, select the Sales Lead object.
4. In the Name field, enter Deal amount more than USD 10000.
5. Select When a record is created event point.
   
      Click on the expression builder icon.
      In the expression builder, enter the following event condition:

   \[ \text{if(DealAmount}>10000 \text{ and } \text{CurrencyCode}=='\text{USD}') \text{ return true;} \]
6. Click OK.
7. In the Actions region, select Business Process Flow and configure the event action as follows:
   a. On the Create Action: Business Process Flow page, enter Sales Lead Approval in the Name field.
   b. In the Project Name field, search and select Sales Lead Approval project that you deployed.

   This figure shows how you search and select a deployed business process. You can also create or edit a business process using this popup.

   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.

You must mark these values as Literal so as to pass the actual name.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>approvers</td>
<td>'SALES_MANAGER'</td>
</tr>
<tr>
<td>heldEntityStatusField</td>
<td>'LargeDealApproval_c'</td>
</tr>
</tbody>
</table>

**Note**

You can map only approvers and heldEntityStatusField values.

8. After inserting the input parameters, click Save to save the business process event action.

9. Click Save to save the object workflow.

You have now configured the object workflow to trigger the approval flow.

**Step 5: Trigger and Test the Approval Process**

In this step, you will first trigger the workflow and test whether the approval process runs fine.

1. Sign in to Oracle Sales Cloud using SALES_REPRESENTATIVE role.
2. Click the Navigator menu at the top.
3. Click Lead Qualification under Sales.
4. Click Create Lead.
5. Enter the Name as New Deal.
6. Create a lead with a Deal Size of 120000 USD.
7. Click Save.
   
   This action triggers the object workflow, because the deal size you have just entered is greater than 10000 USD.
   
   To verify whether the workflow has been triggered, you may check the status in the approval field.
8. Edit New Deal lead.
   
   Note the status in the Large Deal Approval field, which has been set to APPROVALPENDING by default.
   
   You are currently signed in with a sales representative role. You must first sign out and then sign in as a sales manager to approve the request.
10. Sign in to Oracle Sales Cloud using SALES_MANAGER role.

   A new task or request appears on the Home page under the Worklist region.

11. Approve the request.

   At this point, you are 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.


13. Sign in to Oracle Sales Cloud with a SALES_REPRESENTATIVE role.


15. Edit New Deal.

16. Verify that the status in the Large Deal Approval has been set to APPROVED.

   You have now verified that the approval process was successfully triggered via object workflows.

FAQs for Creating Object Workflows

Why are some e-mail templates invalid?

E-mail templates can become invalid if the corresponding template file of the e-mail template cannot be found in the file repository. This can happen if the template file was migrated elsewhere, or deleted accidentally. Contact your application administrator to attempt to recover the template file by reviewing your file repository setup. If the file cannot be recovered, then you can delete the e-mail template and create a new one.

Can I deploy multiple approval flows for the same object, but with different conditions?

Yes. You can create an object workflow with a combination of trigger conditions, for example, when an Opportunity is created with a specific value and when a specific discount is given. You can then create a business process flow with the desired approvals.

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.
Creating Custom Subject Areas: Overview

Read this chapter to learn about how you create and configure custom subject areas and then use them to build reports using Oracle Business Intelligence (BI) composer.

In this chapter, you will learn about:

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

Custom Subject Areas: Explained

A report subject area is a set of entities, attributes, and measures that represent information about the areas of an organization’s business. You select a report subject area from within the Oracle Business Intelligence (BI) Composer when building reports. Such reports display only those records that meet the criteria that is defined as part of the report creation flow. To build reports, use either the predefined report subject areas that are delivered with Oracle Sales Cloud applications, or create a report subject area using a wizard. The report subject areas that you create using the wizard are called custom subject areas, which is the subject of this topic.

Before you begin to create your own custom subject areas, review these topics:

• 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 area. These concepts are discussed in their related topics in details.

- **Objects**: Object within Oracle Sales Cloud represents an entity; it can be, for example, a sales account or an opportunity.
  - Standard object: An object that is delivered when the Oracle Sales Cloud is first used is called standard object, for example, opportunity.
  - 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**: 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

Oracle Sales Cloud is delivered with several entities, called standard objects, which store entity-specific information such as Marketing Campaign, Opportunities, and more.

Oracle Sales Cloud also allows the creation of objects to store specific information that is not delivered with the application. For example, one may want to track trouble tickets and would then create a custom object called Trouble Ticket.

In Application Composer, when you select the desired Application, the standard and custom objects are shown in two groups on an object's navigation tree in the left pane.

Standard and Custom Objects

Predefined configuration delivered with Oracle Sales Cloud is considered standard; therefore, an object that is delivered when the Oracle Sales Cloud is first used is called a standard object, such as Opportunity.

Any configuration that you perform and add to Oracle Sales Cloud is considered custom. If a new object or subject area is added during the deployment of Oracle Sales Cloud, it is then custom or called a custom object, custom field or custom attribute, or custom subject area.

When planning the design of your custom subject area, use Application composer to identify standard and custom objects or fields within Oracle Sales Cloud.
Primary Object

A primary object is any reportable top-level object that a custom subject area is based on. The primary object is the focus of the created report using the custom subject area.

This figure shows the Opportunity object as the primary object for a custom subject area that is created to report on opportunities and winning partner.

This figure shows the Edit Dynamic Choice List page for the custom Winning Partner dynamic choice list field.

Note

After you save or submit a custom subject area, you cannot change its primary object; however, you can create a custom subject area with a different primary object.

The list of available primary objects includes all reportable objects. Reportable objects are either top-level, custom objects, or standard objects that have been configured to be reportable by the owning Oracle Sales Cloud application.

Note

The common object Notes and Tasks are not reportable. They cannot be included in a custom subject area.

Child Objects

A child object is an object that has one-to-many relationship with a parent object and can be a parent object of another child object. 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 grandchild objects.

Note

You can add only one child to each parent, up to four levels.
Once you publish a custom subject area, you cannot add new child objects; however, you can create a (separate or another) custom subject area with a different primary object if desired.

**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. It is possible to add one or more related objects to a custom subject area. For example, Opportunity being the primary object, the administrator user could select the Opportunity object and then click **Select Fields** to add Opportunity related objects.

When the Select Fields screen is displayed the administrator user can add or remove Opportunity fields and also add or remove related objects and its fields from the custom subject area.
The same process is followed for any other child object added to the custom subject area. After you publish a custom subject area, you cannot remove related objects; however, you can create a new custom subject area with different related objects.

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

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. Oracle Sales Cloud is delivered with predefined fields, but also allows you to create custom fields to store your own custom information.

The Application Composer displays custom and standard fields in separate tabs on the Fields page.

**Reportable Field Types**

The Application Composer supports the following field types for reporting in any custom subject area:

- Text
- Number
- Date
- Percentage
- Date time
- Currency
- Check box
- Fixed choice lists
- Dynamic choice lists

**Custom Attributes on Standard Objects**

For key objects in Oracle Sales Cloud applications, predefined extension dimensions exist which include custom or extended attributes. This functionality enables users to create reports on extensions that are made to standard objects. Extension dimension attributes are not available for reporting until custom attributes have been specified. When expanding the Opportunity extension dimension, only the Opportunity type attribute is displayed.

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 will be displayed as either 0 or 1 on your custom reports.
• Number
• Currency
• Date
• String
• Percentage
• Phone
• Date time

To create reports using extension dimension attributes:

1. Navigate to the Application Composer.

2. Create custom fields for standard objects, and ensure that the custom fields are exposed on the user interface. This enables users to enter data for these new attributes.

3. Navigate to Oracle Business Intelligence Composer through Reports and Analytics option in the Navigator Menu.

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 select report columns, you can see that custom fields automatically appear under the extension dimension folder.

This figure shows the extensible Opportunity object with a custom Date field available for reporting automatically after its sandbox is published.
The custom Date field is available in the Opportunity Extension folder in the predefined or standard Opportunity subject areas.

Measures in Custom Subject Areas: Explained

Measures are a set of functions that you can apply on the date, numeric, or currency type fields of the selected primary or child objects while defining a custom subject area. Measures enable you to apply aggregation rules on the report data for a customized view using Oracle Business Intelligence (BI) Composer.

Measures available to a particular type of field may differ depending on the field type. Once you define the measures for the desired fields and publish a custom subject area, you can view and select these fields along with the applied measures when defining the report data in the Oracle Business Intelligence Composer.

Here are some measures you can apply to numeric, currency, or date type fields.

- For numeric and currency fields, a measure can be:
  - All
  - Sum
  - Average
  - Count
  - Count Distinct
  - Maximum
  - Minimum
  - First
  - Last
  - Median
  - Standard Deviation
  - Standard Deviation Population

- 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.
Implicit Fact Column in Custom Subject Areas: Explained

The 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.

In the following logical business intelligence representation example, there are several paths to be chosen when reporting only on attributes from the included dimensions. The implicit fact column tells the desired path that will represent the data and join all the three entities correctly.

![Logical Business Intelligence Representation Example](image)

**Note**

Only one implicit fact column can be selected per custom subject area.

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](image)

To use this hierarchy, create a report that shows total opportunity sales amount for each year. Drill down from year to show sales per quarter, sales per month, sales per week and the sales per day, and then drill back up to sales per year. The date hierarchy will aggregate the measures by desired hierarchy or leveling period.

To configure date leveling when defining a custom subject area, use the **Configure Date Leveling** step of the custom subject area wizard to either allow or disallow leveling. You may have to expand the field list in the **Date** field to select or clear the Date Leveling check box, as applicable.
Creating and Editing Custom Subject Areas: Explained

Using the available wizard, you create a custom subject area by selecting a primary object, related objects, and specific fields. When you later build a report within the Business Intelligence (BI) Composer, the custom subject area that you choose as the basis for the report controls how the data is displayed on the report.

Creating Custom Subject Areas

To access the custom subject area wizard in Oracle Sales Cloud applications:

1. Navigate to Application Composer from the Navigator menu, under Tools > Customization.
2. On the main Overview page, select an application from the Application choice list.
3. Select the Custom Subject Areas link in the Common Setup pane, or in the local area of the main Overview page.
4. Click the new icon to create a custom subject area.
   The custom subject area wizard is displayed.
5. Enter the following values for the custom subject area:
   • Label: Opportunity Sales Account
   • Description: Opportunity and sales account information.
   • Primary Object: Opportunity
6. Click Next.

   **Note**

   Subject areas usually have names or labels that correspond to the type of information that they contain, such as service requests and orders. Display labels have the Custom: prefix added automatically.

7. Select the objects that are part of the custom subject area and its label.
8. Click Next.
   When you select related objects for your custom subject area, consider the following points:
   • For a one-to-many primary-child relationship, you can add only one child for up to four levels like parent-child-grandchild-grandchild.
   • For a many-to-one primary-related relationship, you can add as many multiple related objects as you want.
   The primary object is added and displayed in this step automatically.
9. Choose the primary object’s Display Label as Opportunity Sales Account.
If the primary object has child objects, then the + icon is available.

10. Click the + icon.

The Add Child Object screen is displayed. If the added child object has children objects then the + icon will be available again. The + icon is available up to four levels of child objects.

All fields from the added objects in the previous steps are automatically added as part of the custom subject area.

To remove undesired fields perform the following steps:

1. Select the opportunity object in the **Fields From** drop down list.

2. Click **Select Fields** to display the Select Fields screen.

3. Select and click on the < to remove the selected field.

4. Using the Select Fields screen, you can also add any available related object from the selected parent object.

   To add a related object:

   a. On the Select Fields dialog, select **Related object** and select the desired related object from the list.

   b. Select the fields in the **Available Fields** list to be added to the custom subject area and click > button.
c. After all related objects and fields are added, click OK.

For each field you added, you can change its Display Label as desired. This label is used when creating a field as an attribute in a folder within the custom subject area.

5. Repeat steps 1 to 4 for removing undesired fields or adding related objects from each object that is added to the custom subject area.

6. Select the desired measures to be generated for number, date, or currency fields type from all available objects added to the custom subject area.

Note

Fields in the Fields list are automatically added to their object folder in the custom subject area. If measures are defined for the fields, those fields are also created as measures in the Fact folder in the custom subject area.

7. Select the Implicit Fact column as desired.

8. Click Next.

9. If required, select the Date columns for date leveling functionality in the custom subject area.

10. Select the desired 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.

Note
Note: The security definition here only control who can access the custom subject area definition to create reports. It does not control data visibility which is automatically controlled based on the user running the reports.

11. Review the custom subject area configuration for all added objects, attributes, and measures, and if satisfied, click Submit. If changes are required click on Back to navigate back to the desired screen and perform the desired changes.

12. When submitted, the custom subject area configuration is prepared for publishing.

Note
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 for publishing a custom subject area before you can select it from within Oracle BI Composer. Once you save or submit a custom subject area, you cannot modify its primary object. Custom subject areas are prefixed with Custom:

13. To access the published custom subject area, use the Navigator menu to go to Reports and Analytics under Tools category.
14. In the Contents page, select Create and then select Analysis:
15. Select the published custom subject area and start creating your report.

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 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. It is possible to edit a custom subject area in any status.

To edit a custom subject area, perform the following:

1. Select **Application Composer** from the **Navigator** menu, under the Tools > Customization category.

2. On the main Overview page, select an application from the **Application** choice list.

3. Select the **Custom Subject Areas** link in the Common Setup pane, or in the local area of the main Overview page.

4. Select the custom subject area that you want to edit and click the **Edit** icon.

5. Make the desired changes. When you are done with your changes, click the **Submit** button to republish the custom subject area.

When editing a 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 application. 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.

**Note**
You can inactivate only those custom subject areas that are published and have OK status, and can activate only previously inactivated custom subject areas.

To change the Active status, select or clear the **Active** check box in the Define Custom Subject Area step of the wizard while editing a custom subject area. This graphic shows the Edit Custom Subject Area page with **Active** check box.

**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 input 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 our 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, can be 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 input into the segment).

Custom Objects Related to B2C Customers (Consumers)

The set up 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 should 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 set up 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 should 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 role names, which determines whether a role name can access a custom subject area. You can also add role names from a predefined list and assign or revoke permissions. This topic covers how you can add or delete role names, or grant or revoke access rights from those role names.

**Managing Role Names and Access Rights**

While defining a custom subject area using the wizard, 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

When you submit the data that you configured for a custom subject area, it becomes available in the Oracle Business Intelligence (BI) Composer for building and viewing reports. The submission processes begin when you submit a custom subject area in the last step of the custom subject area configuration wizard. This topic covers what happens when you submit a custom subject area, and what the submission statuses indicate.

Understanding the Publishing Process
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.

Note
You must refresh the status to know whether the custom subject area is submitted successfully. You may have to refresh the status multiple times, as the Oracle ADF and RPD artifacts creation may require longer time.

After the status of a custom subject area changes to **OK**, you can use Oracle BI Composer to create reports using the objects, attributes, and measures that you configured for the subject area.

A custom subject area can have one of the following submission statuses:

- **Pending**: Save and close a custom subject area at any point before publishing is completed. You can return to the pending custom subject area to complete the configuration process. This status may also indicate a failure in the background processes while creating Oracle ADF and RPD artifacts for a custom subject area.

- **In Process**: 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.

- **OK**: A custom subject area is published successfully and is available to the Oracle BI Composer for report configuration and analyses.

Creating a Custom Object and Associating it With a Custom Dynamic Choice List Field: Worked Example

This example illustrates the creation of a custom object called Winning Partner and associating it with the Opportunity object as a custom Dynamic Choice List field.
This example covers these tasks:

- Winning Partner Custom Object configuration
- Winning Partner Dynamic Choice List
- Opportunity Winning Partner Custom Subject Area
- Viewing the Published Custom Subject Area

This figure illustrates the proposed transactional model.

Winning Partner Custom Object Configuration

In this section, you are creating a custom object called "Winning Partner". Before making any customization in Oracle Sales Cloud applications, you should have a sandbox session active.

For more information on sandboxes and how to use them, refer Guidelines for customizing Oracle Sales Cloud using Application Composer and Sandboxes (Doc ID 1484889.1) on My Oracle Support (MOS) at https://support.oracle.com/.

1. When your Sandbox session is active, click the Navigator menu.
2. Click on Application Composer menu item.
   
   You are on the main page of Application Composer. You are adding an attribute to a standard object that belongs to the Sales application.

3. In the left pane, click the Application list.
4. Click the Sales list item.
5. Click the Create button in the Objects pane or click the Custom Object node and then the Create a New Object icon in the custom object table.
6. Create the custom object configuration as follows:

The following standard fields are created automatically for new custom objects:
7. Create a new Workarea by clicking in the Pages node under the Winning Partner custom object in the Objects pane.

8. Configure the Workarea as follows:
   a. Enter Winning Partner in the Menu Item Display Label.
   b. Define the Summary Table as shown:
   c. Define the Creation Page as shown:
   d. Define the Details Page, just the Default Summary
9. Configure a picker for the Winning Partner custom object as follows:

**Winning Partner Dynamic Choice List**

In this section, you are creating a custom dynamic choice list under the Opportunity object based on the created Winning Partner custom object.

To do this, navigate to Application Composer and perform the following:

1. Expand the **Standard Objects** node in the Objects pane.
2. Select and expand the **Opportunity Object** node.
3. Select the **Fields** node, and in the Custom tab, click the create a custom field icon.
4. Configure the custom **Dynamic Choice List** as follows:

   5. Click **Next**.
6. Click Submit.

   At this point you have associated your custom object with the standard Opportunity object as a related object.

7. Test your customizations to ensure they work properly.

8. Publish your sandbox.

   At this point, everything is ready for the custom subject area creation.

**Opportunity Winning Partner Custom Subject Area**

In this section, you are creating a custom subject area to enable reporting on Opportunities and Winning Partners.

To do this, navigate to Application Composer and perform the following:

1. In the Application list in the left pane, select Sales application. This is where you created your Winning Partner customizations.

2. Navigate to the Custom Subject Area page

3. Click on Create icon

4. Enter "Opportunity Winning Partners" in the Label field.

5. Select Opportunity as the Primary Object.

6. Click Next. The Opportunity becomes the display label for the Opportunity object.

7. Click Select Fields to open the Select Fields screen.

8. Select Related object option.

9. Select WinningPartner_c custom object.

10. Add the Id and Partner Name fields to the Selected Fields list.

11. Click OK.

12. Select the desired measures for the Opportunity and Winning Partner related object.
Note

By default, all supported types fields are automatically included; however, if you want to remove some of the fields, you can remove them by clicking on the Select Fields button.

13. Click Next.


15. Click Next.

16. Select the Date Leveling check box for the Date field.

17. Accept the default Role Name security settings.

18. Click Next.

19. Review the custom subject area configuration.
Create Custom Subject Areas: Review and Submit

20. Click Submit.

Wait until you are navigated back to the custom subject area table.

21. Click Refresh icon after few minutes until you see the status of the created custom subject area as OK.

Creating and Viewing a Report for the Custom Subject Area

In this section, you first create a report using the published custom subject and then view it.

1. Click the Navigator menu item.
2. Under the Tools category, click Reports and Analytics.
3. Click Create, and then click Analysis.
4. Under Select Subject Area, locate and select the published Custom: Opportunity Winning Partners custom subject area.
5. Review the custom subject area: Expand each folder and review its attributes and measures.
6. Follow the steps in the report creation wizard to create a report:
   a. Add the attributes and measures to the selected columns list.
b. Click **Next**.

c. Select Table and Graph to be included in the report.

d. Click **Next**.

e. Define the Table layout options.

f. Click **Next**.

g. Define how data will be sorted or filtered in the report.
h. Click Next.

i. Define any desired column format.

j. Click Next.

k. Enter the Analysis Name as Opportunity Winning Partners.

l. Click Submit and wait until the Confirmation screen is displayed.

7. Run the created report by navigating to the folder where you saved the report.


9. Click View to run the report.

Creating Custom Fields for a Standard Object: Explained

This example illustrates how you can create custom fields for the Sales Account standard object and how you use them for creating reports.

You will create several custom fields under the Sales Account standard object. Before making any customizations in Oracle Sales Cloud applications, you should have a sandbox session active.

You will create the following types of fields under the Sales Account standard object:

- Text
- Number
- Choice List (Fixed)
- Date
- Percentage

For more information on sandboxes and how to use them, see Guidelines for customizing Oracle Sales Cloud using Application Composer and
Adding Custom Fields Using Application Composer

To add custom fields:

1. When your sandbox session is active, click the Navigator menu.
2. Click Application Composer.
   You are on the main page of the Application Composer. You are adding an attribute to a standard object that belongs to the Customer Center application.
3. In the left pane, click the Application list.
4. Click the Customer Center list item.
5. In the Objects pane, expand the Standard Objects node.
6. Expand the Sales Account node and select the Fields node.
   The Custom tab is displayed on the Fields screen
7. Click the create a custom field icon to start adding the following fields:
   • Add a Text field.
   • Add a Number field.
• Add a Fixed Choice List field

• Create the Lookup Values for the Fixed Choice List.
• Create a Date field.

• Create a Percentage field.

You will now expose these new custom fields in the Sales Account Workarea.
To expose the custom fields in the workarea:

1. In the Objects pane, click in the Pages node to display the Page configuration screen.
2. Click the Edit Creation Page link and add the created custom fields.
3. Click Save and Close to return to the Page configuration screen.
4. Click in the Edit Summary Form link and add the created custom fields.
5. Click Save and Close to return to the Page configuration screen
6. Click the Navigator menu item.
7. Under the Sales category, click the Customers link.
8. Click the Manage Customers link in the Tasks regional pane.
9. Edit or Create a Customer using the Sales Account screen. Ensure your custom fields are exposed correctly in the user interface.
10. Enter sample values for each custom field.
11. Click Save and Close to return to the Sales Account Workarea.
12. Test your customizations to ensure they work properly.
13. Publish your sandbox.

At this point, everything is ready for the report creation.

**Reporting on the Created Custom Fields**

In this section, you create a report using the custom fields that you have created for the Sales Account standard object.

**Note**

After the sandbox is published, you can access the custom fields for Sales Account object to create a report.

To create a report:

1. Click the Navigator menu.
2. Navigate to Reports and Analytics under Tools category.
3. In the Contents page, select Create and then select Analysis.
4. Select a Subject Area, in our example, select Sales - CRM Sales Activity. The Create Report screen is displayed.
5. Expand the Sales - CRM Sales Activity subject area and navigate to the Sales Account Extension folder. This folder contains the created custom fields that you will use for reporting.

6. Start adding fields from the subject area as desired. Make sure to include the custom fields from the Sales Account Extension folder in your report.

7. Click Next.

8. In the Create Analysis: Select Views step of the wizard, enter the Title as Sales Account Extension and select the Table format.

