Oracle Fusion Applications CRM Extensibility Guide

11*g* Release 7 (11.1.7)

Part Number E20388-06

July 2013

Oracle® Fusion Applications CRM Extensibility Guide

Part Number E20388-06

Authors: Kristin Penaskovic, Iqtidar Hussain, Michael Phelan, Nuzhath Shereen, Charles Cowan

Contributors: Sean Frogner, Claudio Moraes, Ying Wang, Sol Solorzano, Steve Muench, Pundir Prabhakar

Contents

5 Application Composer: Extending Simplified Pages

6 Application Composer: Using Groovy Scripts

7 Application Composer: Creating Object Workflows

8 Application Composer: Creating Custom Subject Areas

9 Page Composer: Customizing Oracle Fusion CRM Applications

10 Oracle Fusion Common CRM Extensibility

11 Oracle Fusion Customer Center Extensibility

12 Oracle Fusion Sales and Oracle Fusion Opportunity Management Extensibility

13 Oracle Fusion Sales Forecasting Extensibility

14 Oracle Fusion Marketing Extensibility

15 Oracle Fusion Partner Relationship Management Extensibility

16 Oracle Fusion Mobile Sales Extensibility

17 Exporting and Importing Customizations

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.

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. You can access the guides from the **Guides** menu in the global area at the top of Oracle Fusion Applications Help pages.

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.

Guide	Intended Audience	Purpose
Common User Guide	All users	Explains tasks performed by most users.
Common Implementation Guide	Implementors	Explains tasks within the Define Common Applications Configuration task list, which is included in all offerings.
Functional Setup Manager User Guide	Implementors	Explains how to use Oracle Fusion Functional Setup Manager to plan, manage, and track your implementation projects, migrate setup data, and validate implementations.
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.

For guides that are not available from the Guides menu, go to Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.

Other Information Sources

My Oracle Support

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

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.

Note
The content of Oracle Enterprise Repository reflects the latest release of Oracle Fusion Applications.

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 the **Send Feedback to Oracle** link in the footer of Oracle Fusion Applications Help.

1

What's New in CRM Extensibility Guide in 11g Release 7 (11.1.7)?

What's New in This Guide?

This topic provides information on what's new in this guide in Release 7. These changes include topics that have been added since the last release.

What's new in this guide?

This section lists the new chapters that have been added to this guide in this release:

The **Application Composer: Extending Simplified Pages** chapter includes the following:

- **Extending Simplified Pages**
  - Extending Simplified Pages for Contacts: Explained
  - Extending Simplified Pages for Customers: Explained
  - Extending Simplified Pages for Leads: Explained
  - Extending Simplified Pages for Opportunities: Explained
  - Extending Simplified Pages for Leads using Oracle Fusion CRM Application Composer: Worked Example
  - Extending Simplified Pages for Opportunities Using Oracle Fusion CRM Application Composer: Worked Example
  - Extending Simplified Pages for Opportunities using Page Composer: Worked Example

The **Exporting and Importing Customizations** chapter includes the following:

- **Exporting and Importing Customizations**

- Exporting and Importing Customizations: Explained
- Supported CRM Application Composer Customizations and Best Practices: Explained
- Moving Customizations Using Customization Sets: Highlights

Important

In this release, new chapters for product-specific extensibility have been created to better organize topics by product area. In some cases, existing content has been moved to a new chapter. In other cases, new content has been added entirely. The below chapters contain new content.

The **Oracle Fusion Sales and Oracle Fusion Opportunity Management Extensibility** chapter includes the following changes:

- **Extending Oracle Fusion Sales and Oracle Fusion Opportunity Management**
  - Customizing Opportunity Pages: Explained
  - Customizing Sales Pages: Explained
  - Customizing the Sales Dashboard Using Page Composer: Worked Example
  - Creating Opportunity Saved Searches by Job Role Using Page Composer: Worked Example
  - Extending Simplified Pages for Opportunities: Explained
  - Extending Simplified Pages for Opportunities Using Oracle Fusion CRM Application Composer: Worked Example
  - Extending Simplified Pages for Opportunities using Page Composer: Worked Example

The **Oracle Fusion Marketing Extensibility** chapter includes the following changes:

- **Extending Oracle Fusion Marketing**
  - Customizing Sales Campaigns in Oracle Fusion Marketing: Explained
  - Sales Lead Approval Flow: Worked Example
  - Extending Simplified Pages for Leads: Explained
  - Extending Simplified Pages for Leads using Oracle Fusion CRM Application Composer: Worked Example

The **Oracle Fusion Mobile Sales Extensibility** chapter includes the following changes:

- **Extending Oracle Fusion Mobile Sales**
  - Mobile Sales Extensibility: Explained

- Customizing Oracle Fusion Mobile Sales: Worked Example
- How can I add an Oracle Business Intelligence report to Oracle Fusion Mobile Sales?
- Testing Oracle Fusion Mobile Sales Customizations Using a Sandbox: Worked Example

What's changed in each chapter?