9. Click Next.

10. In the Create Analysis: Edit Table step, review the Table Layout.

11. Click Next.

12. In the Create Analysis: Sort and Filter step, define how to want to sort the columns or apply filters.

13. Click Next until the Save screen is displayed.

14. In the Create Analysis: Save step, enter the Analysis Name as Sales Account Extension.

15. In the Save In area, select the folder where you want to save the analysis.

16. Click Submit.

17. To access your created report:
   a. Locate your report under Folders in the Contents regional pane.
   b. Click Sales Account Extension, which is the report you created.
   c. Click the View link.

The report is run and displayed.

**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 will add a custom, fixed-choice field called Strategic Value to the opportunity object.

1. Click the Navigator menu item.
2. Click the more... >> link.
3. Click the Application Composer link.
   You are on the main page of Application Composer. You are adding an attribute to a standard object that belongs to the Sales application.
4. In the left pane, click the Application list.
5. Click the Sales list item.
6. In the Objects region on the left, locate the opportunity object.
7. Click the Expand button of the Standard Objects tree.
8. Under the Standard Objects tree, click the Expand button of the Opportunity tree item.
9. Click the Fields link under the Opportunity tree item.
   You are on the Fields page.
10. In the Custom tab, click the Create button.
11. In the Select Field Type popup, click the Choice List (Fixed) option.
12. Click OK.
   You are on the Create Fixed Choice List page.
13. In the Appearance region, enter "Strategic Value" in the Display Label field.
14. Enter "Select whether the deal is strategic" in the Help Text field.
15. In the List of Values region, click the Search and Select Lookup Type button.
16. In the Search and Select: Lookup Type popup, search for existing Yes/No fields.
   In the Search region, enter "Yes" in the Meaning field.
17. Click Search.
18. In the search results, click the Yes / No Lookup Type cell.
19. Click OK.
20. You are back to the Create Fixed Choice List page. You will now set a default value for a new deal. In the Default Value region, click the Fixed Value list.
21. Click the No list item.
22. 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 Runtime

In this example, you will 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 more... >> link.
3. Click the Reports and Analytics link.
   You are on the Reports and Analytics page. Use the toolbar in the left pane to navigate to Oracle Business Intelligence Answers.

4. 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.

5. Double-click the Shared Folder tree.

6. In the Shared Folders tree, double-click the Sales tree item.

7. In the Sales tree item, double-click the Subject Area Contents tree item.

8. In the Subject Area Contents tree item, select Sales - CRM Pipeline.

9. 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. Locate the Opportunity Extension tree item to see the attribute you added.

10. Double-click the Opportunity Extension tree item.

11. In the Opportunity Extension tree item, locate the Strategic Value field.
   You have completed the activity of verifying that a custom attribute is reportable at run time.

Creating a Custom Subject Area
In this example, you will create a custom subject area using the opportunity object.

1. Click the Navigator menu at the top.

2. Click the more... >> link.

3. Click the Application Composer link.
   You are on the main page of Application Composer. You are creating a custom subject area using the opportunity object, which belongs to the Sales application; therefore, you will select the Sales application for your activity.

4. In the upper-left region of the page, click the Application list.

5. Click the Sales list item.

6. 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.

7. In the Search Results region, click the Create button.

8. Enter a name for the custom subject area that you are creating. Enter "Opportunity_Contact" in the Label field.

9. In the Primary Object region, click the Primary Object list.

10. Click the Opportunity list item.
11. In the upper-right region of the page, click the Next button.
12. In the upper-right region of the page, click the Add Child Object button.
13. In the Add Child Object popup, click the Child Object list.
14. Click the OpportunityContact list item.
15. Click the OK button.
16. In the upper-right region of the page, click the Next button.
   You will now define measures for date and numeric fields for the Opportunity primary object. You do not need measures applied to all Date and Numeric fields, which is currently selected by default; therefore, first remove the default selection, and then add measures to the fields you require for your custom subject area.
17. Ensure that the selected value in the Fields From list is Opportunity. Click the Actions menu.
18. Click the Deselect All Dates as Measures menu item. This action will deselect all Measures selected by default for the Date fields.
19. Click the Actions menu.
20. Click the Deselect All Numerics as Measures menu item. This action will deselect all Measures selected by default for Numeric fields.
21. You will now specify the fields on which you want to apply measures. Click the Measure option for the Revenue field.
   In this activity, you will not be applying measures for fields in the OpportunityContact child object.
22. In the upper-right region of the page, click the Next button.
23. You will now select fields to apply date leveling. In the Date Field Leveling table, click the Expand button of the Opportunity object.
24. Select the Allow Leveling option for the RevenueEffective date field.
25. In the upper-right region of the page, click the Next button.
26. Leave the default role access of Read for Everyone. In the upper-right region of the page, click the Next button.
27. Review your custom subject area. In the upper-right region of the page, click the Save button.
28. In the upper-right region of the page, click the Submit button.
29. A Confirmation message appears. Click the OK button.
   You have successfully created a custom subject area.

Creating a Report Using Subject Area

In this example, you will 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 more... >> link.
3. Click the Reports and Analytics link under Tools.
4. You are on the Reports and Analytics page. In the left pane, click the Create button.

From the Select Subject Area popup 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.

5. In the Select Subject Area popup, locate and click the Sales - CRM Pipeline link.

6. 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.

7. 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.

8. In the upper-right region of the page, click the Next button.

   You are in the Select Views step of the wizard.

9. Enter "Opportunity Count By Sales Stage" in the Title field.

10. Click the Graph list.

11. Click the Bar (recommended) list item.

12. Click the Preview option on the right of the Title field.

13. In the upper-right region of the page, click the Next button.

14. You are in the Edit Graph step of the wizard. From the Group By box, exclude the fields you do not want in your report. Notice how the Preview changes.

15. In the Group By box under the Graph Layout area, click the Opportunity Name option.

16. Click the Move To list adjacent to the Group By heading.

17. Click the Excluded list item.

18. In the Group by box, click the Employee Name option.

19. Click the Move To list adjacent to the Group By heading.

20. Click the Excluded list item.

21. In the upper-right region of the page, click the Next button.

22. You are in the Sort and Filter step of the wizard. In this step, you will select the following filters for your report:
   - Sales Stages to display in your report
   - The Customer for which these Sales Stages should be displayed.

23. In the Filter region, click the Add Filter list.
24. Click the Sales Stage Name list item.
25. Click the Operator list of the Sales Stage Name filter.
26. Click the is in list item.
27. Click the Value list of the Sales Stage Name filter.
   Select these options:
   • 02 - Negotiation
   • 07 - Closed
   • Short List
   • Solution Presentation
28. In the Filter region, click the Add Filter list to add another filter.
29. Click the More Columns... list item.
30. You are on the Select Column popup. In the popup, click the Expand button of the Customer tree.
31. In the Customer tree, click the Customer Name tree item.
32. Click the OK button.
33. You are back to the Sort and Filter step. In the Filter region, click the Operator list of the Customer Name filter.
34. Click the is in list item.
35. You will now search for a customer name. Click the Search button adjacent to the Value list of the Customer Name filter.
36. You are on the Select Values popup. Ensure that the Name field contains begins with value. Enter the desired information into the Search Criteria field. Enter "Pinnacle".
37. Click the Search button.
38. Click the Pinnacle Technologies item in the Available box.
39. Click the Move selected items to other list button in the middle.
40. Click the OK button.
41. In the upper-right region of the page, click the Next button.
42. You are in the Save step of the wizard. Enter the desired information into the Report Name field. Enter "Opportunity Count By Sales Stage".
43. In this activity, you will save your report in only My Folders. In the Save In area, click the My Folders tree.
44. In the upper-right region of the page, click the Submit button.
45. A Confirmation message appears. Click the OK button.
46. You may now view the report you just created. Click the Expand button of the My Folders tree.
47. Locate the report you just created.
You have successfully created a report in BI Composer using CRM - Sales Pipeline subject area.

**Extensibility Analytics: Worked Example**

You can use the Applications Composer to extend the following analytics capabilities:

- **Specify the aggregate formula to apply a measure so that the subject area is better defined and includes only the measures that you are interested in analyzing.** Aggregated measures include CURRENCY data type measures, DATE data type measures, and NUMBER data type measures. You can only specify aggregate formulas to apply to a measure when creating a custom subject area. It is not possible to change predefined measures or edit an already published custom subject area measure.

- **Use implicit facts to ensure the data representation displayed in the report is presented correctly when using only dimensional data in the report.** An implicit fact is a fact that has been specified for a subject area to be used as part of any dimensional browsing query. Oracle Sales Cloud applications then use this fact as part of the query during dimensional browsing. Typically, the most frequently used fact or the functionally most relevant fact is specified as the implicit fact. For example, a revenue fact is stamped as the implicit fact in the Pipeline subject area, and a sales account fact is marked as an implicit fact in the Customers subject area.

- **Make ID Fields available for reporting by selecting ID fields as part of the Custom Subject Area.** You can then run reports on the ID fields.

The following example scenario demonstrates how to configure these enhancements.

**Extend Analytics**

1. Navigate to **Edit Custom Subject Area: Fields**.

2. Specify the aggregate formula to apply so that the subject area includes only the measures that you want to analyze. You can only define measures for number, date, or currency field types that have not being defined as a measure when editing a custom subject area. In the **Select 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.

3. After you specify all measures and related aggregations in the Fields step of the wizard, in the guided step called Configure Implicit Fact, specify one of the measures as the implicit fact. You can specify only one measure as the implicit fact for a subject area.

4. Select the ID fields **Row ID**, **Record Type**, and **Record Number** as part of the Custom Subject Area. For instance, Opportunity Row ID and Opportunity Number are attributes that you can report on.
Navigate to the specific opportunity or partner page and use the ID as a parameter to run reports on the ID fields.

**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.

**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 once 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.
Page Composer: Customizing Oracle Sales Cloud

Customizing Oracle Sales Cloud Using Page Composer: Overview

Read this chapter to learn about how to use Page Composer, also known as Oracle Composer in previous releases, to customize the look and feel of Oracle Sales Cloud user interface pages.

In this chapter, you will learn about:

- 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

Important
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.

Before You Customize Existing Pages: Points to Consider

Before you start your customizations, consider these recommended prerequisite tasks:

1. Familiarize yourself with the Oracle Fusion application architecture that enables customization.
2. Become knowledgeable of the typical workflows for working with run time customizations.
3. Verify that the page is customizable in your Oracle Fusion application.
4. Confirm that your privileges are sufficient for customizing the page.
5. Set up a sandbox for working on your customizations.
6. Determine the appropriate layer for the customization.

Customizing Oracle Sales Cloud Pages Using Page Composer: Explained

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 it 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 runtime customization using Page Composer. These pages and regions are delivered already enabled for page editing. Administration and setup pages are not Page Composer-enabled.

**Note**

Page Composer does not support the customization of custom object pages and custom fields. You can customize extended pages and fields only with the Application Composer.

The following figure shows how Page Composer affects only the UI layer.

**Page Composer Customization Modes**

Page Composer supports two editing modes: Design View and Source View. In Design View mode, you can edit by direct manipulation of the target
components and their properties. In the Source View mode, you can edit from
the programmatic code of the target component. In Oracle Sales Cloud, Page
Composer is launched in Design View mode by default. 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.

The Page Composer Design View mode has two selection submodes:

- Design mode is launched by default when opening a page with Page
  Composer. You can also select this mode by clicking the Design tab on the
  top left corner of a page in Page Composer mode.
- Direct Selection mode is activated when you click the Select tab that is
  next to the Design tab.

The following figure shows a Page Composer-enabled dashboard in Design View
mode and Design Selection mode.

The Direct Selection mode is mainly used to select and edit UI components,
such as form fields and table columns. In Direct Selection mode, the enabled UI
components become apparent when you move the cursor over the component.
Enabled UI components get in focus on mouseover and are highlighted by a light
blue border for editing. When you click the highlighted component, the border
changes to dark blue and an Edit window appears.

The window allows you to select whether to edit the selected component or the
parent component. Examples of parent components include tables that hold
columns and forms that hold fields.

**Note**

In Oracle Sales Cloud, the Direct Selection mode is available when you
customize pages through the Administration link, but not when you personalize
them from the Personalization link.

**Accessing Page Composer**

Access Page Composer from the Administration global menu. Access to the
Administration menu is controlled through the Administration Link View Duty.
The following table lists examples of duty and job roles that must be assigned to
you so that you can access the Page Composer editor.
<table>
<thead>
<tr>
<th>Application</th>
<th>Duty Role</th>
<th>Job Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDH</td>
<td>Master Data Management Application Administrator</td>
<td>Master Data Management Application Administrator</td>
</tr>
<tr>
<td>Sales Catalog</td>
<td>Sales Catalog Administrator Duty</td>
<td>Sales Catalog Administrator</td>
</tr>
<tr>
<td>Sales</td>
<td>Sales Administrator Duty</td>
<td>Sales Administrator</td>
</tr>
<tr>
<td>PRM</td>
<td>Channel Administrator Duty</td>
<td>Channel Administrator</td>
</tr>
<tr>
<td>PRM</td>
<td>Channel Partner Portal Administrator Duty</td>
<td>Channel Partner Portal Administrator</td>
</tr>
</tbody>
</table>

For additional details on these security privileges, contact your security administrator.

Page Composer also supports sandboxes. The sandbox provides temporary storage for your customization changes until you are ready to commit them to the back end. Before working with sandboxes, review the guidelines and recommendations on using sandboxes, available in related help topics.

Use the following steps to open Page Composer:

1. Navigate to the Page Composer-enabled page.
2. Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.
3. Select the Administration global menu.
4. In the Customize <page name> Pages dialog box, select the MDS layer that you want to customize.
5. Click OK.
6. Click Customize <page name>.
7. Select Customize Work Areas for dashboard customization.
Page Composer opens.
8. Customize the page.
9. When you are done, click Close to leave the Page Composer editor.

Available Customization Options

In Oracle Sales Cloud, you can use the following customization options:

- Transactional, landing, and dashboard page customizations. Only administrators can do these customizations.
- Partner Portal UI shell and branding customizations. Only Partner Portal administrators can do these customizations.
- User-level page personalization for limited personalization tasks, such as hiding or adding content. Personalization changes are only visible to the user who is doing them.

Additionally, you can do Page Composer customizations in any of the following MDS customization layers:
• Site: Your customizations are visible to all site users.
• Job Role: Your customizations are visible to users who have the selected job role.
• External or Internal: Depending on your selection, your customizations are visible to either external or internal users. External users could be your partners or anonymous users. Internal users could be your employees.

When Page Composer opens, the Customize <page name> Pages dialog box opens, where you can select the MDS layer to customize. The layer that is selected in the Edit column is the layer that you want to edit. The layers that are selected in the Include column inherit any changes you make to the layer you edit.

By default, changes made at higher levels (Site) are propagated to lower level layers (External or Internal and Job Role), unless you uncheck these layers in the Include column.

The following figure shows selecting the MDS layer for customization.

---

**Note**
Oracle Sales Cloud Welcome dashboard does not support job role or External or Internal layer changes. All customizations to this dashboard are applied at the site level.

**Available Customization Tasks**
Design and Direct Selection modes support different sets of customization tasks (although in some instances a task can be performed in either mode). The Design mode is intended for overall UI composition customizations, such as adding content, changing the dashboard layout, and changing region properties. The Direct Selection mode is intended for customizations at the component rather than the region level, mainly for fields, forms, tables, and tree nodes customizations.

**Tip**
It might be necessary to toggle between Direct Selection and Design mode to navigate between pages when performing Direct Selection customizations, since you can only navigate in Design mode.

All Oracle Sales Cloud dashboard pages support the design mode of the Page Composer-enabled regions. External facing landing and transactional pages that are expected to need customizations also support the design mode for the following Page Composer-enabled regions:

• Partner Snapshot
• Edit Partner Profile
• Edit Partner Public Profile
• Edit Personal Profile
• Partner Landing
• Partner Registration Landing
• Partner Registration: Partner Information
In general, use Direct Selection mode to do the following customizations:

- Fields, such as input text fields, output fields, and list of values fields:
  - Show or hide component
  - Make read-only
  - Make required
  - Edit label

- UI Containers (forms, tables and trees):
  - Show or hide child components, for example, fields in the form
  - Reorder child components, for example, reorder fields in a form

**Note**

UI components can be protected from updates to preserve the product business logic, for example, a field is read-only or not based on a security privilege already defined in the application. If protected from updates, the UI components are dimmed and cannot be customized in Page Composer.

The following table lists the customization tasks available through Page Composer and the modes in which they are supported.

<table>
<thead>
<tr>
<th>Customization Task</th>
<th>Design View - Standard Mode</th>
<th>Design View - Direct Selection Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change local area layout. For example, change a two-column layout to three-column layout (eight layouts are available).</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add, rename, remove dashboard tabs (except for Home tab).</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Expand, collapse dashboard Regional pane (or move the splitter location).</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add and remove panel boxes to or from the dashboard local area.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add predefined content to dashboard panel boxes (Business Intelligence reports, CRM portlets, and common components such as Calendar).</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Edit dashboard panel box properties: show, hide box, reorder child regions, display options, and style.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Customization Task</td>
<td>Design View - Standard Mode</td>
<td>Design View - Direct Selection Mode</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>----------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Add and remove ADF Components to or from dashboards (for example, regions,</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>hyperlinks, images, text boxes, movable boxes, and Web pages) and edit their</td>
<td></td>
<td></td>
</tr>
<tr>
<td>properties.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customize saved searches (create and edit).</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hide or show field.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Change field label.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Make field required or not.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Make field read-only or updateable.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reorder fields in a form.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Reorder table columns.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hide or show table columns.</td>
<td>Yes (end users can optionally display columns at run time)</td>
<td>Yes (end users cannot display columns at run time)</td>
</tr>
<tr>
<td>Set table column width with the mouse.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Set table column width and minimum width in percent or pixels.</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable, disable column sorting.</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Customizing the Oracle Sales Cloud Welcome Dashboard**

Also known as the Oracle Sales Cloud Home page, the Welcome dashboard is the application suite’s default starting page. It is composed of a collection of tabs that are visible based on the roles assigned to users. The Welcome dashboard provides a collection of high-level data summaries meant for quick monitoring and navigation to key business objects. You can customize this page.

To customize the Welcome dashboard:

1. Navigate to the dashboard.
2. Click the Administration global menu.
3. Select **Customize Workarea Pages**....

The Welcome dashboard consists of a local area only. The supported customization tasks, which you do in Design mode, are:
• Change the local area layout.
• Add or remove panel boxes.
• Add seeded content to dashboard panel boxes.
• Edit dashboard panel box properties: show or hide a box, reorder child regions, change display options and style.
• Add, edit, and remove ADF components to or from dashboards, such as regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

Note
The Welcome dashboard does not support role-based or External or Internal interface customization. All customizations are site-wide.

Customizing Simplified Pages Using Page Composer: Points to Consider

On a simplified page, you can click your name and select Customize User Interface to customize the UI using Page Composer. When customizing a simplified page, consider the customization layer to choose, the types of customizations you can make, and labels for your saved changes.

Customization Layers

The customization layer that you select before making changes to the page determines the scope of users impacted by your customizations. If you are not presented with customization layers to choose from after you select Customize User Interface, then your changes are made to the site layer. For more information to understand customization layers, see the Oracle Fusion Applications Extensibility Guide for Business Analysts.

Types of Customizations

In simplified pages, customization using Page Composer is limited to what you can change with component properties. For example, you can show or hide fields or make a check box required, but you cannot add new components or change the layout of the page.

After you select a customization layer, if available, you can click:
• Design to navigate around and get to the components you want to customize. You cannot make any customizations in this mode.
• Select to select a component on the page and open its properties.

Each component has its own set of properties, which may include some of the properties in this table.

<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>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</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 Required</td>
<td>Whether an asterisk is displayed to indicate that the component is required</td>
</tr>
</tbody>
</table>

When you access component properties on a workstation page using Page Composer, more properties are available.

**Save and Label**

When you save, you can label your changes so that you can later revert to your saved customizations. Labels are stored with a prefix of `composer_`. For example, if you enter `myLabel`, then the label is `composer_myLabel`.

As needed later, you can click your name in the global area and select Manage Customizations. Click Promote for the desired component and select the label to revert to.

**Personalizing Dashboards, Transactional Pages, and Landing Pages Using Page Composer: Overview**

Users can customize their own dashboard pages. These customizations are visible only to the user who made them.

The following figure shows the Page Composer - Personalization menu.

Personalization tasks are available only in Design Edit mode, and they include:

- Change the local area layout. For example, change a two-column layout to a three-column layout. In all, eight layouts are available for selection.
- Add, rename, or remove subtabs.
Restriction
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 dashboard panel box properties: show or hide a box, reorder child regions, or change the display and style options.
- Add, remove, and edit dashboard ADF components: regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

Customizing User Interface Components with Direct Selection Support Using Page Composer: Overview

Use this topic to learn about the user interface components that support Direct Selection Mode functionality in Page Composer.

The Page Composer Design View mode has two selection submodes:
The Direct Selection mode is mainly used to select and edit UI components, such as form fields and table columns. In Direct Selection mode, the enabled UI components become apparent when you move the cursor over the component. Enabled UI components get in focus on mouseover and are highlighted by a light blue border for editing. When you click the highlighted component, the border changes to dark blue and an Edit window appears.

The following table lists:
- User interface components that support Direct Selection mode
- Customization properties of the components

<table>
<thead>
<tr>
<th>ADF Component Name</th>
<th>Type</th>
<th>Selectable</th>
<th>Rendered Attribute</th>
<th>Read-Only Attribute</th>
<th>Required Attribute</th>
<th>Label Attribute</th>
<th>Minimum Width</th>
<th>Maximum Width</th>
<th>Sortable</th>
<th>Header Text</th>
<th>Reorder Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>af: activeOutputText</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputComboboxListOfValues</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputDate</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputFile</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputListOfValues</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputNumberSlider</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputList</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputNumber</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>ADF Component Name</td>
<td>Type</td>
<td>Selectable</td>
<td>Rendered Attribute</td>
<td>Read-Only Attribute</td>
<td>Required Attribute</td>
<td>Label Attribute</td>
<td>Minimum Width</td>
<td>Maximum Width</td>
<td>Sortable</td>
<td>Header Text</td>
<td>Reorder Children</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
<td>------------</td>
<td>--------------------</td>
<td>--------------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------</td>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>af: inputNumber</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputRange</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: inputTextField</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>af: outputNumber</td>
<td>Field</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
<td>NA</td>
</tr>
<tr>
<td>af: panelLayout</td>
<td>Container</td>
<td>Yes</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
<td>NA</td>
</tr>
<tr>
<td>af: richTextField</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectBoolean</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectOne</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectOne</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectOne</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectOne</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af: selectMany</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Page layout defines the number, placement, and orientation of content regions on a page. You set the layout style when you create the page. Some layouts allow choosing a different layout even after you add content to the page; others do not support switching layouts after creation.

To change the layout of a page:
1. Access the page in Design view.
2. Click \textit{Change Layout}.
3. Select the new layout.

\section*{Page Component Properties: Explained}

All components have configurable properties that control, or express, their appearance and behavior. Many properties are common to all component types and some properties are unique to one component type. You access the properties of a component by opening the Component Properties dialog box. The dialog box organizes properties of similar function under tabs that name the category of the properties. Properties and tabs can vary from component to component.

\subsection*{Component Property Tabs}
This table describes the tabs that you might see in a component properties dialog box.

<table>
<thead>
<tr>
<th>ADF Component Name</th>
<th>Type</th>
<th>Selectable</th>
<th>Rendered Attribute</th>
<th>Read-Only Attribute</th>
<th>Required Attribute</th>
<th>Label Attribute</th>
<th>Minimum Width</th>
<th>Maximum Width</th>
<th>Sortable</th>
<th>Header Text</th>
<th>Reorder Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>af:selectOrderShuttle</td>
<td>Field</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af:ColorPanel</td>
<td>Container</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
<td>Yes</td>
<td>NA</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>NA</td>
</tr>
<tr>
<td>af:panelContainer</td>
<td>Container</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af:showDeleteTable</td>
<td>Container</td>
<td>Yes</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af:table</td>
<td>Container</td>
<td>Yes</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>fnd:application</td>
<td>Container</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>af:panelContainer</td>
<td>Container</td>
<td>No</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Yes</td>
</tr>
<tr>
<td>Tab</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>Settings that can control component aspects that are specific, or often unique to the component.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, on a page that contains a map, a component might have a parameter that provides a choice between units of measurement.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display Options</td>
<td>Settings that affect the chrome of a component, including:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Header, header text, and border</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Actions menu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Edit, Remove, Expand, Collapse and other icons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tooltips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, display options on image layout components specify the image source URL and its optional link target.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Child Components</td>
<td>The list of all the components contained within and under the control of the parent component including controls for ordering the child components.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Style</td>
<td>Settings that affect the look and feel of the component chrome or the component contents and that override corresponding values from a parent object, such as a component, page, and application providing an opportunity to fine-tune appearance.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Content Style</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Some style properties might be disabled at the component level if other page or application elements (such as the skin) do not support modification to the property.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, font, color, and dimension.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Events</td>
<td>Events and event handlers for all the components on the current page that the current component can consume.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For example, an event could be selecting a check box within the current context. The code that runs and drives the result of a event, such as making another component visible, is an event handler.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Working With Components in Page Customizations: Procedures

The Resource Catalog provides a selection of task flows, portlets, and layout components that you can use to customize pages. You access the Resource Catalog in Page Composer from either Design view or Source view.

Aspects of components pertinent to page customizations include:

- Accessing the Resource Catalog
- Adding components
- Moving components
- Hiding components manually
- Hiding components programmatically

Accessing the Resource Catalog
To access the Resource Catalog when in Design view:

1. Access the page you want to customize.
2. Open the page in Page Composer.
3. From the existing components, select the one that you want to be the parent component.
   Alternatively, insert a box component and select it as the parent component.
4. Click the Add Content button associated with the parent component.
   The Resource Catalog appears.

Adding Components
To add components when in Design view:

1. Access the Resource Catalog.
2. Navigate through the catalog of components until you find the component you want.
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, you:

1. Select the container component in the selection pane.
2. In the Source view toolbar, click Add Content.
3. When the Resource Catalog appears, find the component you want.
4. Click the Add icon.
Moving Components

Move page components in these ways:

- In Design view, drag and drop the component.
- In Source view, you can:
  - Cut and paste.
  - Drag and drop.
  - Access the Component Properties for the container component and rearrange the components on the Child Components tab.

Hiding Components Manually

The Show Component property is a display option property that determines whether the component appears to users at run time. The default state of a component is visible. To manually hide a component, you deselect the Show Component check box in the Component Properties dialog box. If the component is a child component, then deselecting the Show Component property affects only the child component itself. However, deselecting the Show Component property of the parent component, hides the parent and all child components it contains. When you hide a parent component, you automatically hide all child components.

You can hide components using any of the following three procedures:

- Hide a child component directly
- Hide a child component from within the parent component
- Hide 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 the Show Component check box.
4. Click OK.

To hide the 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 the parent component and all child components:

1. Click the Edit icon in the box header.
2. Click the Display Options tab.
3. Deselect the Show Component check box.
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 checked. 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 is 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. Show the component: Button B</td>
<td>Determines the component that triggers the event.</td>
</tr>
<tr>
<td></td>
<td>The implication is that button B is hidden until the event occurs.</td>
<td>Determines what expression to write.</td>
</tr>
<tr>
<td>What happens when the condition is met?</td>
<td></td>
<td>Determines the effect of the action.</td>
</tr>
<tr>
<td>What happens when the event happens?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What property determines whether a component is visible?</td>
<td>The Show Component property</td>
<td>Determines the property the code affects.</td>
</tr>
</tbody>
</table>

So the logic is: If 2 is checked, then the Show Component property of B is activated.

You place the expression on the component that receives the action.

Here is sample code that we might add to the component.

```#{if checkbox2.selected = true}
After you think through the logic and find the correct expression, you add it to the property.

To access the EL Editor and add an expression to a property:

1. Click the Edit icon in the component header.
2. Click the Display Options tab.
3. Find the Show Component property and click the chevron to open the EL Editor.
4. Add EL to check for an event or condition and set the property, and based on the result, turn the property on or off.

To hide a parent component and all child components programmatically:

1. Click the Edit icon in the box’s header.
2. Click the Display Options tab.
3. Find the Show Component property and click the chevron to open the EL Editor.
4. Add EL to check for an event or condition and set the property and, based on the result, turn the property on or off.

**Deleting Components**

Delete a component only if you are certain that no other components or processes depend on the component you delete. If you have reservations or are unsure whether any dependencies exist, then hide the component instead of deleting it. If you are sure no other components or processes are dependent on a component, click the Delete icon in the component header to delete the component from the page.

---

**Warning**

If you delete a parent component, you delete all of the child components automatically.

---

**Note**

Some components, such as mandatory or indexed fields or components that are installed as part of the Oracle Fusion Applications, cannot be deleted using Page Composer.

---

**Customizing the Global Page Template: Explained**

The global page template, or UI Shell 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.

Use either of these methods to open the global page template in Page Composer:

- Click Customize Global Page template in your Settings and Actions menu.
- Open the UI Shell Template task flow.

You can make the following customizations to the global page template in Page Composer:

- Add components
- Edit components

Example: Add Expression Language to hide the Tags link.
• Delete components
  
  Example: Removing the Tags link.

Tip

When you move your cursor over the global page template, the areas that you can edit display a blue outline.

Adding Components to the Global Page Template

You can use Page Composer to open and add components to the global page template.

To add components to the global page template:

1. If you have not already opened the global page template in Page Composer, open it now as previously described.
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 be displayed.
6. After you finish making changes, click Close. When prompted, click Save to save your changes.

Editing Components in the Global Page Template

You can use Page Composer to open and edit components in the global page template.

To edit components in the global page template:

1. If you have not already opened the global page template in Page Composer, open it now as previously described.
2. Select the component that you want to edit.
3. Click Edit.
4. Edit the component properties, then click OK to save your changes.
5. After you finish making changes, click Close. When prompted, click Save to save your changes.

Deleting Components from the Global Page Template

You can use Page Composer to open and delete components from the global page template.

To delete components from the global page template:
1. If you have not already opened the global page template in Page Composer, open it now as previously described.

2. Select the component that you want to delete.

---

Tip

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. If you are done making changes to the global page template, click **Close**. When prompted, click **Save** to save your changes.

---

**Editing Footers in the Global Page Template**

You can use Page Composer to open and edit footers in the global page template.

To edit footers in the global page template:

1. If you have not already opened the UI Shell template in Page Composer, open it now as previously described.

2. Select the left region of the footer area.

---

Tip

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, right-click 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 you finish making changes, click **Close**. When prompted, click **Save** to save your changes.

---

**Note**

For information on making skin customizations, such as selecting a different color palette, see the Oracle Fusion Applications Extensibility Guide for Developers.

---

**Changing Application Logo in the Global Page Template**

To change the application logo in the global page template:

1. If you have not already opened the UI Shell template in Page Composer, open it now as previously described.
2. Click on the existing logo.

3. Click **Edit**, and edit the component properties.

---

**Note**

You can change the text in the Short Desc field. This text will display when you hover over the logo.

---

4. Click the **Style** tab.

5. In the **Background Image** field, enter the URL to access the logo.

6. In the **Other CSS** field, enter the width and height values of the logo’s image in the format: "background-size: <width>px <height>px;". For example, "background-size: 119px 25px;".

---

**Note**

Adjust the dimensions of the new logo such that it replaces the existing logo appropriately.

---

7. Click **OK** to save your changes.

---

**FAQs for Customizing Pages Using Page Composer**

**How do I back out or roll back customizations I made to a specific page?**

You should use Save and Label to save your work. The label stores the metadata for the page at a specific point in time. Then, open the page in Page Composer and access the Manage Customizations dialog box. Select a label that is older than the current label and promote the older label to the tip layer. This effectively rolls back the customizations.

---

**Note**

When you use the Manage Customizations dialog, you roll back the customizations for the page and its pagedef file only. You do not roll back the other customizations made at the label save point.

---

**How do I use metadata to perform customization-related tasks that cannot be done in Page Composer?**

In Page Composer, you can access the Manage Customizations dialog box, where you can download the metadata. You then manipulate the metadata to perform direct customization and upload the changed metadata using the same dialog box.
What happens if my customizations or personalizations make the page inaccessible?

You must contact an administrator and she can use the Manage Customizations task to view and, if necessary, delete your changes.

How can I work on customizations, but prevent users from seeing them until the customizations are complete?

You should create or select an appropriate sandbox. Set it as active to capture your customizations using Page Composer. When you are ready, publish the sandbox to make your changes available to users.

How can I reset a page or task flow to a previously saved version?

Press the Reset Page button in Page Composer (Design or Source view) to reset pages to a previously saved version or to the original out-of-the-box state.

Press the Reset Task Flow button in Page Composer (Source view only) while you have the task flow open to reset it to a previously saved version or to the original out-of-the-box state.
Oracle Sales Cloud Common Extensibility

Oracle Sales Cloud Common Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to the Common application container.

In this chapter, you will learn about:

- Each Standard object in the Common application container, and how to extend those objects using Application Composer.

Oracle Sales Cloud Common Objects: Explained

You can customize a variety of pages and regions throughout Oracle Sales Cloud using Application Composer. Application Composer lets you create custom fields and objects for a specific application, which you then add for display at run time. Objects can be standard or custom, and are attached to a specific application container, such as Marketing or 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. This topic provides an overview of each standard object in the Common application container. (This topic does not address the common component objects which are also listed within the Common application container: the interaction, note, and task objects.)

The Common standard objects that you can extend are:

- Address
- Customer Contact
- Household
- Account
- Contact
- Relationship
- Resource
Note
Some of these objects are also documented in the related topic, Customizing Oracle Sales Cloud Customer Center Pages: Explained.

Address
The Address object is an intersection object that stores a reference to a physical location/address, as well as a reference to an organization, person, or group. Any extensions to this object are available for display in customer, contact, partner, and resource address user interfaces within Oracle Sales Cloud applications.

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.

<table>
<thead>
<tr>
<th>Configuration Page in Application Composer</th>
<th>Related Page or Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Address Creation Region</td>
<td>Edit Customer - Address Create region in Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Contact - Address Create region in Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Partner Profile - Address Create region in Partner Center</td>
</tr>
<tr>
<td></td>
<td>Edit Partner - Partner Members - Address Create region within Partner Center</td>
</tr>
<tr>
<td>Edit Address Summary Table</td>
<td>Edit Customer - Addresses region in Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Contact - Addresses region in Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Partner Profile - Address region in Partner Center</td>
</tr>
<tr>
<td></td>
<td>Edit Partner - Partner members - Address region within Partner Center</td>
</tr>
<tr>
<td>Edit Person Quick Creation Region</td>
<td>Consumer Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Contact Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Person Create region within Customer Hub</td>
</tr>
<tr>
<td>Edit Organization Quick Creation Region</td>
<td>Customer Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Organization Create region within Customer Hub</td>
</tr>
</tbody>
</table>
Note

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.

Customer Contact

The Customer Contact 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 services. 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 Customer Contact object.

This table indicates, for the Customer 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>
<tbody>
<tr>
<td>Edit Person Quick Creation Region</td>
<td>Consumer Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Contact Create region within Customer Center</td>
</tr>
<tr>
<td>Edit Contact Summary Table</td>
<td>Edit Customer - Contacts table in Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Partner - Members table in Partner Center</td>
</tr>
<tr>
<td>Edit Contact Read-Only Region</td>
<td>Edit Contact - Basic Information region in Customer Center</td>
</tr>
<tr>
<td>Edit Contact Details Region</td>
<td>Edit Customer: Contact Details within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Edit Organization: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Edit Partner - Partner Members - Person Details region within Partner Center</td>
</tr>
</tbody>
</table>
**Household**

The Household object stores all intrinsic information for all households associated with the deploying company. In Oracle Sales Cloud, groups could be customers of the deploying company. The Household object is a generic profile that is applicable to all types of groups in the system. The Sales Account object is applicable only to those groups 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.

**Note**

Extend this object only when your customization is required for all households associated with the deploying company.

In Oracle Sales Cloud, groups are available as customers, so any extensions to this object are available for display in the Customer pages. Generic group extensions are also visible from the Oracle Sales Cloud Customer Hub service in the Groups work area. Extensions are not available for display in Customer Center.

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 Summary Table</td>
<td>Oracle Fusion Customer Hub application in the Groups work area</td>
</tr>
<tr>
<td>Edit Group Update Form</td>
<td>Oracle Fusion Customer Hub application in the Groups work area</td>
</tr>
<tr>
<td>Edit Group Create Form</td>
<td>Oracle Fusion Customer Hub application in the Groups work area</td>
</tr>
</tbody>
</table>

**Account**

The Account object stores all intrinsic information for all accounts associated with the deploying company. Examples of organizations 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 Customer Center 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. 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.

**Note**

Extend this object only when your customization is required for all accounts associated with the deploying company.

Any extensions to this object are available for display in various pages within Oracle Sales Cloud services. Generic organization extensions are also visible from the Oracle Sales Cloud Customer Hub service in the Organization 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 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 Details Region</td>
<td>Customer Details region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Organization Details region within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Competitor Summary region within Sales</td>
</tr>
<tr>
<td>Edit Organization Quick Creation Region</td>
<td>Customer Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Organization Create region within Customer Hub</td>
</tr>
<tr>
<td>Edit Organization Read-Only Region</td>
<td>Basic Information Region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Basic Organization Information Region within Customer Hub</td>
</tr>
</tbody>
</table>

**Contact**

The Contact object stores all intrinsic information for all persons associated with the deploying company. Examples of persons are customers (also called consumers), contacts, internal employee resources, and partner members. To extend the customer contact relationship, use the Customer Contact object. To extend internal resources, use the Resource object. To extend person customers (also called consumers), use the Sales Account object (under the Customer Center 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.) 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.

**Note**
Extend this object only when your customization is required for all contacts associated with the deploying company.

Any extensions to this object are available for display in various pages within Oracle Sales Cloud services. 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 Contact object.

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>
<tbody>
<tr>
<td>Edit Person Read-Only Region</td>
<td>Basic Information region within Customer Center (for Consumer or Person customer)</td>
</tr>
<tr>
<td></td>
<td>Contact Basic Information within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Basic Person Information region within Customer Hub</td>
</tr>
<tr>
<td>Edit Person Details Region</td>
<td>Customer Details region within Customer Center (for Consumer or Person customer)</td>
</tr>
<tr>
<td></td>
<td>Manage Contacts: Contact Details within Customer Center</td>
</tr>
<tr>
<td>Edit Person Quick Creation Region</td>
<td>Consumer Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Contact Create region within Customer Center</td>
</tr>
<tr>
<td></td>
<td>Person Create region within Customer Hub</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.

**Note**
Though a customer contact is also a type of relationship, use the Customer Contact 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 the Customer Hub. Accordingly, any extensions to this object are available for display in various pages within Oracle Sales Cloud services. 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>Edit Organization: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Edit Person: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Edit Group: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Create Group: Group Members within Customer Hub</td>
</tr>
<tr>
<td>Edit Relationship Creation Region</td>
<td>Edit Organization: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Edit Person: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Edit Group: Relationships within Customer Hub</td>
</tr>
<tr>
<td></td>
<td>Create Group Members within Customer Hub</td>
</tr>
</tbody>
</table>

**Resource**

The Resource object defines users of the Oracle Sales Cloud service. 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 that 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 Details Region</td>
<td>The Resource Profile details page is available in the Resource Directory work area, or from the Manage Resources task in the Setup and Maintenance work area.</td>
</tr>
</tbody>
</table>
Extending Simplified Pages for Contacts: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for contacts.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Create Contact page
- Contact detail page (Profile page)

**Note**
Use the Contact object in Application Composer, available with the Common application, to extend the simplified set of pages that are available for contacts.

Note that these pages are not extensible:

- Customer Contacts overview table
- Contact Overview page

**Create Contact Page**

You can extend the following items on the Create Contact page:

- Hide or show existing fields that belong to the Contact object.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).

Custom fields that you add to the Contact object appear on the Create Contact page, before the addresses set of fields.
Note that you cannot add custom buttons and actions to this page.

**Contact Detail Page (Profile Page)**

You can extend the following items on the Contact detail page (Profile page):

- Hide or show existing fields that belong to the Contact object.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

Custom fields that you add to the Contact object appear on the Contact detail page (Profile page), before the Contact Information region.

- Extend the Interactions subtab.

In Application Composer, the Interaction object is available under the Common application.

Note that you cannot make changes to these components on the page:

- Subtab regions
- Notes overview page
- Tasks
- Appointments
- Actions button
- Custom buttons and actions

You cannot create new subtabs.
Oracle Sales Cloud Account and Contact Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Sales Cloud Accounts and Contacts.

In this chapter, you will learn about:

- Each Account or Contact object, and how to extend those objects using Application Composer
- Which Oracle Sales Cloud Common objects are associated with Account and Contact pages
- How to customize Account and Contact pages using Page Composer

Customizing Oracle Sales Cloud Customer Center Pages: Explained

You can customize a variety of pages and regions in Oracle Sales Cloud Customer Center using Application Composer. Application Composer lets you create custom fields and objects, which you then add for display in the run time Customer Center application. To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

Customizing Customer Center Pages Using Application Composer

In general, every top-level object has a work area, which includes an overview page, a creation page, and a details page. When you make changes to the object, those changes can be reflected in the object’s associated work area. Customer Center, however, is unique in that its user interface pages do not include the traditional work area combination of overview page, creation page, and details page. Instead, Customer Center has a series of tree nodes that, when selected,
display user interface pages, and even a single page can be associated with multiple business objects.

This table lists Customer Center pages, and the related objects that you can access in Application Composer to customize those pages.

<table>
<thead>
<tr>
<th>Customer Center Page</th>
<th>Customer Center Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer Profile</td>
<td>Addresses region</td>
<td>Common</td>
<td>Address</td>
</tr>
<tr>
<td>Consumer Profile</td>
<td>Consumer Basic</td>
<td>Common</td>
<td>Contact</td>
</tr>
<tr>
<td></td>
<td>Information region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consumer Profile</td>
<td>Consumer Details region</td>
<td>Common</td>
<td>Contact</td>
</tr>
<tr>
<td>Consumer Profile</td>
<td>Sales Account region</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>Contact Profile</td>
<td>Address region</td>
<td>Common</td>
<td>Address</td>
</tr>
<tr>
<td>Contact Profile</td>
<td>Basic Information</td>
<td>Common</td>
<td>Customer Contact Profile</td>
</tr>
<tr>
<td></td>
<td>region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact Profile</td>
<td>Contacts region</td>
<td>Common</td>
<td>Customer Contact Profile</td>
</tr>
<tr>
<td></td>
<td>(also known as the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Contacts List)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Consumer page</td>
<td>New fields are added</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>and also the Quick</td>
<td>at the bottom of the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Consumer page</td>
<td>page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Consumer page</td>
<td>No specific region</td>
<td>Common</td>
<td>Contact</td>
</tr>
<tr>
<td>and also the Quick</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Contact page</td>
<td>No specific region</td>
<td>Common</td>
<td>Customer Contact Profile</td>
</tr>
<tr>
<td>and also the Quick</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Customer page</td>
<td>Contact Information</td>
<td>Common</td>
<td>Customer Contact Profile</td>
</tr>
<tr>
<td>and also the Quick</td>
<td>region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Customer page</td>
<td>Customer Information</td>
<td>Common</td>
<td>Account</td>
</tr>
<tr>
<td>and also the Quick</td>
<td>region</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Customer page</td>
<td>New fields are added</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>and also the Quick</td>
<td>at the bottom of the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Customer page</td>
<td>page</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Profile</td>
<td>Addresses region</td>
<td>Common</td>
<td>Address</td>
</tr>
<tr>
<td>Customer Center Page</td>
<td>Customer Center Region</td>
<td>Application</td>
<td>Underlying Business Object</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Customer Profile</td>
<td>Basic Information region</td>
<td>Common</td>
<td>Account</td>
</tr>
<tr>
<td>Customer Profile</td>
<td>Customer Details region</td>
<td>Common</td>
<td>Account</td>
</tr>
<tr>
<td>Customer Profile</td>
<td>Sales Account region</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>Edit Customer page</td>
<td>Team Members region</td>
<td>Customer Center</td>
<td>Sales Account Resource (child object to the Sales Account)</td>
</tr>
<tr>
<td>Overview &gt; Summary tab</td>
<td>Sales Accounts region (also known as the Sales Account List)</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>Real-Time Search:</td>
<td>Search region, by way of</td>
<td>Customer Center</td>
<td>Sales Account</td>
</tr>
<tr>
<td>Customers page</td>
<td>Advanced Search &gt; Add Fields</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**

To make only minor user interface changes to Customer Center pages without creating objects or fields, use Page Composer instead of Application Composer.

**Common Objects**

The common objects that are associated with Customer Center pages are:

- Account
- Contact
- Address
- Customer Contact

This table indicates which Common objects populate which Customer Center 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 Customer Center Page</th>
<th>Related Customer Center Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>Edit Customer Quick Creation Form</td>
<td>Create Customer page and also the Quick Create Customer page</td>
<td>Customer Information region</td>
</tr>
<tr>
<td>Account</td>
<td>Edit Read Only Form</td>
<td>Customer Profile</td>
<td>Basic Information region</td>
</tr>
<tr>
<td>Account</td>
<td>Edit Details Form</td>
<td>Customer Profile</td>
<td>Customer Details region</td>
</tr>
<tr>
<td>Contact</td>
<td>Edit Contact/ Consumer Quick Creation Form</td>
<td>Create Consumer page and also the Quick Create Consumer page</td>
<td>No specific region</td>
</tr>
</tbody>
</table>
### Customer Center Objects

The Customer Center objects that are associated with Customer Center pages are:

- Sales Account
- Sales Account Resource (child of the Sales Account)

This table indicates which Customer Center objects populate which Customer Center 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 Customer Center Page</th>
<th>Related Customer Center Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>Edit Read Only Form</td>
<td>Consumer Profile</td>
<td>Consumer Basic Information region</td>
</tr>
<tr>
<td>Contact</td>
<td>Edit Details Form</td>
<td>Consumer Profile</td>
<td>Consumer Details region</td>
</tr>
<tr>
<td>Address</td>
<td>Edit Detail Form</td>
<td>Customer Profile</td>
<td>Addresses region</td>
</tr>
<tr>
<td>Address</td>
<td>Edit Detail Form</td>
<td>Contact Profile</td>
<td>Addresses region</td>
</tr>
<tr>
<td>Address</td>
<td>Edit Detail Form</td>
<td>Consumer Profile</td>
<td>Addresses region</td>
</tr>
<tr>
<td>Customer Contact</td>
<td>Edit Creation Form</td>
<td>Create Customer page and also the Quick Create Customer page</td>
<td>Contact Information region</td>
</tr>
<tr>
<td>Customer Contact</td>
<td>Edit Creation Form</td>
<td>Create Contact page and also the Quick Create Contact page</td>
<td>No specific region</td>
</tr>
<tr>
<td>Customer Contact</td>
<td>Edit Read Only Form</td>
<td>Contact Profile</td>
<td>Basic Information region</td>
</tr>
<tr>
<td>Customer Contact</td>
<td>Edit Contact Details Form</td>
<td>Contact Profile</td>
<td>Contact Details region</td>
</tr>
<tr>
<td>Customer Contact</td>
<td>Edit Summary Table</td>
<td>Contact Profile</td>
<td>Contacts region (also known as the Contacts List)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Business Object</th>
<th>Configuration Page in Application Composer</th>
<th>Related Customer Center Page</th>
<th>Related Customer Center Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Account</td>
<td>Edit Creation Form</td>
<td>Create Customer page and also the Quick Create Consumer page</td>
<td>New fields are added at the bottom of the page.</td>
</tr>
<tr>
<td>Sales Account</td>
<td>Edit Creation Form</td>
<td>Create Consumer page and also the Quick Create Consumer page</td>
<td>New fields are added at the bottom of the page.</td>
</tr>
</tbody>
</table>
Using the Pages Overview Page
To add custom fields to the Customer Center pages listed in the tables above, use Application Composer's various configuration pages. You access the configuration pages in Application Composer from each object's Pages Overview page. Before you access the configuration pages, you must have already created your custom fields using Application Composer.

To access the Pages Overview page:
1. Select either the Common or Customer Center application on the main Overview page.
2. In the object tree, select the object you want to customize.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page hyperlink related to the Customer Center page that you want to customize.

Adding Custom Reports Using Page Composer
The customizations that you can make in Customer Center also include the creation of reports. This type of customization does not involve the creation of fields or objects within Application Composer. Instead, create a report using BI Answers, save the report to the Resource Catalog, and then use Page Composer to add the report to two pages: the Customer Snapshot and the Customer Overview, Analysis tab.

**Note**
The reports that you add to the Customer Snapshot are within the context of a single customer, because you view a single customer when viewing the Snapshot. The reports that you add to the Analysis tab on the Customer Overview provide context across multiple customers, because you are viewing multiple customers on the Analysis tab.

<table>
<thead>
<tr>
<th>Business Object</th>
<th>Configuration Page in Application Composer</th>
<th>Related Customer Center Page</th>
<th>Related Customer Center Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Account</td>
<td>Edit Details Form</td>
<td>Customer Profile</td>
<td>Sales Account region</td>
</tr>
<tr>
<td>Sales Account</td>
<td>Edit Details Form</td>
<td>Consumer Profile</td>
<td>Sales Account region</td>
</tr>
<tr>
<td>Sales Account</td>
<td>Edit Summary Table</td>
<td>Overview, then Summary tab</td>
<td>Sales Accounts region (also known as the Sales Account List)</td>
</tr>
<tr>
<td>Sales Account</td>
<td>Not applicable. Custom fields are automatically available from the list of additional fields.</td>
<td>Real-Time Search: Customers page</td>
<td>Search region, by way of Advanced Search, then Add Fields</td>
</tr>
<tr>
<td>Sales Account</td>
<td>Edit Regional Panes</td>
<td>Customer Work Area</td>
<td>Panes in the regional area</td>
</tr>
<tr>
<td>Sales Account Resource (child object to the Sales Account)</td>
<td>Edit Summary Table</td>
<td>Edit Customer: Sales Account Team page</td>
<td>Team Members region</td>
</tr>
</tbody>
</table>

---

Oracle Sales Cloud Account, Contact, and Household Extensibility 12-5
Customizing Oracle Sales Cloud Customer Center Using Page Composer: Explained

To customize the Customers work area and Oracle Sales Cloud Customer Center pages, use the following steps:

1. Navigate to the Customers work area.
2. Click the Administration global menu.
3. Select Customize Workarea Pages.

Note
To access Page Composer in Customer Center 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 following modes:

- Design mode
- Direct Selection mode

Customization Tasks in Design Mode
In the Customers work area and pages, in Design mode for enabled components on the Site and Job Role layers, you can perform the following customization tasks:

- Change the default saved search.
- Edit saved searches.
- Add or remove predefined content from the Customers Analytics subtab. See "Adding Custom Reports" below.
- Manage the Customers tree: hide or show nodes, change the default node, expand or collapse nodes, hide or show the Snapshot Details region.
- Add or remove predefined content to or from the Add Content to the Customers Snapshot page. See "Adding Custom Reports" below.
- Customize list of values (LOV) default values, for example, in the Regional Search region.

Customization Tasks in Direct Selection mode
In Direct Selection mode for enabled components, you can perform the following customization tasks:

- Manage the Customers tree.
- Hide or show fields.
- Change field labels.
- Make a field required.
- Make a field read-only or updateable.
- 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

The customizations that you can make in Customer Center also include the creation of reports. Create a report using BI Answers, save the report to the Resource Catalog, and then use Page Composer to add the report to two pages: the Customer Snapshot and the Customer Overview, Analysis tab.

Note

The reports that you add to the Customer Snapshot are within the context of a single customer, because you view a single customer when viewing the Snapshot. The reports that you add to the Analysis tab on the Customer Overview provide context across multiple customers, because you are viewing multiple customers on the Analysis tab.

Customizing Oracle Sales Cloud Customer Center Using Page Composer: Worked Example

This example demonstrates a set of use cases for customizing Oracle Sales Cloud Customer Center.

This example covers the following scenarios:

- Making a Form Field Read-only for a Specific Role
- Hiding a Table Column for All Users Using Design Mode
- Removing a Table Column from the View Menu for All Users Using Direct Selection Mode
- Changing the Column Label for All Users in Direct Selection Mode
- Changing the Column Width for All Users in Direct Selection Mode
- Customizing Customer Center Tree Nodes

Making a Form Field Read-only for a Specific Role

In the Customer Profile page, Customer Details region, make the customer Name field read-only for users with the Sales Representative job role.

1. Sign in with the Sales Administrator job role.
2. Navigate to the Customers work area.
3. Drill down to the Edit Customer <Customer Name> Profile page for any customer.

   Important
   Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.

4. In the Customize Customers Pages dialog box, select the Job Role layer.
5. Choose Sales Representative from the list.
6. Click OK.
   Page Composer opens in design mode.
7. Click Select.
The Direct Selection mode opens.

8. Mouse-over the **Name** field in the **Customer Details** region.
   The field is outlined automatically with a light blue border.

9. Click the **Name** field.
   The Edit property panel opens.

10. Click **Edit Component**.
    The Component Properties: Name dialog box opens.

11. Select **Read-Only**.

12. Click **OK** and validate your changes:

13. Verify your changes:

14. Click **Close**.
   a. Sign out of Oracle Applications.
   b. Sign in to Oracle Applications as a sales representative.
   c. Navigate to the Customers page and confirm your changes.

15. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization's established practices.

### Hiding a Table Column for All Users Using Design Mode

Hide the Sales Account Owner column in the Sales Account Search Results table in the Customers Summary page.

1. Sign in with the Sales Administrator job role.

2. Navigate to the Customers work area.
   The Overview, Summary Page and the Sales Account Search region opens.

   **Important**
   Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.

3. Click **Administration**.

4. In the Customize Customers Pages dialog box, select the Site layer and click **OK**.
   The page opens in Page Composer design mode.

5. To uncheck and hide this column for all users, click the table **View** menu and select **Columns Sales Account Owner**.
   The Sales Account Owner column no longer appears in the Sales Account Search Results table.

**Note**
Individual users can optionally use the View menu to display this column for themselves through personalization.

6. Validate your changes:
   a. Click Close to sign out of the Page Composer.
   b. Sign out of Oracle Applications.
   c. Sign in to Oracle Applications with any job role.
   d. Navigate to the Customers Summary page.
   e. Confirm you changes.
      The Sales Account Owner column no longer appears in the Sales Account Search Results table.

7. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization’s established practices.

8. Follow the same process to display a column that is hidden by default.

**Removing a Table Column from the View Menu for All Users Using Direct Selection Mode**

Remove the Registry ID column from the Sales Account Search Results table.

1. Sign in with the Sales Administrator job role.

2. Navigate to the Customers work area.
   The Overview, Summary Page and the Sales Account Search region appear.

   **Important**
   Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.

3. Click Administration.

4. In the Customize Customers Pages dialog box, select the Site layer.

5. Click OK.
   The page opens in the Page Composer Design mode.

6. Click Select to enter the Direct Selection mode.

7. Mouse over the Registry ID column header in the Sales Account Search Results table and click the column header.
   The column header is outlined automatically with a dark blue border and a window opens.

8. Click Edit Parent Component.
   The table column parent component properties dialog box opens with options to hide and reorder the table columns.
9. Click to uncheck the Registry ID.

The item is removed from the table View menu and from the table.

The following figure shows the Component Properties page.

![Component Properties](image)

The Registry ID column no longer appears in the Sales Account Search Results table or in the View menu.

The following figure shows the Table View Menu without the Registry column.

![Table View Menu](image)

The Registry column is no longer available at run time. Users without access to Page Composer cannot restore the removed column to the View menu.
10. Verify your changes:
   a. Click Close to sign out of Page Composer.
   b. Sign out of Oracle Applications.
   c. Sign in to Oracle Applications with any job role.
   d. Navigate to the Customers Summary page.
   e. Confirm that the Registry ID column no longer appears in the View menu.

11. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization's established practices.

Changing the Column Label for All Users in Direct Selection Mode

Change the label text for the Sell-to Address column in the Sales Account Search Results table in the Customers Summary page.

1. Sign in with the Sales Administrator job role.
2. Navigate to the Customers work area.
   The Overview, Summary Page and the Sales Account Search region appear.

   **Important**
   Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.

3. Click the Administration global menu.
4. In the Customize Customers Pages dialog box, select the Site layer.
5. Click OK.
   The page opens in the Page Composer Design mode.
6. Click Select to toggle Page Composer to Direct Selection mode.
7. Mouse-over the Sell-to Address column header in the Sales Account Search Results table and click the column header.
   The column header is automatically outlined with a blue border.
   The following figure shows the Sell-to Address link.

8. Click Edit Component.
   The following figure shows the Edit Component link.
9. In the Component Properties dialog list, you can edit the column label with the Header Text field. The column label comes from a message bundle by default. The bundle name is listed below the Header Text field. To change the label, you can use the following option: Select Text Resource. You can use this option to search for or create a new resource bundle entry. This is the production approach so that the text can be translated.

10. Click the downward arrow to the right of the Header Text field.

11. Click **Override**.

   The following figure shows the Override Menu item in the Change Property tab.

![Override Menu Item](image)

The blue dot to the right of the text field indicates that a value has been changed.

12. To enter the new label value, click the downward arrow to the right of the Header Text field again.

13. Click **Expression Builder**.

14. Enter the new value Address into the entry field on the Expression Builder Edit dialog box.

15. Click **OK** in the caption expression window.

16. Click **OK** in the Column Properties window.

   The table should now appear with the updated Address column label.

17. Verify your changes:
   a. Click **Close** to sign out of Page Composer.
   b. Sign out of Oracle Applications.
   c. Sign in to Oracle Applications with any job role.
   d. Navigate to the Customers Summary page.
e. Confirm that the column is now labeled "Address".

18. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization's established practices.

**Tip**

You can use the Reset option to restore the Header Text to the delivered message bundle string.

---

**Changing the Column Width for All Users in Direct Selection Mode**

In the Sales Account Search Results table, Customers Summary page, change the width of the Named Sales Account column to 120.

1. Sign in with the Sales Administrator job role.
2. Navigate to the Customers work area.
3. Click **Administration**.
4. In the Customize Customers Pages dialog box, select the **Site** layer.
5. Click **OK**.
6. Click **Select** to toggle Page Composer to Direct Selection mode.
7. Mouse-over the Named Sales Account column header in the Sales Account Search Results table and click the column header.
8. Click **Edit Component**.
9. In the Component Properties dialog list, change the Width field to 120.

The following figure shows the Change Property tab in the Component Properties page.
10. Click OK on the caption expression window.

11. Click OK on the Column Properties window.

   The table should now display with a wider Named Sales Account column.

12. Verify your changes:
   a. Click Close to sign out of Page Composer.
   b. Sign out of Oracle Applications.
   c. Sign in to Oracle Applications with any job role.
   d. Navigate to the Customers Summary page.
   e. Confirm the width of the Named Sales Account column.

13. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization’s established practices.

**Customizing Customer Center Tree Nodes**

Make the Customer Snapshot page the default node in Customer Center for users with the Sales Representative job role.

1. Sign in with the Sales Administrator job role.

2. Navigate to the Customers work area.

   The Overview, Summary page appears.

   **Important**

   Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making your changes.
4. In the search results table, drill into the customer record.

The Edit Customer <Customer Name> Profile page appears by default.

In the following figure the Profile node is selected and highlighted in the Customer Center Tree.

5. Click **Administration**.
6. In the Customize Customers Pages dialog box, select **Edit** for the Job Role layer.
7. Select Value, **Sales Manager**.
8. Click **OK**.

The page opens in the Page Composer Design mode.

9. In the Customer Center pane, select **Actions - Manage Customer Tree**.

The following figure shows the Edit Customer page with the Manage Customer Tree item selected in the Customer Center pane.
The Manage Customer Tree dialog box appears.

10. To change the default display to the Snapshot page, click the **Snapshot** row in the Tree Nodes table to set it as the default.

11. Click **Save**.

**Tip**

In the Manage Customer Tree dialog box, you can show or hide the nodes. In the View menu, you can expand and collapse the top level nodes.

The following figure shows the Manage Customer Tree Dialog box.
12. Alternatively, you can set the node to appear by default in two additional ways:

- Select the Set as Default menu item in the table's Action menu.
- Click the Set as Default icon in the table toolbar.

The following figure shows the Set as Default menu item in the Actions menu.

13. Click the **Set as Default** icon.

14. Click **Save and Close**.

15. Verify your changes:
   a. Click **Close** to sign out of Page Composer.
   b. Sign out of Oracle Applications.
   c. Sign in to Oracle Applications with the Sales Manager role.
   d. Navigate to the Edit Customer <Customer Name>Profile page.
12-18 Oracle Sales Cloud Extending Sales

Drill down into the customer record and confirm that the Snapshot page appears by default.

16. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization’s established practices.

Extending Simplified Pages for Customers: Explained

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for customers. The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Customers overview table
- Create Customer page
- Edit Customer page (Profile page)
- Create Consumer page
- Edit Consumer page (Profile page)

Note
To extend the simplified set of pages that are available for customers, use the Contact and Account objects in Application Composer, available with the Common application. You will also use the Sales Account object, which is available with the Customer Center application.

Note that these pages are not extensible:
- Customer Overview page
- Consumer Overview page

Customers Overview Table
You can extend the following items in the Customers overview table:
- Hide or show existing fields that belong to the Sales Account object.
- Reorder columns.
- Change column labels.
- Add custom fields (all types), using the Sales Account object.
Note that you cannot make changes to these components on the page:

- Show filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
  - Custom font size for Name
  - Drilldown
- Custom buttons and actions

**Create Customer Page**

You can extend the following items on the Create Customer page:

- Hide or show existing fields that belong to the Account and Sales Account objects.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
  - Custom fields that you add to the Account object appear on the Create Customer page, before the addresses set of fields.
  - Custom fields that you add to the Sales Account object appears at the bottom of the page.

Note that you cannot add custom buttons and actions to this page.

**Edit Customer Page (Profile Page)**

These regions on the Edit Customer page (Profile page) are extensible:

- Basic Information region

  Use the Account object in Application Composer to extend this region.
• Sales Account region

Use the Sales Account object in Application Composer to extend this region.

• Sales Account Team subtab

Use the Sales Account Resource object in Application Composer to extend this region. Then, navigate to the Pages node for the Sales Account object, and edit the Sales Account Team subtab in the Subtabs region.

You can extend the following items on the Edit Customer page (Profile page):

• Hide or show existing fields.
• Reorder fields.
• Change display labels.
• Add custom fields (all types).

Note that you cannot make changes to these components on the page:

• Subtab regions
• Contacts
• Opportunities
• Leads
• Actions button
• Custom buttons and actions
• You cannot create new subtabs.

Create Consumer Page

You can extend the following items on the Create Consumer page:
• Hide or show existing fields that belong to the Contact and Sales Account objects.

• Reorder fields.

• Change field labels.

• Add custom fields (all types).

  • Custom fields that you add to the Contact object appear on the Create Consumer page, before the addresses set of fields.

  • Custom fields that you add to the Sales Account object appears at the bottom of the page.

Note that you cannot add custom buttons and actions to this page.

**Edit Consumer Page (Profile Page)**

These regions on the Edit Consumer page (Profile page) are extensible:

- Basic Information region
  
  Use the Contact object in Application Composer to extend this region.

- Sales Account region
  
  Use the Sales Account object in Application Composer to extend this region.

- Sales Account Team subtab
  
  Use the Sales Account Resource object in Application Composer to extend this region. Then, navigate to the Pages node for the Sales Account object, and edit the Sales Account Team subtab in the Subtabs region.

You can extend the following items on the Edit Consumer page (Profile page):
• Hide or show existing fields.
• Reorder fields.
• Change display labels.
• Add custom fields (all types).

Note that you cannot make changes to these components on the page:

• Subtab regions
  • Opportunities
  • Leads
• Actions button
  • Custom buttons and actions

You cannot create new subtabs.
Oracle Sales Cloud Sales and Opportunity Management Extensibility

Read this chapter to learn about how to extend objects and pages that belong to Oracle Sales Cloud Sales and Oracle Sales Cloud Opportunity Management.

In this chapter, you will learn about:

- Each Oracle Sales Cloud Sales and Oracle Sales Cloud Opportunity Management object, and how to extend those objects using Oracle Application Composer
- How to customize Oracle Sales Cloud Sales and Oracle Sales Cloud Opportunity Management pages using Page Composer

Customizing Sales Pages Using Application Composer: Explained

You can customize a variety of regions in Oracle Sales Cloud, including sales Competitor and sales Reference Customer regions, using Application Composer. Application Composer lets you create custom fields and objects, which you then add for display in the runtime Competitor and Reference Customer modules. To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

Understanding Which Oracle Sales Pages Are Extensible

To customize Sales Competitor and Sales Reference Customer 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 Competitor and Sales Reference Customer objects that are associated with sales pages are:

- Sales Competitor
- Sales Reference Customer
The following table lists sales Competitor and sales Reference Customer pages and regions, and the related objects that you can access in Application Composer to customize those pages. For example, the Edit Competitor page is extensible. To create custom fields that you can later add to the Edit Competitor page, you must select the Competitor object in Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Sales Page</th>
<th>Sales Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competitors Overview page</td>
<td>Competitors Summary (List View)</td>
<td>Sales</td>
<td>Sales Competitor</td>
</tr>
<tr>
<td>Create Competitor page</td>
<td>Default details region</td>
<td>Sales</td>
<td>Sales Competitor</td>
</tr>
<tr>
<td>Edit Competitor page</td>
<td>Default details region</td>
<td>Sales</td>
<td>Sales Competitor</td>
</tr>
<tr>
<td>Sales References Overview page</td>
<td>References Summary (list view)</td>
<td>Sales</td>
<td>Sales Reference Customer</td>
</tr>
<tr>
<td>Edit Sales Reference page</td>
<td>Default details region</td>
<td>Sales</td>
<td>Sales Reference Customer</td>
</tr>
</tbody>
</table>

Next, expose those custom fields that you created by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**

To make only minor user interface changes to sales Competitor and sales Reference Customer pages without creating new objects or fields, use Page Composer instead of Application Composer.

**Adding Your Changes to the Runtime Application**

To add custom fields to the Sales Competitor and Sales Reference Customer 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 Sales Competitor and Sales Reference Customer regions. You access the configuration pages in Application Composer from the Pages nodes under the following objects: Sales Competitor and Sales Reference Customer.

To access the sales Competitor and sales Reference Customer configuration pages:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the object whose pages or regions you want to customize. For example, select the Sales Competitor object.
3. Next, select the Pages node for the object.

The following table indicates which Sales Competitor and Sales Reference Customer objects populate which Sales Competitor and Sales Reference Customer 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 Page</th>
<th>Related Sales Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Competitor</td>
<td>Competitors Summary (list view) in Overview page: Select Sales Competitor object, select Pages link, and then select the Edit Summary Table link.</td>
<td>Competitors Overview page</td>
<td>Competitors Summary (list view) region</td>
</tr>
<tr>
<td>Sales Competitor</td>
<td>Create Competitor page: Select Sales Competitor object, select Pages link, and then select the Edit Creation Page link.</td>
<td>Create Competitor page</td>
<td>Create Competitor page</td>
</tr>
<tr>
<td>Sales Competitor</td>
<td>Edit Competitor page, default details: Select Sales Competitor object, select Pages link, and then select the Edit Summary Form link.</td>
<td>Edit Competitor page</td>
<td>Edit Competitor page</td>
</tr>
<tr>
<td>Sales Reference Customer</td>
<td>References summary (list view) in Overview page: Select Sales Reference Customer object, select Pages link, and then select the Edit Summary Table link.</td>
<td>References Overview page</td>
<td>References summary (list view)</td>
</tr>
<tr>
<td>Sales Reference Customer</td>
<td>Edit Reference page: Select Sales Reference Customer object, select Pages link, and then select the Edit Summary Form link.</td>
<td>Edit Reference page</td>
<td>Edit Reference page</td>
</tr>
</tbody>
</table>

**Customizing Opportunity Pages Using Application Composer: Explained**

You can customize a variety of opportunity page and regions using Application Composer. Application Composer lets you modify pages and create custom fields and objects, which you then make available in the run time opportunity application. To access Application Composer, select **Application Composer** from the Navigator menu, under the Tools > Customization category. The opportunity areas that can be customized are listed under **Sales** in the Application list of values.

Note that in the Sales area in the Application Composer screens, you can also make changes to the base sales pages, including the competitors and reference...
customer regions and fields. For more information on customizing these areas, see the topic, Customizing Sales Pages Using Application Composer: Explained.

**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 the Application Composer to customize those pages.

The Sales objects that are associated with opportunity pages are:

- Opportunity
- Opportunity Contact
- Opportunity Revenue
- Opportunity Team Member

The following table lists the opportunity pages and regions that you can access in the Application Composer to customize those pages. For example, in the Edit Opportunity page, both the summary portion and the details portion, are extensible. To create custom fields that you can later add to the Edit Opportunity page, you must select the Opportunity object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Sales Page</th>
<th>Sales Region</th>
<th>Application</th>
<th>Underlying Business Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunity Overview page</td>
<td>Opportunity summary (list view) table</td>
<td>Opportunity</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Create Opportunity page</td>
<td>Create Opportunity page</td>
<td>Opportunity</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Summary region (above the Detail region available by selecting the Show More option)</td>
<td>Opportunity</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Detail region (expand the Show More option)</td>
<td>Opportunity</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Revenue table</td>
<td>Opportunity</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>
<td>Opportunity</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Contacts summary table</td>
<td>Opportunity</td>
<td>Opportunity Contact (child object of opportunity)</td>
</tr>
<tr>
<td>Edit Opportunity page</td>
<td>Contacts summary table actions</td>
<td>Opportunity</td>
<td>Opportunity Contact (child object of opportunity)</td>
</tr>
</tbody>
</table>
Next, expose those custom fields on the opportunity page or region by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**
To make only minor user interface changes to opportunity pages without creating new objects or fields, use Page Composer instead of the Application Composer.

### Adding Your Changes to the Runtime Application
To add custom fields to the opportunity regions 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 desired opportunity regions. You access the configuration pages in the Application Composer from the Pages nodes under the following objects: Opportunity, Opportunity Contact, Opportunity Revenue, and Opportunity Team Member.

To access the opportunity configuration pages:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the object whose pages or regions you want to customize. For example, select the Opportunity object.
3. Next, select the select the Pages node for the object. For a few objects, you can only customize fields, not pages or regions. For those objects, a Pages node will not be available.

The following table indicates which opportunity objects populate which opportunity 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></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunity</td>
<td>Opportunity summary (list view) in Overview page: Select Opportunity object, select Pages link, and then select the Edit Summary Table link</td>
<td>Overview page</td>
<td>Opportunity summary (list view)</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------</td>
<td>-------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Create Opportunity page: Select Opportunity object, select Pages link, and then select the Edit Creation Page link.</td>
<td>Create Opportunity page</td>
<td>Create Opportunity page</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Summary region, Default area (above the Show More area): Select Opportunity object, select Pages link, and then select the Edit Summary Form link.</td>
<td>Edit Opportunity page</td>
<td>Summary region, Default area</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Summary region, Detailed area (available by expanding the Show More area): Select Opportunity object, select Pages link, and then select the Edit Summary Form link.</td>
<td>Edit Opportunity page</td>
<td>Summary region, Detailed area</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Revenue Items table: Select Opportunity object, select Pages link, and then select the Edit Revenue Table link.</td>
<td>Edit Opportunity page</td>
<td>Revenue Items table</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Revenue table, detail stamp: Select Opportunity object, select Pages link, and then select the Edit Revenue Table link.</td>
<td>Edit Opportunity page</td>
<td>Revenue Items table, detail stamp</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Opportunity contact summary (list view) table: Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Contact in the table and edit it using Actions menu.</td>
<td>Edit Opportunity page</td>
<td>Contacts tab, contacts summary (list view) table</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Edit Opportunity page, Opportunity contact summary (list view) table actions: Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Contact in the table and edit it using Actions menu.</td>
<td>Edit Opportunity page</td>
<td>Contacts tab, contacts summary (list view) table actions</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Team in the table and edit it using Actions menu.</td>
<td>Edit Opportunity page</td>
<td>Opportunity team summary (list view) table</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Team in the table and edit it using Actions menu.</td>
<td>Edit Opportunity page</td>
<td>Opportunity team summary (list view) table, detail stamp</td>
</tr>
<tr>
<td>Opportunity Contact</td>
<td>Opportunity Contact detail view fields: Select Opportunity Contact &gt; Fields: Select Opportunity Contact object and then select Fields link.</td>
<td>Edit Opportunity page</td>
<td>Contact detail view (fields only)</td>
</tr>
<tr>
<td>Opportunity Revenue</td>
<td>Opportunity Revenue Items summary (list view) table columns: Select Opportunity Revenue object, select Pages link, and then select Edit Summary Table link.</td>
<td>Edit Opportunity page</td>
<td>Opportunity Revenue Items summary (list view) table</td>
</tr>
<tr>
<td>Opportunity Team Member</td>
<td>Opportunity Team member detail view fields: Select Opportunity Team Member object and then select Fields link.</td>
<td>Edit Opportunity page</td>
<td>Opportunity Team member detail view (fields only)</td>
</tr>
</tbody>
</table>
The Oracle Sales Cloud Sales dashboard, Opportunity work area, and related pages support Page Composer customization tasks in both Design and Direct Selection mode, similar to the Customers work area. To customize the Sales dashboard, work areas, and pages, you must have the Sales Administrator privileges.

**Customizing the Sales Dashboard**

The Sales dashboard consists of a regional pane and a local pane. The local pane has two subtabs: Home and Pipeline. To customize the Sales dashboard, use the following steps:

1. Navigate to the dashboard page.
2. Select *Customize Home Pages* from the Administration global menu.

In the Design Mode at the Site and Job Role layers, you can do the following customization tasks:

- 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 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: show or hide box, reorder child regions, display options, and style.
- Add, edit, or remove from dashboards the ADF components, for example, regions, hyperlinks, images, text boxes, movable boxes, and web pages.

The Direct Selection Mode customizations are available at the Site and Job Role layers. You can do the following customization:

- Change the order of links in the Tasks regional bin.

**Customizing Sales Pages**

To customize Sales pages, you must have Sales Administrator privileges.

1. Navigate to the Sales work area.
2. Navigate to the page that you want to customize, for example, the Edit Opportunity and Revenue Item pages.
3. You can launch Page Composer in any of these pages by selecting *Administration, Customize work area Pages*.

You can do the following customizations:

- Hide or show a field.
• Change a field label.
• Make a field required or not.
• Make a field read-only or updateable.
• 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.

Hiding and Reordering Opportunity Fields Using Page Composer: Worked Example

This example demonstrates how, using Page Composer, you can hide fields, make fields required, and change the order of fields in the edit opportunity pages, for specific job roles. The purpose of the tasks in this example is to make these changes for all salespeople using the instance.

Prerequisites
Before you begin, consider the following setup requirements or prerequisites:
• You will perform this task as a user with the Sales Administrator job role.
• You will first make the custom changes in a Page Composer sandbox and then roll the changes out to all users with the Sales Representative job role by publishing the sandbox. Therefore, before you begin, familiarize yourself with your organization's process for creating and publishing Page Composer sandboxes.

Invoking Page Composer Edit and Direct Selection Modes
First, invoke Page Composer's edit mode:
1. Sign in to the application as a user with the Sales Administrator job role.
2. From the Navigator, select Opportunities.
3. Drill into an opportunity record: Select the name hyperlink of an opportunity in the list.
   The edit page for the opportunity you selected opens.
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, check the Edit option for the Job Role layer.
6. In the Value column next to Job Role, select Sales Representative from the drop list.
7. Select **OK**.

The page opens in Page Composer design mode. A bar appears across the top of the page along with the text, “Editing: Opportunities” and “Edit Layer: Job Role”.

8. Next, toggle Page Composer to Direct Selection mode: Select the **Select** icon in the global region.

Direct Selection mode is now invoked, allowing you to make field-level changes.

**Hiding a Field and Changing the Order of a Field**

To hide the **Worst Case** field and change the order of the **Attachments** and **Partners** fields, use the following steps:

1. With Page Composer in Direct Selection mode, in the Additional Details region of the edit opportunity page, hover over the **Worst Case** field.

A colored box appears around the field and its label.

2. Click inside the box around the **Worst Case** field.

A dialog box opens with two options: Edit Component and Edit Parent Component.

3. Select the **Edit Parent Component** option.

The Component Properties: panelformlayout window opens.

4. In the Component Properties: panelformlayout window, clear the check box next to the **Worst Case** field.

The following figure shows the Component Properties dialog box with Worst Case selected.
5. Select the down arrow to the right of the **Attachments** field to move it below the **Partners** field.
The order of the two fields changes.

6. Click **OK**.

**Making a Field Read-Only**

To make the **Win Probability** field read-only, use the following steps:

1. While still in Page Composer Direct Selection mode, hover over and then click the **Win Probability (%)** label.
   A dialog box opens with two options: Edit Component and Edit Parent Component.

2. Select the **Edit Component** option.
   The Component Properties: Win Probability (%) window opens.

3. In the Component Properties: Win Probability (%) window, select the **Read only** check box.
   The field becomes read-only, as shown in the following figure.

4. Select **OK**.

**Making a Field Required and Unsortable**

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 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. Select 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. Select 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 Using Page Composer: Worked Example

This example demonstrates how, using Page Composer, you can add a new subtab or page labelled “Quota” to the Sales Dashboard, plus add two sales quota reports to the page. The purpose of the task is to create, for all salespeople using the instance, a unique subtab or page in the Sales Dashboard with quota reports.

Prerequisites

Before you begin, consider the following setup requirements or prerequisites:

- Perform this task as a user with the Sales Administrator job role.
- First make the custom changes in a Page Composer sandbox and then roll the changes out to all users with the Sales Representative job role by publishing the sandbox. Before you begin, familiarize yourself with your organization’s process for creating and publishing Page Composer sandboxes.

Start Page Composer Design Mode

You must start Page Composer’s edit mode:

1. Sign in to the application as a user with the Sales Administrator job role.
2. From the Navigator, select **Sales Dashboard**. Or, select the **Sales** tab on the landing page.
3. 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.
4. In the Customize Opportunities Pages dialog box, check the **Edit** option for the Job Role layer.
5. In the Value column next to Job Role, select Sales Representative from the drop list.
6. Select **OK**.
The page opens in Page Composer design mode. A bar appears across the top of the page along with the text, "Editing: Sales" and "Edit Layer: Job Role".

**Add New Subtab to Dashboard**

Now, add a new, blank subtab (or page) to the dashboard:

1. With Page Composer still in design mode, in the subtab bar on the Sales dashboard, click the + Tab tab.
2. Type "Quota" in the text box to rename the subtab.
3. Select the Rename this tab button.

The subtab now appears with the name, Quota.

**Add Reports to Quota Subtab**

Next, add reports to the Quota subtab:

1. With Page Composer still in design mode, select the Add Content button in one of the portlets present on the page.
2. In the Add Content dialog box, open the Oracle Business Intelligence folder.
3. Open the Shared Sales Reports folder.
4. Open the Sales Quota Management folder.
5. Select the Add icon next to the Resource Quota History Chart link.
   In the background, the report will be added to the page. The Add Content dialog will remain open.
6. Select the Add icon next to the Territory Quota History Bar Chart.
   In the background, the report will be added to the page. The Add Content dialog will remain open.
7. Close the Add Content dialog.
   You now have reports on the page.

**Change Page Layout**

Optionally, you can change the layout of the page from the default three-column layout, following these steps:

1. With Page Composer still in design mode, select the Layout button in the upper right corner of the dashboard page.
2. Select one of the layout options from the choices presented.

**Commit Your Changes**

When you are ready to commit your customization changes to the main line, perform these steps:

1. Select the Close button in the global region to sign out of the 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. Select the Publish button to commit your changes.

**Verify the Changes**

Finally, sign in to the applications as a salesperson and verify your changes:

1. Select the Close button in the global region to sign out of the Page Composer editor.

2. Sign out of the application.

3. Sign in as a user with the Sales Representative role.

4. Navigate to the Sales Dashboard: Select Sales Dashboard from the Navigator or select the Sales tab on the landing page.

5. Verify that the Quota tab is present and the reports are displaying as expected.

**Creating Opportunity Saved Searches Using Page Composer: Worked Example**

This example demonstrates how, using Page Composer, you can create a custom opportunity saved search for a specific job role or at site level.

**Prerequisites and Preliminary Concepts**

Before you begin, consider the following setup requirements or prerequisites:

- Perform this task as a user with the Sales Administrator job role.
- Familiarize yourself with your organization’s process for creating and publishing customization sandboxes. As a best practice, your organization may 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.
- To enable custom saved searches at the site level, follow the same exercise as described in this topic, except pick Site as the customization level.

**Tip**

Sales opportunities (and other Sales Cloud objects like leads) have the concept of record sets, and it is this set of data that is being queried for in opportunity search. Record sets represent a user’s data set, or the data that he has access to. A record represents a single opportunity and a record set represents multiple opportunities. Keep in mind that a user’s membership in the opportunity sales team and territory hierarchy determine the records that he has access to. Refer to the help topic, How Opportunity Information Is Secured: Explained, for more information on opportunity data security.

**Start Page Composer**

First, start Page Composer’s design mode:
1. Sign in to the application as a user with the Sales Administrator job role.
2. From the Navigator, select Opportunities.
3. 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.
4. In the Customize Opportunities Pages dialog box, check the Edit option for the Job Role layer.
   If a saved search is needed for all roles, then select Site layer.
5. In the Value column next to Job Role, select Sales Representative from the drop list.
   Selecting Sales Representative means only users with this role can see the saved search.
6. Select OK.
   The page opens in Page Composer design mode. A bar appears across the top of the page along with the text, "Editing: Opportunities" and "Edit Layer: Job Role".

Create and Run the Search

Next, create and run the custom saved search.

In this example, you are creating a custom saved search based on:
- Record sets where the salesperson is on the opportunity sales team;
- Opportunity close dates that fall within the current quarter of the calendar; and
- Opportunities in open status.

1. With Page Composer in design mode, open the Search panel in the Opportunities overview page/work area.
2. From the Saved Searches list, retrieve the Open Opportunities predefined search.
   Alternatively, you can use the Close Date predefined search as a template and add the fields listed to this search
3. Set up the search criteria for the new saved search, using the values in the following table:

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Set</td>
<td>Equals: Records where I am on the team</td>
</tr>
<tr>
<td>Close Period</td>
<td>Equals: Current Quarter</td>
</tr>
<tr>
<td>Status</td>
<td>Equals: Open</td>
</tr>
</tbody>
</table>
4. Run the search: Select the **Search** button.

5. Save the search: Select the **Save** button.

6. In the Create Saved Search dialog box, enter a meaningful name in the **Name** field. Since this query returns all opportunities that the salesperson is actively involved as an opportunity team member for next quarter, you might name it “All my next quarter opportunities”.

7. Set these other options, as desired:
   - **Set as Default**: This option makes the saved search the default in the drop-list of saved searches.
   - **Run Automatically**: This option makes the saved search run automatically when users navigate to the search panel in the Opportunities work area.

8. Select **OK**.

9. Select the Close button in the global region to sign out of Page Composer design mode.

**Verify the Changes**

Finally, sign in to the applications as a salesperson and verify your changes:

1. Sign in to the applications as a user with the Sales Representative role.
2. Navigate to the Opportunities work area.
3. Confirm that the saved search you created is available to you.

**Extending Simplified Pages for Opportunities: Explained**

Using Application Composer, you can change many items that appear on the simplified set of pages that are available for opportunities.

The pages that are available for extensibility using Application Composer are listed below, and are described in this topic.

- Opportunity landing page
- Create Opportunity page
- Edit Opportunity page

**Opportunity Landing Page**

You can extend the following items in the list that appears on the Opportunity overview table:

- Hide or show existing fields.
- Reorder columns.
- Change column labels.
- Add custom fields (all types).

This figure shows the Opportunities landing page.
Note that you cannot make changes to these components on the page:

- Show filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
  - Win Probability
  - Custom font size for Name
- Drilldown
- Embedded analytics
- Custom buttons and actions

**Create Opportunity Page**

You can extend the following items on the Create Opportunity page:

- Hide or show existing fields.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
- Add custom fields to the Revenue Items table.

  In Application Composer, add custom fields to the Opportunity Revenue object, and then use the Edit Revenue Table configuration page available for the Opportunity object, under the Pages node.

Note that you cannot add custom buttons and actions to this page.

**Edit Opportunity Page**

These regions on the Edit Opportunity page are extensible:

- Opportunity region
- Revenue Items table
In Application Composer, add custom fields to the Opportunity Revenue object, and then use the Edit Revenue Table configuration page available for the Opportunity object, under the Pages node.

- Opportunity Contacts subtab
  In Application Composer, add custom fields to the Opportunity Contact object. Then, navigate to the Pages node for the Opportunity object, and edit the Contacts subtab in the Opportunity Detail Tab region.

- Opportunity Team subtab
  In Application Composer, add custom fields to the Opportunity Team object. Then, navigate to the Pages node for the Opportunity object, and edit the Team subtab in the Opportunity Detail Tab region.

- Notes subtab
  In Application Composer, the Note object is available under the Common application.

- Interactions subtab
  In Application Composer, the Interaction object is available under the Common application.

This figure shows the Edit Opportunity page.

You can extend the following items on the Edit Opportunity page:

- Hide or show existing fields.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

Note that you cannot make changes to these components on the page:

- Subtab regions
  - Notes overview page
  - Tasks
  - Appointments
- Appointments/Task region on Profile page
- Actions button
- Custom buttons and actions
- You cannot create new subtabs.
Extending Simplified Pages for Opportunities Using Application Composer: 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.
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.

   ![Application Composer](image)

   This figure shows the simplified page for Opportunities

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 **Fuse Opportunity Create** region.

8. In the Fuse Opportunity Create page, select **Budget Amount** from the **Available Fields** list in the **Configure Detail Form** region. This figure shows the Configure Detail Form in the Fuse Opportunity Create page.

9. Click the First arrow button to move the Budget Amount 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.

10. Click Save and Close.

### Extending Simplified Pages for Opportunities using Page Composer: Worked Example

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 it for adding or removing predefined content from the Resource Library. All changes are done and stored in the UI layer.

You can extend simplified pages using Page Composer. Perform the changes in sandbox to test it first.

This example demonstrates how you can hide the **Include in Forecast** field for a sales representative in the Opportunities page and leave it available for the rest of the roles.

### Extending Simplified Pages for Opportunities Using Page Composer

The steps to hide the **Include in Forecast** field for a sales representative are:

1. Within a sandbox, sign in as a Sales Administrator in the Oracle Sales Cloud Sales Application to perform your configuration.

2. Click the role choice list and select Customize Pages. In this example, click Matt Hooper and select Customize Pages. This figure shows the Simplified Page for the Opportunity page with the **Customize User Interface** option selected.
3. In the Customize Application Pages, select the layer that you want to edit.

You can customize three layers:

- Global
- External/Internal
- Job Role

4. Click **Job Role** and select Sales Representative.

This figure shows the Customize User Interface window.

5. After you invoke the Page Composer in a particular layer (in this case, the Job Role layer) two buttons are displayed.
The two buttons are:

- **Design**: Represents design mode used to extend transactional pages.
- **Select**: Allows you to select components in a page to edit.

This figure shows the Page Composer view in the Job Role layer.

6. Drill into an opportunity and click Select.

7. In the Edit Opportunity page that appears, select the **Include in Forecast** field that you want to hide.

This figure shows the Edit Opportunity page.

8. After you select the field, two options appear: **Edit Component** and **Edit Parent Component**.

9. Select **Edit Component**.

10. In the Component Properties window, notice that the **Show Component** check box is selected, which means that this field will be visible in the Opportunity page.

This figure shows the Component Properties window.
11. Deselect **Show Component** to hide this field in the Opportunity page. Click **Apply** and click **OK**.

12. Notice that the field no longer appears in the Edit Opportunities page. Click **Close** to close Page Composer.

13. Note that in the Edit Opportunity Page, the field is still present because you are in a sales administrator role.

14. Sign out of the application and sign in as a sales representative.

15. On the Opportunity landing page, drill into an opportunity. In the Edit Opportunity page, notice that the **Include in Forecast** field is hidden or not visible.

   This figure shows the Edit Opportunity page for the sales representative role.
Oracle Sales Cloud Sales Forecasting Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Sales Cloud Sales Forecasting.

In this chapter, you will learn about:
- Each Oracle Sales Cloud Sales Forecasting object, and how to extend those objects using Oracle Application Composer

Customizing Oracle Sales Cloud Sales Forecasting Pages: Explained

You can customize a variety of regions in Oracle Sales Cloud Sales Forecasting using Application Composer. Application Composer lets you create custom fields and objects, which you then add for display in the runtime Sales Forecasting application. To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

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.
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 Runtime Application**

To add custom fields to the Sales Forecasting 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 Sales Forecasting regions. You access the configuration pages in Application Composer from the Pages node under the Forecast Territory Details object.

To access the Pages Overview page:

1. Select the Sales application on the main Overview page.
2. In the object tree, select the Forecast Territory Details object.
3. Select the Pages node.

<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>Edit Forecast page</td>
<td>Forecast Items tab</td>
<td>Sales</td>
<td>Forecast Item (child of Forecast Territory Details)</td>
</tr>
</tbody>
</table>
4. On the Pages Overview page, 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 click Edit Summary Form.</td>
<td>Edit Forecast page</td>
<td>Territory Details region</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To view your custom fields in the Territory Details region, you must expand the Show More option which appears above the Forecast Items tab.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Oracle Sales Cloud Marketing Extensibility

Oracle Sales Cloud Marketing Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Sales Cloud Marketing.

In this chapter, you will learn about:

• How to customize various sales cloud marketing-related pages
• How to extend the sales campaign object using Application Composer
• How to create a sales lead validation rule using Application Composer

Customizing Marketing Campaign Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Campaign object, which you then add for display in the Campaigns work area. This topic describes the extensible options for marketing multistage campaigns.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Campaigns work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing 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. Choose the **Fields** node to create custom fields, or modify standard fields.
3. Choose 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 need to 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 new object records
- Detail pages 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 page</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>
<tr>
<td></td>
<td>Campaigns, Edit Multistage Campaign, Overview region</td>
</tr>
<tr>
<td></td>
<td>Campaigns, Edit Multistage Campaign, Design tab Show Table stage list table region</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Manage Campaign page by accessing the appropriate Application Composer configuration page, listed in the following section.

Note

To make only minor user interface changes to Marketing Campaign pages without creating new objects or fields, use Page Composer instead of the Application Composer.
Adding or Modifying Fields

To add custom fields to the marketing campaign regions listed above, 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 Runtime Application

To add custom fields to the Campaigns 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 desired 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 Marketing 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 hyperlink related to the Campaigns region that you want to customize.

Customizing Marketing Campaign Stage Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Campaign Stages object, which you then add for display in the Campaigns work area. This topic describes the extensible options for marketing campaign stages.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Campaign Stages work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing 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. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose 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 need to 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 new object records
- Detail pages 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 Uti</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 user interface changes to Campaign Stages pages without creating new 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 listed above, 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 Runtime Application**

To add custom fields to the Campaign Stage regions 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 desired 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 **Marketing** 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 hyperlink related to the Campaign Stages region that you want to customize.

## Customizing Marketing Campaign Member Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Campaign Member object, which you then add for display in the Campaigns work area. This topic describes the extensible options for marketing campaign members.

Access Application Composer from any Sales Cloud service at run time by using the **Navigator** menu, and selecting **Application Composer** under the **Tools > Customization** category.

### Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Campaign Member work area.

Follow these steps:

1. Navigate to the **Application Composer**, select **Marketing** 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. Choose the **Fields** node to create custom fields, or modify standard fields.
3. Choose 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 need to 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 new object records
• Detail pages 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 regions, 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 user interface changes to Campaign Member pages without creating new 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 listed above, 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 Runtime Application**

To add custom fields to the Campaign Member 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 desired Campaign Member regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Campaign Member object.

To access the Pages Overview page:

1. Select the Marketing 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 hyperlink related to the Campaign Member region that you want to customize.

**Customizing Marketing List Pages Using Application Composer: Explained**

Use the Application Composer to create custom fields for the Marketing List object, which you then add for display in the Audience work area. Marketing
List does not have any child objects. This topic describes the extensible options for marketing lists.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

**Extensibility: Quick Steps**

This section provides a brief overview of the steps to customize the audience work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing 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. Choose the Fields node to create custom fields, or modify standard fields.

   3. Choose 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 need to 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 new object records
- Detail pages 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 Work Area 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 user interface changes to Marketing List pages without creating new objects or fields, use Page Composer instead of the Application Composer.

---

**Adding or Modifying Fields**

To add custom fields to the Audience regions listed above, 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 Runtime Application**

To add custom fields to the Audience regions 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 desired Audience List regions. You 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 Marketing 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 hyperlink related to the Audience work area that you want to customize.

---

**Customizing Marketing Treatment Pages Using Application Composer: Explained**

Use the Application Composer to create custom fields for the Marketing Treatment object, which you then add for display in the Treatments work area. This topic describes the extensible options for marketing treatments.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

**Extensibility: Quick Steps**

This section provides a brief overview of the steps to customize the Treatment work area.

Follow these steps:
1. Navigate to the **Application Composer**, select **Marketing** 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. Choose the **Fields** node to create custom fields, or modify standard fields.
3. Choose 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 need to 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 new object records
- Detail pages 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 any of these tabs, 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 user interface changes to Treatment pages without creating new 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 Runtime Application

To add custom fields to the Treatments regions 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 desired Treatments regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Treatment object.

To access the Pages Overview page:

1. Select the Marketing 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 hyperlink related to the Treatments region that you want to customize.

Customizing Marketing Response Pages Using Application Composer: Explained

Use Application Composer to create custom fields for the Marketing Response object, which you then add 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.

To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the response processing work area.

Follow these steps:

1. Navigate to Application Composer, select Marketing 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. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose 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 need to 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 new object records
- Detail pages 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>Edit Summary Table overview page</td>
<td>Manage Responses, Responses, Responses table details</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Manage Responses, Create Response, Summary region</td>
</tr>
<tr>
<td></td>
<td>Manage Responses, Edit Response, Summary 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 interface changes to Manage Response pages without creating new 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 Runtime Application

To add custom fields to the Manage Response region 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 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 Marketing 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 hyperlink related to the Manage Response region that you want to customize.

**Customizing Marketing Budget Pages Using Application Composer: Explained**

Use the Application Composer to create custom fields for the Marketing Budget object, which you then add for display in the Budgeting work area. This topic describes the extensible options for marketing budgets.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

**Extensibility: Quick Steps**

This section provides a brief overview of the steps to customize the Campaigns work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand the Marketing Budget Version 2 node.

**Note**

If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Budgeting work area to view your changes.

**Understanding Which Marketing Budget Pages Are Extensible**

To customize marketing budget pages, you need to 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 new object records
- Detail pages show additional details about an individual object record

The following table lists marketing budget 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 Budget object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Budget Configuration</th>
<th>Marketing Budget UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Budgets, Overview summary table</td>
</tr>
<tr>
<td>Edit Creation Page</td>
<td>Budgets, Create Budget creation page</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Budgets, Edit Budget page</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Manage Budgets page by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**

To make only minor user interface changes to Marketing Budget pages without creating new objects or fields, use Page Composer instead of the Application Composer.

**Adding or Modifying Fields**

To add custom fields to the marketing budget regions listed above, first create your custom fields by selecting the Fields node under the Marketing Budget object. Then, select either a standard field to modify, or create a custom field.

**Adding Your Changes to the Runtime Application**

To add custom fields to the Budget 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 desired Budget regions. You access the configuration pages in the Application Composer from the **Pages** node under the **Marketing Budget** object.

To access the Pages Overview page:

1. Select the **Marketing** application on the main Overview page.
2. In the object tree, select the **Marketing Budget Version 2** object.
3. Select the **Pages** node.
4. On the Pages Overview page, select the configuration page hyperlink related to the Budget region that you want to customize.
Customizing Marketing Budget Entry Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Budget Entry object, which you then add for display in the Budgeting work area. This topic describes the extensible options for marketing budget entries.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Campaigns work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand the Marketing Budget Entry Version 2 node.

   Note
   If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Budgeting work area to view your changes.

Understanding Which Marketing Budget Entry Pages Are Extensible

To customize marketing budget entry pages, you need to 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 new object records
- Detail pages show additional details about an individual object record

The following table lists marketing budget entry 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 Budget Entry object in the Application Composer and create your custom fields.
Next, expose those custom fields on the Manage Budgets page by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**

To make only minor user interface changes to Marketing Budget Entry pages without creating new objects or fields, use Page Composer instead of the Application Composer.

**Adding or Modifying Fields**

To add custom fields to the marketing budget entry regions listed above, first create your custom fields by selecting the Fields node under the Marketing Budget Entry object. Then, select either a standard field to modify, or create a custom field.

**Adding Your Changes to the Runtime Application**

To add custom fields to the budget entry 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 desired budget entry regions. You access the configuration pages in the Application Composer from the **Pages** node under the **Marketing Budget Entry Version 2** object.

To access the Pages Overview page:

1. Select the **Marketing** application on the main Overview page.
2. In the object tree, select the **Marketing Budget Entry Version 2** object.
3. Select the **Pages** node.
4. On the Pages Overview page, select the configuration page hyperlink related to the Budget Entry region that you want to customize.

**Customizing Marketing Fund Request Pages Using Application Composer: Explained**

Use the Application Composer to create custom fields for the Marketing Fund Request object, which you then add for display in the Budgeting work area. This topic describes the extensible options for marketing find requests.
Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Fund Request work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand the Marketing Fund Request Version 2 node.

Note

If you can't see the choice list of applications, make sure the regional area of the page is exposed.

2. Choose the Fields node to create custom fields, or modify standard fields.

3. Choose the Pages node to show or hide fields, either custom or standard.

4. Test your work by navigating to the marketing fund request work area to view your changes.

Understanding Which Marketing Fund Request Pages Are Extensible

To customize marketing fund request pages, you need to 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 new object records
- Detail pages show additional details about an individual object record

The following table lists marketing fund request 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 Fund Request object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Fund Request Configuration</th>
<th>Marketing Fund Request UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Budgets, Manage Fund Requests summary table</td>
</tr>
<tr>
<td>Edit Creation page</td>
<td>Create Fund Request creation page</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Budgets, Edit Fund Request page</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Fund Request page by accessing the appropriate Application Composer configuration page, listed in the following section.
Note
To make only minor user interface changes to Marketing Fund Request pages without creating new objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields
To add custom fields to the marketing fund request regions listed above, first create your custom fields by selecting the Fields node under the Marketing Fund Request object. Then, select either a standard field to modify, or create a custom field.

Adding Your Changes to the Runtime Application
To add custom fields to the marketing fund request 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 desired marketing fund request regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Fund Request object.

To access the Pages Overview page:
1. Select the Marketing application on the main Overview page.
2. In the object tree, select the Marketing Fund Request Version 2 object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page hyperlink related to the Marketing Fund Request region that you want to customize.

Customizing Marketing Claim Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Claim object, which you then add for display in the Budgeting work area. This topic describes the extensible options for marketing claims.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps
This section provides a brief overview of the steps to customize the Marketing Claim work area.

Follow these steps:
1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand the Marketing Claim Version 2 node.
Note

If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the marketing claim work area to view your changes.

Understanding Which Marketing Claim Pages Are Extensible

To customize marketing claim pages, you need to 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 new object records
- Detail pages show additional details about an individual object record

The following table lists marketing claim 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 Claim object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Claim Configuration</th>
<th>Marketing Claim UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Summary Table overview page</td>
<td>Budgets, Manage Claims summary table</td>
</tr>
<tr>
<td>Edit Creation page</td>
<td>Create Claim creation page</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Budgets, Edit Claim page</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing Claims page by accessing the appropriate Application Composer configuration page, listed in the following section.

Note

To make only minor user interface changes to Marketing Claim pages without creating new objects or fields, use Page Composer instead of the Application Composer.

Adding or Modifying Fields

To add custom fields to the marketing claim pages listed above, first create your custom fields by selecting the Fields node under the Marketing Claim object. Then, select either a standard field to modify, or create a custom field.
Adding Your Changes to the Runtime Application

To add custom fields to the Marketing Claim page 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 desired marketing claim pages. You access the configuration pages in the Application Composer from the Pages node under the Marketing Claim object.

To access the Pages Overview page:

1. Select the Marketing application on the main Overview page.
2. In the object tree, select the Marketing Claim Version 2 object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page hyperlink related to the marketing claim region that you want to customize.

Customizing Marketing Claim Settlement Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the Marketing Claim Settlement object, which you then add for display in the Budgeting work area. This topic describes the extensible options for marketing claim settlements.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the Marketing Claim Settlement work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand the Marketing Claim Settlement Version 2 node.

   **Note**

   If you can’t see the choice list of applications, make sure the regional area of the page is exposed.

2. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the marketing claim settlement work area to view your changes.

Understanding Which Marketing Claim Settlement Pages Are Extensible

To customize marketing claim settlement pages, you need to 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 new object records
- Detail pages show additional details about an individual object record

The following table lists marketing claim settlement 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 Claim Settlement object in the Application Composer and create your custom fields.

<table>
<thead>
<tr>
<th>Marketing Claim Settlement Configuration</th>
<th>Marketing Claim Settlement UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit Creation page</td>
<td>Create Claim creation page</td>
</tr>
<tr>
<td>Edit Summary Form details page</td>
<td>Budgets, Edit Claim, Settlement tab</td>
</tr>
</tbody>
</table>

Next, expose those custom fields on the Marketing Claim Settlement page by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**

To make only minor user interface changes to Marketing Claim Settlement pages without creating new objects or fields, use Page Composer instead of the Application Composer.

**Adding or Modifying Fields**

To add custom fields to the marketing claim regions listed above, first create your custom fields by selecting the Fields node under the Marketing Claim Settlement object. Then, select either a standard field to modify, or create a custom field.

**Adding Your Changes to the Runtime Application**

To add custom fields to the Marketing Claim Settlement 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 desired claim settlement regions. You access the configuration pages in the Application Composer from the Pages node under the Marketing Claim Settlement object.

To access the Pages Overview page:

1. Select the Marketing application on the main Overview page.
2. In the object tree, select the Marketing Claim Settlement Version 2 object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page hyperlink related to the marketing claim settlement region that you want to customize.

**Customizing Sales Campaigns in Oracle Sales Cloud Marketing: Explained**

Use Application Composer to create custom fields for the sales campaign object, which you then add for display in the Sales Campaign work area. This topic describes the customization options for sales campaigns. Note that the fields and sections modified apply to mini campaigns as well as sales campaigns. To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

**Extensibility: Quick Steps**

This section provides a brief overview of the steps to customize the sales campaign work area.

Follow these steps:

1. Navigate to Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then select Sales Campaign.
2. Choose the Fields node to create custom fields, or modify standard fields.
3. Choose the Pages node to show or hide fields, either custom or standard.
4. Test your work by navigating to the Sales Campaign work area to view your changes.

**Understanding Which Sales Campaign Pages Are Extensible**

You can extend only the region at the top of the Sales Campaign page. 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.

Depending on the sales campaign status, the fields that you add to this region might or might not be editable. The following list shows which regions and 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>
The following screen shot shows some of the page sections that can be customized. The regions that you can customize are indicated by red boxes.

Adding or Modifying Object Fields

To add custom fields to the Sales Campaign regions listed above, 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
• Updateable
• Depends On

Adding Your Changes to the Run Time Application

After you’ve created custom fields for the sales campaign object, use Application Composer’s configuration pages to add those custom fields to the extensible regions listed above. 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 Marketing application on the main Overview page.
2. In the object tree, select the Sales Campaign object.
3. Select the Pages node.
4. On the Enterprise Pages tab, select the configuration page hyperlink 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 cannot 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.
Customizing Sales Lead Pages Using Application Composer: Explained

Use the Application Composer to create custom fields for the marketing 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.

Access Application Composer from any Sales Cloud service at run time by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.

The marketing objects that are associated with the lead management application are:

- Sales Leads
- Sales Lead Contacts (child object of Sales Lead)
- Sales Lead Resources (child object of Sales Lead)
- Marketing Response (related object of Sales Lead)

Extensibility: Quick Steps

This section provides a brief overview of the steps to customize the sales lead work area.

Follow these steps:

1. Navigate to the Application Composer, select Marketing as the Application, expand the Standard Objects tree node, then expand Sales Lead, Sales Lead Contacts, 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. Choose the Fields node to create custom fields, or modify standard fields.
   3. Choose 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 need to 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 new 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.

<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>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>

Next, expose those custom fields on the Sales Lead page by accessing the appropriate Application Composer configuration page listed in the following section.

**Note**

To make only minor user interface changes to Sales Leads pages without creating new 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 resources object. Then, select either a standard field to modify, or create a
custom field.

**Adding Your Changes to the Runtime 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 desired Sales Leads regions. 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 Marketing application on the main Overview page
2. In the object tree, select the Sales Lead object.
3. Select the Pages node.
4. On the Pages Overview page, select the configuration page hyperlink
   related to the Sales Lead region that you want to customize.

**Creating a Sales Lead Validation Rule Using Application Composer: 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 might want to create business rules to
control when a lead’s status can actually be changed to Qualified.

To do this, use Application Composer to create validation rules to enforce certain
criteria before allowing a sales lead status to be changed to Qualified.

In this example, you will use Application Composer to create a validation rule
that ensures a primary product and sell-to address exist, before a lead’s status
can be changed to Qualified.

Create a validation rule for Sell-to Address

1. Go to Navigator, Application Composer.
2. From the Application field, select Marketing.
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 to go to the Create Object Function screen.
5. In the Function Name field, enter the following name without spaces:
   isSellToExists In the Returns field, select Boolean.
6. In the Function Body area enter the following:
   - 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;

7. Click Validate.

8. Click Save and Close.

Create a rule for enforcing the Primary Product association for the sales lead

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 screen.

2. In the Function Name field, enter the following name without spaces: isPrimaryProductAssigned. In the Returns field, select Boolean.

3. In the Function Body area enter the following:
   if((null != PrimaryInventoryOrgId && null != PrimaryInventoryItemId) || null != PrimaryProductGroupId) return true; else return false;

4. Click Validate.

5. Click Save and Close to return to the Server Scripts Sales Lead screen.

Create a validation rule to check for Lead qualification

1. Click the Validation Rules tab and from the Object Rules area, click the Add a new validation rule icon.

2. In the Create Object Validation Rule screen, go to the Rule Name field and enter the following rule name without spaces: qualifyLead

3. In the rule definition area, enter the following:
   if(isAttributeChanged('StatusCode') && getAttribute('StatusCode') == 'QUALIFIED') return (isPrimaryProductAssigned() && isSellToExists()); else return true;

4. Go to the Error Message section and enter the following message text: Primary Product and Sell to Address are required for qualifying a Lead.

5. Click Save and Close to complete the task of using Application Composer to create validation rules for the primary product and sell-to address fields when setting the lead to qualified.

Test the rule

1. Go to Navigator, Lead Qualification, Edit Lead page.

2. From the Actions menu, select Qualify.

   If the Primary Product and Sell to Address fields contain no data, you should receive the following error message text:
Primary Product and Sell to Address are required for qualifying a Lead.
Oracle Sales Cloud Quota Management Extensibility

Oracle Sales Cloud Quota Management Extensibility: Overview

Read this chapter to understand how you customize and extend Oracle Sales Cloud Quota Management using Application Composer.

In this chapter, you will learn:

- 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 and objects, which you then add for display in the runtime Quota Management application. To access Application Composer, select Application Composer from the Navigator menu, under the Tools > Customization category.

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.

The Oracle Territories and Quotas object that is associated with Sales Quota pages is Sales Goal.

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>Manage Sales Goal page</td>
<td>Search region &amp; Search Results region</td>
<td>Territories and Quotas</td>
<td>Sales Goal</td>
</tr>
</tbody>
</table>
Next, expose those custom fields on the Products tab by accessing the appropriate Application Composer configuration page, listed in the following section.

**Note**

To make only minor user interface changes to Sales Quota pages without creating new objects or fields, use Page Composer instead of Application Composer.

**Adding Your Changes to the Runtime Application**

To add custom fields to the Sales Quota 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 Sales Quota regions. You access the configuration pages in Application Composer from the Pages node under the Sales Goal object.

To access the Pages Overview page:

1. Select the **Territories and Quotas** application on the main Overview page.
2. In the object tree, select the Sales Goal object.
3. Select the **Pages** node.
4. On the Pages Overview page, select the configuration page hyperlink related to the Sales Quota region that you want to customize.

This table indicates which Sales objects populate which Sales Quota 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>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 Products subtab row, and click the Edit icon to navigate to the Edit Subtab: Child or Related Object page.</td>
<td>Edit Sales Goal page</td>
<td>Edit Sales Goal, Products tab</td>
</tr>
</tbody>
</table>
Oracle Sales Cloud Partner Management Extensibility

Oracle Fusion Partner Relationship Management Extensibility: Overview

Read this chapter to learn about how to extend the pages that belong to Oracle Fusion Partner Relationship Management (PRM).

In this chapter, you will learn about how to use Page Composer to:

- Customize internal-facing PRM pages
- Customize external-facing PRM pages
- Customize the Partner Portal UI shell

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 > Customization category.

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.

**Tip**
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 &gt; Edit Summary Table link</td>
</tr>
<tr>
<td>Partner Management Page</td>
<td>Partner Management Region</td>
<td>Underlying Business Object</td>
<td>Configuration Page in Application Composer</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------</td>
<td>---------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>New Partner Registration: Partner Information page</td>
<td>Company Details region</td>
<td>Partner</td>
<td>Partner: Pages node &gt; 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 &gt; Profile: Profile Header Form, Profile: Profile Key Details Form, and Profile: Profile Additional Information Form tree node links</td>
</tr>
<tr>
<td>Partner Center Profile Public Profile page</td>
<td>Entry and Preview regions</td>
<td>Partner</td>
<td>Partner: Pages node &gt; Public Profile: Public Profile Entry Form, Public Profile: Public Profile Preview 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 &gt; 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 &gt; 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 &gt; Profile: Profile Header Form tree node link</td>
</tr>
<tr>
<td>Partner Center Enrollments page</td>
<td>Summary table region</td>
<td>Program Enrollments</td>
<td>Partner: Pages node &gt; 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 &gt; 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 &gt; 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 &gt; Edit Summary Table link</td>
</tr>
</tbody>
</table>
Oracle Fusion Partner Management (PRM) consists of external facing pages, for partner users, and internal facing pages for employee users. Several external facing pages must be customizable to allow brand owners to tailor the presentation and content to the specific needs of the external user.

In PRM, by using Page Composer, you can customize any of the following pages:

- **External pages**
  - Partner Dashboard
  - Edit Partner Profile
  - Edit Partner Public Profile
  - Edit Personal Profile
  - Partner Landing
  - Partner Registration Landing
  - Partner Registration: Partner Information
  - Partner Registration: Review and Accept Terms
  - Partner Registration Confirmation

- **Internal Pages**
  - Channel Dashboard
  - Partner Snapshot
  - Edit Partner Profile
  - Edit Partner Public Profile

### Customizing External Facing PRM Pages

To customize eligible external facing pages, you must have Channel Partner Portal Administrator privileges.

To access the Partner dashboard:

1. Navigate to the dashboard page.
2. Select Administration Customize Workarea Pages....
Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Direct Selection customization modes. You can do the following customizations:

- Change the local area layout. For example, change a two-column layout to a three-column layout (eight layouts are available).
- 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 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, change display and style options.
- Add, remove, and edit ADF components to or from dashboards, for example: regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

To customize the remaining external partner pages (Edit Partner Profile, Edit Partner Public Profile, Edit Personal Profile, Partner Landing, and Partner Registration Landing), use the following steps:

1. Navigate to the Partner dashboard.
2. From the Administration menu at the top of the page, click Customize Workarea Pages...
   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.
   The Partner Registration Landing Page opens.
2. Click Register Your Company as a New Partner.
   The partner registration page opens.

Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Direct Selection customization modes. You can do 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 ADF components to or from dashboards, such as regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

**Customizing Internal Facing PRM Pages**

To customize the Channel dashboard or any of the eligible 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. Select Administration Customize Workarea Pages.

Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Direct Selection customization modes. You can do 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 (by moving the page splitter location).
- 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, and change display and style options.
- Add, edit, or remove ADF components to or from dashboards, for example, regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

To customize the Edit Partner Profile, Partner Snapshot, and Edit Partner Public Profile 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 and the Edit Partner Public Profile pages. You can launch Page Composer in any of these pages by selecting Administration, Customize Workarea Pages.

Customizations are available at the Site, External or Internal, and Job Role layers, in either Design and Direct Selection customization modes. You can do 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, change display and style options.
- Add, edit, or remove ADF components to or from dashboards, such as regions, hyperlinks, images, text boxes, movable boxes, and Web pages.

Customizing the Partner Portal UI Shell
This customization workflow allows a brand owner to customize the standard Oracle Fusion user interface shell for the application’s external facing pages. Select the External option to make changes at the Internal or External MDS layer. To access and execute this flow, you must be assigned the Channel Partner Portal Administrator job role, which has the Partner Portal Customize Links Duty role. The administrator can customize the user interface shell by:=
1. Select the regional task list, **Update Partner Portal UI Shell**.
This customization task requires the use of Page Composer's Source View, which is enabled only for this task in Oracle Sales Cloud.

Application Composer supports the following user interface shell and branding customization supported tasks:

- Modify the user interface shell header. For example, you can add new content, hide or show global hyperlinks, and hide, show, or add menu items.
- Modify the user interface shell footer, such as add, edit, or hide 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.

### Customizing Partner Relationship Management Using Page Composer: Worked Example

Administrators can change the UI shell branding for external users. These customizations are visible to partner users who access the partner portal and to any user who accesses anonymous partner pages, such as the Partner Landing and Partner Registration pages.

This example covers the following customization tasks:

- Replacing the standard Oracle Fusion logo with a new logo.
- Changing the company name in the header area text.
- Adding a link to an external Web site in the global area.
- Hiding a global link.
- Hiding a menu item link.

#### Customizing the Partner Relationship Management UI Shell for External Users

1. Sign in as the Channel Partner Portal Administrator job role, and navigate to the Partners dashboard.

2. Before making changes using Page Composer, confirm that you are working in the correct sandbox. If not, create a sandbox before making changes.

3. In the Tasks region, click **Update Partner Portal UI Shell**.
   The UI Shell Template customization workflow runs and the Partner dashboard page opens in the Page Composer design edit mode.

4. Open the Source edit mode by selecting **View - Source**.

   **Tip**
   By default, the Source window is collapsed in the top position. To change the Source window position, select **View - Source Position**. To see the source content, you might have to drag the page splitter.

   You are now ready to perform any of the customization tasks in the following sections.
Changing the Shell Logo

1. Click the logo field in the top-left region.
   The source tree moves to the corresponding spacer component.

2. Click Edit.

3. In the Component Properties dialog box, select the Style tab.

4. Click the arrow next to the Background Image field.

5. Enter the URL for the new image file location.

   **Note**
   The new image file should be located in a server and must be accessible through a URL. The recommended image size is 119px X 25px (width by height) for best results, so as to replace the Oracle logo. If it is not this size already, then you can set the new image to this size in the Other CSS field. Enter the value as "background-size:119px 25px;" to set the image to the correct size.

   The following figure shows editing the UI Shell logo image URL.

   ![Editing UI Shell logo image URL](image)

6. Click Display Options and modify the Short Desc value with a description.

   **Tip**
   This value appears when you mouse-over the logo.
7. Click **OK**.

The new logo image appears in the UI.

**Changing the Company Name in the Header**

1. In the global region, click the **Fusion Applications** text box near the logo.
2. Click **Edit** in the Source window toolbar.
3. In the Component Properties window, select **Display Options**.
4. Click the arrow next to the **Value** text box.

The following figure illustrates the Component Properties window.

5. Click the expression builder.
6. Select **Type a value or expression** and enter the new company name.
7. Click **OK** to sign out of the expression builder.
8. Click **OK** to sign out of the **Component Properties** window.

**Adding an External Link to the Shell Global Area**

1. Click the links in the global region.

   The source tree moves to the panelGroupLayout:horizontal component.
2. Click the **Add Content** source window toolbar.

   The Add Content dialog box opens.
3. Click **ADF Faces Components**.
4. Click the **Hyperlink** component.
5. Click the **Add** icon.

   The new link appears on the page labeled as "Click edit to define hyperlink".

6. Click the hyperlink and then **Edit** in the Source window toolbar.
7. Click **Display Options**.
8. Enter the URL in the **Destination** field.
9. Click the arrow next to the Text box to enter the label for the new hyperlink. You can enter the label through a resource bundle (if available), the expression builder, or by overriding the existing label.

   Expression Builder and Override are only supported on dashboard pages.
10. Click **OK** to sign out of the Component Properties dialog box.

   The new hyperlink should now appear in the UI shell global area.

**Hiding a Link in the Global Area**

1. In Source View mode, select a global link to hide, for example, the Personalization link.

   The Source tree moves to the selected item.
2. Click **Edit** in the Source window toolbar.
3. In the Component Properties dialog box, deselect the **Show Component** check box.
4. Click **OK**.

   The hyperlink or menu link should no longer appear in the UI shell global area.

**Validating Your Customizations**

1. Click **Close** to sign out of the Page Composer editor.
2. Sign out of Oracle Applications.
3. Access the external landing pages as an anonymous user, or access the Partner dashboard with any external job role.
4. Verify your customizations:
   - The new logo and company name are displayed.
   - The new link is displayed in the global area.
5. When you are ready to commit your customizations to the back end, publish your sandbox according to your organization’s established practices.
Oracle Sales Cloud Mobile Sales Extensibility

Oracle Sales Cloud Mobile Sales Extensibility: Overview

Read this chapter to learn about how to extend Oracle Sales Cloud Mobile Sales.

In this chapter, you will learn about:

• How to use Application Composer to manage which objects and fields are visible on the Mobile Sales application without having to do specific customizations for any particular device
• How to add an Oracle Business Intelligence report to Mobile Sales
• How to test Mobile Sales customizations using a sandbox

Oracle Sales Cloud Mobile Extensibility: Explained

Application Composer lets implementors customize the Oracle Sales Cloud Mobile Sales iPhone 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 do specific customizations for any particular device.

Implementors can manage the following for the Oracle Sales Cloud Mobile application:

• Enable standard Oracle Sales Cloud Sales, Customer Center, Marketing, and Common objects that are not enabled by default for smartphones.
• Enable custom Sales, Customer Center, Marketing, and Common objects for smartphones.
• Change the fields (including custom fields) visible on Oracle Sales Cloud Mobile for mobile-enabled Sales, Customer Center, Marketing, and Common objects (standard or custom objects).
Customizing Oracle Sales Cloud Mobile Sales: Worked Example

This example shows you how to customize Oracle Sales Cloud Mobile Sales for a specified sales object, using Application Composer. For more details about Application Composer, refer to the Oracle Sales Cloud: Extending Sales Guide.

1. Open Application Composer by using the Navigator menu, and selecting Application Composer under the Tools > Customization category.
2. Select the application you want to customize within Application Composer.
3. Within the application you chose in step 1, select the parent 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.

How can I add an Oracle Business Intelligence report to Oracle Sales Cloud Mobile Sales?

Navigate to the Application Composer, select the Sales application, and select Mobile Pages under the Common Setup list. In the Mobile Pages page, select Manage Mobile Reports and create the report, entering the Oracle Business Intelligence Analyses report details. Finally, add the report to the Mobile Reports Springboard page, or the Mobile Reports Sales Account page, by selecting either Configure Mobile Reports: Springboard or Configure Mobile Reports: Sales Account.

Note that you can add Oracle Business Intelligence Analyses reports, but you cannot add Oracle Business Intelligence Publisher reports.

Testing Oracle Sales Cloud Mobile Customizations Using a Sandbox: Worked Example

The following steps illustrate how to test Oracle Sales Cloud Mobile customizations using a sandbox. Sandboxes are standalone environments where you can define and test customizations, before deploying the customizations to the main Oracle Sales Cloud application. It is recommended that you test all of your customizations in a sandbox before publishing them to the main application.

In this topic you will use the Application Composer to customize Oracle Sales Cloud Mobile pages or objects in a sandbox environment, and then view your customizations on your smartphone prior to publishing the changes.
Open a Sandbox

1. Log 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 which is available next to your user name. Then select **Manage Sandboxes**, under the Administration subheading.

3. Select the sandbox in which you wish to make your customizations. You may need to make a sandbox active, or create a sandbox, if a suitable sandbox does not exist. Refer to the Setting Up Sandboxes chapter of the Oracle Fusion Applications Extensibility Guide for Business Analysts for more information about creating sandboxes.

Configure Oracle Sales Cloud Mobile Using the Application Composer

1. Open the Application Composer by selecting **Application Composer** under the Tools > Customization category in the Navigator menu.

2. Select the application you want to customize within Application Composer, and then select the parent object you want to configure.

3. Select the **Pages** node in the navigation tree, and then select the Mobile Pages tab to see the mobile configuration options for the parent and its child objects.

4. Configure the mobile pages as desired and log out of the Oracle Sales Cloud application.

Check Your Customizations in the Oracle Sales Cloud Mobile Application

1. Log in to Oracle Sales Cloud application as an Oracle Sales Cloud Mobile user that 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 smartphone and log in using the same user you used to log in to Oracle Sales Cloud. Logging in as the same user in step 1 enables you to view the sandbox you selected in step 2 on your smartphone’s Oracle Sales Cloud Mobile application. Note that only your user is accessing the sandbox on the Oracle Sales Cloud Mobile application (as long as the Oracle Sales Cloud browser window is open); all other users will view only the published version of the application.

4. Check the pages you have customized to ensure that they are working as expected.

5. To distribute your customizations to all Oracle Sales Cloud Mobile users you will need to publish your sandbox. Refer to the Publishing Sandboxes chapter of the Oracle Sales Cloud: Extending Sales Guide for more information about publishing sandboxes.
Exporting and Importing Customizations

Exporting and Importing Customizations: Overview

Read this chapter to learn about how to export and import customizations across Oracle application instances on the same release.

In this chapter, you will learn about:

- How to use customization migration to move customizations
- How to create and apply customizations using customization set

Note

You can also import and export custom object transactional data as opposed to metadata. Two processes exist to enable the importing and exporting of object data: file-based import and bulk export. After you create custom objects in Application Composer, you must then manually generate the object artifacts required for both file-based import and bulk export. This process, which is unrelated to the exporting and importing of customizations, is described in a separate topic, "Importing and Exporting Custom Objects: Explained."

To access customization set migration (CSM) in the source or target environment, select Navigator - Tools - Customization Migration. CSM is accessible as a link called Customization Migration under the Tools menu item.

Exporting and Moving Customizations: Points to Consider

Customizations are stored in XML files. This enables you to export customizations for the following reasons:

- To move customizations and extensions to another Oracle Fusion Applications environment, such as the production environment.
- To diagnose issues noticed in the test environment.
- To send files to Oracle Support Services for further diagnosing.
- To import a customization into another environment. For example, a customization developer using JDeveloper might need to see customizations done by someone else.
Exporting and importing customization files also help in backing up and restoring customizations.

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 Oracle Fusion Applications environment.</td>
<td>Customization Set Migration.</td>
</tr>
<tr>
<td>Move only MDS customizations made to pages and the user interface to another Oracle Fusion Applications environment.</td>
<td>Oracle Enterprise Manager Fusion Applications Control (Fusion Applications Control).</td>
</tr>
<tr>
<td>Note</td>
<td>You can also use Fusion Applications Control to download and upload a set of customizations.</td>
</tr>
<tr>
<td>Move only descriptive flexfield configurations to another Oracle Fusion Applications environment.</td>
<td>Oracle Fusion Functional Setup Manager (Functional Setup Manager). Moves configurations for a specified module. To move configurations for all modules, use Customization Set Migration.</td>
</tr>
<tr>
<td>Move only extensible flexfield configurations to another Oracle Fusion Applications environment.</td>
<td>Functional Setup Manager. Moves configurations for a specified module. To move configurations for all modules, use Customization Set Migration.</td>
</tr>
<tr>
<td>Move only value set configurations to another Oracle Fusion Applications environment.</td>
<td>Functional Setup Manager. Moves configurations for a specified module. To move configurations for all modules, use Customization Set Migration.</td>
</tr>
<tr>
<td>Move only lookups to another Oracle Fusion Applications environment.</td>
<td>Functional Setup Manager. Move application standard lookups, application common lookups, or both.</td>
</tr>
<tr>
<td>Move only data security policies to another Oracle Fusion Applications environment.</td>
<td>Functional Setup Manager.</td>
</tr>
<tr>
<td>Note</td>
<td>It does not 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 JDeveloper.</td>
<td>Manage Customizations dialog box.</td>
</tr>
</tbody>
</table>

**Note**

Fusion Applications Control is not available in Oracle Cloud implementations. Therefore, in Oracle Cloud implementations, to perform tasks that require use of Fusion Applications Control, log a service request using My Oracle Support at https://support.oracle.com.
Downloading Customizations

On the Manage Customizations dialog box, from the Administration - Manage Customizations menu, you can download customization files for a given page. You can use these files for diagnosing customization issues.

You can also download all customizations of the page for all layers (AllCustomization.zip), using the Download Customizations for All Layers link located at the bottom of the dialog box. The file contains all the customization XML files for the page.

Using Customization Migration to Move Customizations: Points to Consider

The Customization Migration page enables you to create a set of all customizations and extensions that have been made to an Oracle Fusion Applications environment, download that customization set, and upload it into another environment. The customization set includes customizations across all Oracle Fusion Applications product families, such as Oracle Fusion Financials, Oracle Sales Cloud, and Oracle Fusion Human Capital Management (Oracle Fusion HCM). The customization set includes only the customizations and extensions that you make using the following tools and features. Personalizations are not included in the set.

To access the Customization Migration page, navigate to Tools - Customization Migration from the Navigator menu.

Contents of the Customization Set

The customization set includes only the customizations that you make using the following tools, features, or tasks:

- Application Composer, with the exception of 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.
- Page Composer
- Content developed using Oracle Business Intelligence Enterprise Edition features including but not limited to Oracle Business Intelligence Answers, Oracle Business Intelligence Delivers, Oracle BI Composer, and Oracle Business Intelligence Interactive Dashboards.

Note
You can move these customizations only if the Disable BI for CSM profile option is set to No.

- Tasks and dialog boxes for configuring descriptive and extensible flexfields and value sets
- Manage Menu Customizations task
- Manage Oracle Social Network Objects task
- Manage Standard Lookups task
- Manage Common Lookups task
- Security settings and changes made in the Application composer including associating privileges to duty roles, which provide access to custom objects are migrated and overwritten to the target. Enterprise roles, new duty roles, and role hierarchy changes, which are made directly in Oracle Authorization Policy Manager (Oracle APM) will not be migrated.
- For non-Cloud implementations, customization metadata that is created from JDeveloper using the Oracle Fusion Applications Administrator Customization role and then packaged and deployed to the source Oracle Fusion Applications environment.

**Warning**

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

- Reports that were submitted by Oracle Enterprise Scheduler to Oracle BI Publisher and were scheduled to execute during the process, will fail.
- The Reports and Analytics pane does not display.
- Oracle Business Intelligence Publisher reports may not display on Oracle Business Intelligence Presentation Services analyses or dashboard pages.
- Users may not be able to access Oracle Business Intelligence Enterprise Edition features, such as Oracle Business Intelligence Answers, Oracle Business Intelligence Delivers, Oracle BI Composer, and Oracle Business Intelligence Interactive Dashboards.

**Note**

The Customization does not include code extensions, such as managed beans, that you implement in JDeveloper using the Oracle Fusion 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**

The Customization Migration page preselects the type of customizations across all applications that will be added to the customization set. You cannot change this selection.

**Tip**

To prevent in-progress customizations from getting included in the customization set, make your customizations in a sandbox. The customization set does not include customizations that are in sandboxes.
While you can use the Customization Migration page to move customizations and extensions from any source environment to any target environment, you should always perform your customizations and extensions in a full test environment and use the Customization Migration page to move these changes to a production environment. As customization set migration does not 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.

Note

The customization set does not 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 Oracle APM have to be manually updated in the target environment.

Creating and Applying Customizations Using a Customization Set: Procedures

Create a customization set to move customizations across all the product families of Oracle Fusion Applications from one environment to another environment. Creating a customization set is beneficial in moving customization in a batch instead of moving customizations one by one.

You need to ensure the following before creating a customization set:

- The source and target Oracle Fusion Application environments are of the same release and that the same standard and one-off patches have been applied to both environments.
- All Application Composer, Page Compose, and JDeveloper customizations made in sandboxes are complete before they are published. All complete customizations must be published before the export process begins.
- All customizations and extensions made using the Manage Menu Customizations task, the Manage Standard Lookups task, and the Oracle Authorization Policy Manager, are complete.
- To move content that was created using Oracle Business Intelligence Enterprise Edition features, ensure that the Disable BI for CSM profile option is set to No in both the source and target environments. To view this profile option, access the Setup and Maintenance work area in the global area of Oracle Fusion Applications and search for the Manage Profile Options task.
- You have been granted access to the FND_CUSTOMIZATION_SET_MANAGEMENT_DUTY role, which enables you to access the Customization Migration page. Contact your security administrator for details.
• Users never make customizations in the target or production environment while applying customizations.

Note
If users must make customizations to the production environment in emergency circumstances, remember to make the same customizations to the test environment to ensure that they are included in the next customization migration.

• Users do not perform customizations in the source environment during the export process.

To create customizations:

1. In the source environment, from the Navigator menu, select Tools - Customization Migration.
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 removes the temporary files that are on the server from the previous customization set creation. You will not be able to 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.

Tip
To see the detailed status of each customization type, expand Customization Details.

Note
The process runs asynchronously. You can exit the dialog box and return to it at a later time.

7. Click Download and specify the name and location for the file that will be created (ensure that the downloaded file is a JAR file), and click Save.
8. After the file is successfully downloaded on your local file system, click Delete to remove the temporary files that were created on the server.

To apply customization to the target environment:

Tip
Apply customizations when a few people are signed into the environment, because end users must sign out and sign in again to see any changes that are made.

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**.

**Note**
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.

**Tip**
The process runs asynchronously. You can exit the dialog box and return to it at a later time.

**Note**
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 Oracle Social Network definitions to the Oracle Social Network server:
   a. In the Setup and Maintenance work area, access the Manage Oracle Social Network Objects task.
   b. For an object that was created or updated as part of the applying customizations process, if its Enabled value is anything other than **No**, trigger the process of sending its definition to the Oracle 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**, disable the object, enable the object, and select the value **Manual**.
   c. Click **OK** and save the changes.

8. If you applied Oracle Sales Cloud application customizations, you must perform the following tasks from Application Composer to complete the movement of Oracle Sales Cloud customizations and extensions:
   - Generate the artifacts that are required to register the migrated extensions. Click **Import and Export** in the Common Setup region
and click **Generate**. This makes the migrated extensions available for importing and exporting.

- From the Common Setup region, click Web Services and complete the following substeps for every secured SOAP web service connection that uses either the **Invoke with separate user credentials over SSL** or the **Invoke with separate user credentials and message protection** authentication schemes to manually migrate the user names and passwords:

  1. Make a note of the name and WSDL URL for the web service and delete the web service.
  2. Click **Create** to recreate the secured web service.
  3. In the Create SOAP Web Service Connection dialog box, type the name and WSDL URL for the web service. The details must match with those of the web service that you deleted.
  4. Click **Read WSDL**.
  5. Ensure that the appropriate authentication scheme is selected.
  6. Click **New** for the Credential Key field.
  7. In the Create Key dialog box, type the credential key, user name and password, and click **OK**.
  8. Click **Save and Close**.

  9. If attachments were enabled on run time pages using Applications Composer, complete the following steps to enable attachments for those pages in the target environment:

    a. In the source environment, in the Setup and Maintenance work area, access the Manage Implementation Projects task.
    b. On the Manage Implementation Projects page, click **Actions - Create**.
    c. On the Enter Basic Information page, either accept the default values or replace them with the required values.

---

**Tip**

The following points must be adhered to:

- Provide a meaningful name for the project. The code and description fields change automatically.
- If you modify the Code value, replace it with a unique code.
- Use the default value for the Start Date and leave the Finish Date blank.
- The user in the **Assigned To** field must have access to Customization Set Migration functionality.

  d. Click **Save and Open Project**.
  e. On the Implementation Project page, click **Actions - Select and Add**.
f. Search for the Manage Attachment Categories and Manage Attachment Entities tasks, click **Apply**, and click **Done**.

g. In the Tasks region, click Manage Configuration Packages.

h. On the Manage Configuration Packages page, click **Actions - Create**.

i. In the Source Implementation Project region, select the implementation project that you created.

j. Click Next and on the consequent page, click **Next**.

k. On the Create Configuration Package: Schedule page, click **Submit**. On the warning message box, click **Yes** to proceed with the export process.

---

**Note**

Periodically click **Refresh** to display the status.

---

l. After the export process is complete, download the configuration package.

m. In the target environment, create a similar implementation project and a configuration project and click **Save and Close**.

n. On the Manage Configuration Packages page, select the configuration package that you just created and click **Upload**.

o. Select the configuration package that you downloaded from the source environment and click **Open**.

p. Click Details and click **Submit**.

q. Select the configuration package, click Import Setup Data and click **Submit**.

---

**Note**

Periodically click **Refresh** to display the status.

---

r. After the import is complete, click **Done**.

10. As an optional step, identify and reschedule any Oracle Sales Cloud processes that failed while applying customizations. To do this, from the Navigator menu, select **Tools - Scheduled Processes** and look for processes with the following statuses:

- Blocked
- Error
- Error Auto-Retry
- Error Manual Recovery
- Paused
- Validation Failed
11. Manually migrate all business processes that were created in the source environment to the target environment. Alternatively, you can export them from the Process Composer. To access the Process Composer, navigate to the Application Composer and select the business process that you want to migrate.

12. When the customizations are successfully applied, perform functional testing to verify the changes. If testing exposes problems with the customizations, such as importing more than you intended, or the changes were not what you expected, access the customization set in the Incoming tab of the Customization Migration page, and click Restore to revert to the state before the customization set was applied. In such cases, skip the next step.

**Note**

Users can monitor the progress of the download or the applying process by viewing the process log. This process takes approximately 15 minutes. If it takes any longer and you do not 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.

**Important**

- After an environment upgrade, any previous imports which were performed in an earlier release cannot 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 are not 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: Status field in its source environment has values Open, Closed. In the target environment it has values Yes, No. After the import, the Status field in target environment has values Open, Closed, Yes, and No.

- After the import, perform the following steps in the target environment to send the new and updated Oracle Social Network definitions to the Oracle Social Network server.

  1. In the Setup and Maintenance work area, access the Oracle Social Network Objects task.

  2. 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 are not automatically revoked in the target environment. For example, if a specific privilege is granted in the target environment but the corresponding privilege does not exist in source environment, during import, the privilege in the target environment will not be automatically revoked. To address this 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.

**Note**

- You can create custom reports directly in the target environment. However, ensure that you create the custom reports and reference them to the already existing custom subject areas (do not create the custom subject areas directly in the target environment).

- Customization export and import tasks can only be initiated from the mainline code. If they are initiated from a sandbox, the process does not 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.

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

**Exporting and Importing Supported Customizations: Explained**

Once you have completed creating customizations in Application Composer, you can export and import these customizations 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.

In order 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 Set Migration to Move Customizations” in the Oracle Fusion Applications CRM Extensibility Guide for Business Analysts on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.
Note

In CSM, the changes are directly copied to the mainline and there is no sandbox 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, do the following:

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. Once the customization set is created, click **Download**.

5. Download the customization set to a local folder.


Importing Customizations

Once you have successfully exported the customizations to a compressed file format, you can import this compressed file into the target application instance.

To import customizations, use the following steps:

1. In the target application instance, click **Navigator - Tools - Customization Migration**.

2. The Customization Migration page opens.

3. Click the **Incoming** tab.

4. Enter the name of the customization set.

5. Click **Apply**.

For more information on using CSM, see "Using Customization Set Migration to Move Customizations" in the Oracle Fusion Applications CRM Extensibility
Exporting and Importing Customizations


If you encounter any errors during the import process, (for example, the connection to the database is lost or certain process issues occur during migrating security policies), then all changes roll back automatically, reverting the target application instance to its version before the import.

If you have made any security changes in the source instance outside of Application Composer, ensure that you manually re-key these security changes in the target instance prior to using CSM for importing. For example: If you have set up a custom security role in the source instance, ensure that you manually add this security role in the target instance, prior to using CSM.

**Note**

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 explains the following:

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

**Supported Customizations**

The supported customizations for the 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 the Application Composer include:

• Import and export artifacts generated for custom objects and fields: Regenerate manually using the Import and Export menu option in the Application Composer.

Customizations made outside the 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

Some of the best practices for using CSM are:

• Do not manually create supported customizations in the target.
• Manually create customizations which are partially supported or not supported by CSM (See list above).
• 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 source to target prior to using CSM.

• During an export or import, it is recommended that you do not make customization changes in the source or target instance.
An End-to-End Configuration Example

End-to-End Configuration Examples: Overview

This chapter contains examples of end-to-end configurations that you could do in Oracle Sales Cloud.

In this chapter, you will 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 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.

Important
Before working with sandboxes, see Guidelines for Customizing Fusion CRM using Application Composer and Sandboxes (1484889.1) on My Oracle Support at https://support.oracle.com.

Creating a Sandbox

1. Sign in to Oracle Sales Cloud and click the Administration menu.
2. Click the Manage Sandboxes list item.
   The Manage Sandboxes window appears, listing the available sandboxes in your environment.
3. Click the Actions menu.
4. Click the New list item.
   The Create Sandbox window appears.
5. Enter your initials in the Sandbox Name field.
   Don’t select the Create Data Security Sandbox. You can use this option later if you want to customize security for your object. By selecting this option, you create a special security sandbox where you can edit security policies.
6. Click the **Save and Close** button. When the confirmation window appears, click **OK**.
   Your sandbox is now listed in the Manage Sandboxes window.

7. To activate the sandbox, select the row for 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 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 are working in is a part of your user profile information.

8. The Sandbox Details window appears. 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.

9. Note the **Download All** button on the right. You can use sandboxes to make an extension, test it, and also download and keep all the MDS files that pertain to that extension. This is one way to migrate changes across environments.

10. Click the **Close** button.
    You can exit the sandbox from the details you see when you hover your mouse over the sandbox name.

11. Click the **Exit Sandbox** link. When the Exit Sandbox window appears, click the **Yes** button.
    The sandbox name no longer displays, confirming that you're no longer in the sandbox session.

### Creating or Extending an Object: Example

This example demonstrates how to create or extend an object. Here, an Oracle Sales Cloud administrator at Pinnacle Corporation uses Application Composer to create a simple help request custom object and work area.

#### Creating a Simple Help Request Custom Object and Work Area

1. Sign in to Oracle Sales Cloud and click the **Navigator** link.

2. Click the **Application Composer** link to display the ApplicationComposer home page.
3. Click the **Application** list, then click the **Sales** list item.

   The application lists the custom objects already created for the Sales application.

4. Click the **Create** button. When the Create Custom Object window appears, click in the **Display Label** field.

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>
</table>
| Display Label | This is the name the users will see. 
Press the Tab key after entering your text to populate the rest of the required fields with the text you enter. | Help Request  |
| Plural Label  | The pluralized version of the display label.                           | help requests |
| Object Name   | The name of the object.                                               | XYHelpRequest |
| Record Name   | The name of the field that identifies each instance of the object. In the help request object, this will be the abstract, where users enter a brief summary of the issue. | Abstract      |

6. Press the Tab key.

7. Click the **OK** button.

8. Click the **Expand** button next to the Custom Objects tree item.

   The list of custom objects now contains the help request object you created.

9. Click the **Expand** button next to the Help Request tree item.
The icons represent what you can create for your object including fields, pages, buttons, 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 with the issue.</td>
</tr>
</tbody>
</table>

10. Under Help Request, click the **Fields** link.

    The Fields page displays.

11. You’ll start by creating the Description field. Click the **Create a custom field** button.

    The Select Field Type window appears.

12. Click the **Long Text** option, then click **OK**.

13. Click in the **Display Label** field and enter what the users see as the field name. Enter "Description".

14. Click in the **Display Width** field and enter the field width. Enter "50".

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

    The application returns you to the Fields page and you are ready to create your second field: Type.

16. Click the **Create a custom field** button.

17. Click the **Choice List (Fixed)** option, then click **OK**.

18. Enter "Type" in the Display Label field.

19. 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.

    The Create Lookup Type window appears.

20. Enter the name of the lookup type in the **Meaning** field Use initials to make it unique. Enter "XY Help Request Type".

21. Enter a unique lookup type technical name in the **Lookup Type** field. Enter "XY_HR_TYPE".
You’re now ready to enter the values users will select (Hardware, Software, and Network) in the Lookup Codes region.

22. Click the **Create Lookup Code** button.

23. Enter "Hardware" as the first value in the **Meaning** field.

24. Enter "Hardware" in the **Lookup Code** field.

25. Enter a number representing the place of this lookup in the sequence of values into the **Display Sequence** field. Enter "10".

26. The rest of the entries have been entered for you in this example. Click the **Save** button.

   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.

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

   The application returns you to the Fields page. You’re now ready to create the third field for your object: Status.

28. Click the **Create a custom field** button.

29. Click the **Choice List (Fixed)** option, then click **OK**.

   The Create Fixed Choice List page appears.

30. Enter "Status" as the field name in the **Display Label** field.

31. In the List of Values region, click the **Create a New Lookup Type** button.

   The Create Lookup Type window appears.

32. Enter "XY Help Request Status" as the name for your lookup in the **Meaning** field.

33. Enter "XY_HR_STATUS" as the technical name in the **Lookup Type** field.

34. Now you’ll add the two Status values, Open and Closed, in the Lookup Codes region. Click the **Create Lookup Code** button.

35. Enter the first value the user will see in the **Meaning** field. Enter "Open".

36. Enter the technical value in the **Lookup Code** field. Enter "Open".

37. Open will be the first value you want displayed in the list, so enter "10" in the Display Sequence field.

38. Use the same procedure to enter the second value: Closed, with a display value of 20.

39. Click the **Save** button.

   The application returns you to the Create Fixed Choice List page.

40. In the Default Value region, you will specify Open as the status of the help request when it is created. Click the **Default Value** list.

41. Click the **Open** list item.

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

43. Click the Create a custom field button.

44. Click the Choice List (Dynamic) option, then click OK.

The application displays the Create Dynamic Choice List: Basic Information page.

45. Enter “Company” in the Display Label field.

46. 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 Customer field list of values.

47. Click the Related Object list.

48. Click the Partner list item.

Now you’ll select the partner attribute you want displayed in the list.

49. Click the List Selection Display Value list.

50. Scroll down and click the Partner Name list item.

You could display additional information in the partner list as well by selecting fields in the Additional List Display Values region.

51. Click the Finish button.

The application returns you to the Fields page.

52. Click the Pages tree item.

The application displays the Help Request: Pages page. You’ll begin by creating the work area for your object.

53. Click the Enterprise Pages tab.

54. Click the Create Work Area link.

The application displays the Create Work Area: Configure Navigator Menu page. Here you specify the Navigator menu item that will bring users to this work area.

55. Click the Menu Category list.

56. Click the Sales list item.

57. Click in the Menu Item Display Label field and enter the desired information. Enter “help requests”.

In the Menu Items region, you can change the position of the different menu items within the menu category.

58. Click the Next button.
The application displays the Configure Search page. Here you add the fields you want to be able to search on.

59. Double-click each of the following list items:
   - Abstract
   - CreationDate
   - Status
   - Type

60. Click the Next button.

The application displays the Configure Overview and Creation Pages page. 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.

61. In the Available Fields region, double-click each of the following list items:
   - Company
   - Description
   - CreationDate
   - CreatedBy
   - Status
   - Type

62. In the Configure Creation Page region, select the fields you want to appear in the creation page. Double-click each of the following list items:
   - Company
   - Description
   - CreationDate
   - CreatedBy
   - Status
   - Type

63. Click the Next button.

The application displays the Configure Details Page Summary page.

64. In the Configure Default Summary 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 each of the following list items:
   - Abstract
- Company
- Status
- Type

65. In the Configure Detailed Summary region, select the fields that will appear in the object's detailed summary, which is the expandable region right after the default summary. Double-click the following list items:
- Description
- CreationDate
- CreatedBy

66. Click the **Allow Attachments** option.
67. Click the **Save and Close** button.

Now you'll test your application's work area.

68. Click the **Navigator** link, then click the **help requests** link.

The application displays the help requests work area.

Begin by creating a new help request in the Overview page.

69. Click the **Create** button.

The Create Help Request page appears.

70. Enter "Laptop doesn't work." in the **Abstract** field.
71. Click the **Search: Company** link.
72. Click the **Absolute Technologies** list item and enter "Smoke is coming out of the back." into the **Description** field.
73. Click the **Type** list.
74. Click the **Hardware** list item.
75. Click the **Save** button.

The application returns you to the work area Overview page. You will now review and update the help request you just created.

76. Click the **Laptop doesn't work.** link.

The Edit Help Request page appears showing the request you created. Now you'll add an attachment.

77. Click the **Expand Show Details** button.
78. Click the **Manage Attachments** button.

The Attachments window appears.

79. Click the **Browse** button and select any file to attach, then click **OK**.
80. 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 the Application Composer to:

- Create a Solutions custom object, and set up a context link subtab for it in the Help Request work area
- Create an Activity child object, and add a child subtab for it in the Help Request work area

Creating a Solutions Custom Object

1. Sign in to Oracle Sales Cloud and click the Navigator link.
2. Click the Application Composer link.
   
   Note
   You might need to click the More >> link if you can’t find 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. You’ll make some changes to some of these 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 Updateable 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. At the bottom of the page, click the **Allow Attachments** option.
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.
46. Click the **Hardware** list item, then click **Save**.
47. Click the **Create** button.
48. Click in the **Solution Name** field and enter "My laptop is on fire".
49. Click in the **Description** field and enter "Certain laptops are highly flammable".
50. Click the **Type** list.
51. Click the **Hardware** list item, then click **Save**.
52. Click the **Create** button.
53. Click in the **Solution Name** field and enter "Windows 7 tips and tricks".
54. Click in the **Description** field and enter "Download TweakUI".
55. Click the **Type** list.
56. Click the **Software** list item, then click **Save**.
57. Click the **Create** button.
58. Click in the **Solution Name** field and enter "Slow DSL Line".
59. Click in the **Description** field and enter "Review upgrade options."
60. Click the **Type** list.
61. Click the **Hardware** list item.
62. Click the **Network** list item, then click **Save**.
63. Click the **Save** button.
64. Next, you’ll create a subtab in your help request object. Click the **Navigator** link.
65. Click the **Application Composer** link.

**Note**

You might need to click the **More >>** link if you don’t see Application Composer.

66. Click the **Application** list.
67. Click the **Sales** list item.
68. Scroll down and click the **Expand** button next to the help request object.
69. Scroll down and click the **Pages** tree item.
70. Click the **Create** button.
71. Click the **Context** link option, then click **Next**.
72. Click the **Data Object** list.
73. Click the **Solution** list item.
74. Click in the **Display Label** field and enter "Solutions".
75. Click the **Field Name** list.
76. Click the **Type** list item.

The **Operator** field is filled in automatically with the **Contains** value.
77. Click the **Value Type** list, then click the **Object** field list item.
78. Click the **Source Object** list, then click the **Type** list item.

79. In the **Configure Summary Table** panel, double-click the following list items:
   - **Description**
   - **Solution Name**
   - **Type**

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

81. Next, you’ll test your subtab. Click the **Navigator** link.

82. Click the **Help Requests** link.

83. Click the **Laptop doesn't work.** link.

   The Solutions tab is displayed, showing solutions that have the same **Type** as the help request.

84. Click the **Hardware** list.

85. Click the **Software** list item.

86. 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 **Application Composer** link.

   **Note**

   You might need to click the **More >>** link if you can’t see Application Composer.

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.

9. Enter ”Activities” in the **Plural Label** field.

10. Click in the **Object Name** field and enter ”Activity”.
11. Click the OK button.

The new Activity custom child object appears in the object tree.

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.

Next, we will look at the help request object.

29. Click the Pages button.

30. 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,
An End-to-End Configuration Example

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. Sign in with User ID "MHOOPE" and Password "Welcome1".
2. Click the Application list, then click the Sales list item.
3. Expand Custom Objects, then expand Activity.
4. Click the Server Scripts link.
5. Click the Triggers link.
6. Click the Add a new Trigger object.
7. Enter "XYonlyIfOpenHR" in the Trigger Name field.
8. Enter "Prevent creation of an activity if the help request is closed." in the Description field.
9. Click the Expand Show Palette button.
10. Click the Fields link.
11. Click the Expand link next to Activity on the Fields tab.
12. Click the Help Request object.
13. Click the Status object.

Note
You might need to scroll to see it.

14. Enter if ( in the Expression field.
15. With Status selected, click the Insert button.

The application inserts code that identifies the Status field.
16. Enter =='Closed')
17. Press [Enter].
18. Enter adf.error.raise(null) in the Expression field.
19. Press [Enter].
20. Enter } in the Expression field.

Your completed expression should look like this: if

(XYHelpRequest_c?.Status_c=='Closed')
{ adf.error.raise(null) }

21. Click the Validate button.

A warning appears; you can safely ignore it in this case.
22. Click the OK button.
23. Click the Expand Error Message link.
25. Click the Save and Close link.
26. Click the Navigator link.
27. Click the Help Requests link.
28. Click the Laptop doesn't work. link.
29. Click the Status list, then click the Closed list item.
30. Click the Save button.
31. 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.
32. Click the OK button.
33. Click the Status list, then click the Open list item.
34. 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.
   Note
   You might need to click the more >> link if you don’t see 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>Value in Meaning Field</td>
<td>Value in Lookup Code Field</td>
<td>Value in Display Sequence Field</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------</td>
<td>--------------------------------</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.

   **Note**
   
   You might need to click the more >> link if you can’t see Application Composer in the Navigator list.

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 Datetime option.
6. Click the **OK** button.
7. Enter "Due" in the **Display Label** field.

   Notice that the application automatically provides a default value for the **Name** field.
8. In the Default Value area, click the **Expression** option.
9. Click the **Enter the expression used for the field's default value** button.
10. Click the **Date** object, then click the **today** object.
11. Click the **Insert** button.
12. Enter "+1" into the **Expression** field, then click **OK**.
13. Click the **Save and Close** button.
14. Under the Activity object, click the **Server Scripts** tree item.
15. Click the **Validation Rules** tab.
16. Under Field Rules, click the **Add a new validation rule** button.
17. Click the **Field Name** list.
18. Click the **Due** list item.
19. Enter "futureDate" in the **Rule Name** field.
20. Enter "Due date must be in the next week." in the **Description** field.
21. Under Rule Definition, click the **Expand Show Palette** button.
22. Click the **Keywords** tab.
23. Click the **newValue** object.
24. Click the **Insert** button.
25. Click the **>=** list item.
26. Click the **Functions** tab.
27. In the Category column, click the **Date** object.
28. In the Function column, click the **today** object.
29. Click the **Insert** button.
30. Click the **&&** button.
31. Enter **newValue<= today() + 7.** into the **Expression** field.
32. Click the **Validate** button.

   A message is displayed, indicating that the expression parsed successfully.
33. Enter "Due date must be in the future but no more than one week from today." in the **Error Message** field.
34. Click the **Save and Close** button.
35. Expand the Help Request object.
36. Scroll and click the **Pages** tree item.
37. In the Subtabs panel, click the **Activities** object.
38. Click the **Edit** button.

39. In the Configure Summary Table area, double-click the **Due** list item to move it to the Selected Fields list.

40. In the Configure Detail Form area, double-click in the **Due** field to move it into the Selected Fields list.

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

42. Click the **Navigator** link.

43. Click the **Help Requests** link.

44. Click the **Laptop doesn't work.** link.

45. The Activities tab should be selected by default; if it is not, select it.

46. Click the **Add Row** button.

47. Enter "Reviewed options with customer." in the **Activity Name** field.

48. Enter "Recommended fire extinguisher." in the **Description** field.

49. At the far right of the **Description** field, click the **Select Date and Time** button.

50. For the purposes of this activity, we will assume today is May 18. Click May 28 on the calendar.

51. 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.

52. 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.

53. Click May 23.

54. Click the **OK** button, then click **Save** twice.

   The date can be saved now, because it is within the allowed range.

55. 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.

   **Note**

   You might need to click the more >> button if you can't see Application Composer in the list.

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** object.
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 **Finish**.
17. In the Objects list, click the vertical scrollbar.
18. Under Help Request, click the **Pages** tree item.
19. The Enterprise 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 and Tree Nodes: 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.
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 show 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.
15. Scroll and click the Primary Web Site object.
16. Click the Insert button.
17. Click the Finish 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 Company URL object.
28. Click the Insert button, then click Save and Close.
29. Click the Navigator link.
30. Click the Help Requests link.
31. Click the Laptop doesn't work. link.
32. 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.
3. Click the Help Request link.
   Notice that you get an error message because your user doesn’t have permission to access this information.
4. 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 Application Composer link under Tools.
   Note
   You might need to click the more >> link to see Application Composer.
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 Request** 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 **Application Composer link** under **Tools**.

   **Note**

   You might need to click the **more >>** link to see **Tools**.

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 Request link.
5. Click the Laptop doesn't work link.

Notice that the Activities tab now shows content, and you can create new Activities.
abstract role
A description of a person’s function in the enterprise that is unrelated to the person’s job (position), such as employee, contingent worker, or line manager. A type of enterprise role.

action
The kind of access named in a security policy, such as view or edit.

activity stream
A feature that tracks and displays actions and messages from people whom you are connected to in your social network, as well as activities from the application.

BPEL
Business Process Execution Language; a standard language for defining how to send XML messages to remote services, manipulate XML data structures, receive XML messages asynchronously from remote services, manage events and exceptions, define parallel sequences of execution, and undo parts of processes when exceptions occur.

business object
A resource in an enterprise database, such as an invoice or purchase order.

chrome
The set of visual elements around the perimeter of a component or task flow that enables users to act directly on the object. Elements that make up the chrome of a component include the header; border; resize handle; edit, collapse, expand, and remove icons; and Actions menu.

condition
An XML filter or SQL predicate WHERE clause in a data security policy that specifies what portions of a database resource are secured.

customization
A change to standard, predefined Oracle Fusion Applications artifacts. Customizations impact multiple users.

customization layer
A level that represents the types of users impacted by a customization, for example all users or only those that meet specific criteria.

data dimension
A stripe of data accessed by a data role, such as the data controlled by a business unit.
**data role**

A role for a defined set of data describing the job a user does within that defined set of data. A data role inherits job or abstract roles and grants entitlement to access data within a specific dimension of data based on data security policies. A type of enterprise role.

**data security**

The control of access to data. Data security controls what action a user can taken against which data.

**data security policy**

A grant of entitlement to a role on an object or attribute group for a given condition.

**database resource**

An applications data object at the instance, instance set, or global level, which is secured by data security policies.

**design time**

The type of activities performed by developers at the code or data model level.

**desktop page**

A page in a user interface that is optimized for extended periods of use with monitors.

**desktop user interface**

A standard user interface that is optimized for extended periods of use with monitors.

**enterprise role**

Abstract, job, and data roles are shared across the enterprise. An enterprise role is an LDAP group. An enterprise role is propagated and synchronized across Oracle Fusion Middleware, where it is considered to be an external role or role not specifically defined within applications.

**entitlement**

Grants of access to functions and data. Oracle Fusion Middleware term for privilege.

**extension**

A completely new artifact in addition to what is predefined in Oracle Fusion Applications, for example a custom 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**
Grouping of extensible data fields called segments, where each segment is an attribute added to an entity for capturing additional information.

**functional area**
A set of simplified pages containing content that a user needs to accomplish a business goal. Examples of this content include searches, data entry fields, and analytics.

**global area**
The region across the top of the user interface. It provides access to features and tools that are relevant to any page you are on.

**job role**
A role for a specific job consisting of duties, such as an accounts payable manager or application implementation consultant. A type of enterprise role.

**object workflow**
Object workflows represent a series of automated tasks configured for a business object, which are invoked based on a set of trigger conditions.

**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.

**personalization**
A change users make to control the look or behavior of the application. Personalizations impact only the user making the change.

**privilege**
A grant or entitlement of access to functions and data. A privilege is a single, real world action on a single business object.

**role**
Controls access to application functions and data.

**run time**
The type of activities performed by users while they are in a running application.
sandbox
A run time session that commits changes out of reach of mainline users.

sandbox
A testing environment that separates sections of an application so that changes made in this environment do not affect the mainline code or other sandboxes.

simplified page
A page in a user interface that is optimized to provide quick access to high-volume, self-service tasks from any device.

simplified user interface
A user interface that is optimized to provide quick access to high-volume, self-service tasks from any device.

site layer
The customization layer in which personalizations are made, which impact only the user making the change.

springboard
The area on the simplified user interface home page, and above all simplified pages, that contains a set of functional area icon buttons.

SQL predicate
A type of condition using SQL to constrain the data secured by a data security policy.

WSDL
Abbreviation for Web Services Description Language. It is an XML format that provides a model for describing Web services.

XML filter
A type of condition using XML to constrain the data secured by a data security policy.