This section lists the chapters that contain the changes specific to this release.

The **Application Composer: Using the Application Composer** chapter includes the following changes:

- **Saved Searches**
  - Creating Saved Searches Using Page Composer: Highlights

The **Application Composer: Using Groovy Scripts** chapter includes the following changes:

- **Invoking Internal or External Web Services Using Groovy Scripts**
  - Calling Web Services from Groovy Scripts: Explained
  - Web Service References for Groovy Scripts: Explained
  - Web Service Calls in Groovy Scripts: Examples
  - Calling an External Web Service from Groovy when No Security Scheme is Required: Worked Example
  - Calling an External Web Service from Groovy with Message Protection Security: Worked Example
  - Calling an External Web Service from Groovy with Separate User Credentials over SSL: Worked Example
  - Calling an Internal Web Service from Groovy with Message Protection Security: Worked Example
  - Calling an Internal Web Service from Groovy with Separate User Credentials over SSL: Worked Example
  - Calling an Internal Web Service from Groovy using SAML for ID Propagation: Worked Example
- **Accessing View Objects Using Groovy Scripts**
  - Accessing View Objects in Scripts: Explained
- **Global Functions**
  - Global Functions: Explained

The **Application Composer: Creating Object Workflows** chapter includes the following changes:

- **Configuring Approval Flows Using Business Process Composer**
  - Object Workflows and Business Processes: How They Work Together
  - Object Workflows and Business Process: Critical Choices
  - Sales Lead Approval Flow: Worked Example

The **Page Composer: Customizing Oracle Fusion CRM Applications** chapter includes the following changes:

- **Customizing applications using Page Composer**
  - Customizing Simplified Pages Using Page Composer: Points to Consider

The **Exporting and Importing Customizations** chapter includes the following changes:

- **Supported customizations and customization sets**
  - Exporting and Importing Customizations: Explained
  - Supported CRM Application Composer Customizations and Best Practices: Explained
  - Moving Customizations Using Customization Sets: Highlights

2

CRM Extensibility: Introduction

CRM Extensibility: Overview

Welcome to the Oracle Fusion Applications CRM Extensibility Guide. You are reading this guide because you are interested in extending and customizing your Oracle Fusion CRM set of applications. This guide introduces you to the various tools that you can use to make application changes.

Oracle Fusion CRM Application Composer

The Oracle Fusion CRM Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application developers, to customize and extend an Oracle Fusion CRM application. Make the type of data model changes which, for non-CRM 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 a CRM 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.

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 Fusion CRM application.

Page Composer

Page Composer is a page editor that you can use to easily edit the user interface composition at run time. In Oracle Fusion CRM, 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 Fusion

CRM 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 Fusion CRM application.

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 Fusion CRM application, 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.

Extending CRM Applications: Top Tasks

Before you start to extend and customize any application within Oracle Fusion CRM, refer first to the Oracle Fusion Applications Extensibility Guide for Business Analysts to learn more about all the extensibility options and tools that are available to you.

The Oracle Fusion Applications Extensibility Guide for Business Analysts walks you through the customization process for all Oracle Fusion applications, not just within Oracle Fusion CRM. After reviewing that guide, you can then review the Oracle Fusion CRM Extensibility Guide to understand in more detail how to use various composers to extend and customize an application within Oracle Fusion CRM.

Getting Started: Review the Oracle Fusion Applications Extensibility Guide for Business Analysts

- Refer to the Oracle Fusion Applications Extensibility Guide for Business Analysts to learn which tools are available to you to change an

application. You will also learn about the customization development life cycle, including how to use the Sandbox Manager and Customization Manager.

See: Introduction to Customizing and Extending Oracle Fusion Applications

- After reviewing the Oracle Fusion Applications Extensibility Guide for Business Analysts, you can then review the Oracle Fusion CRM Extensibility Guide, which provides detailed information about using the Application Composer.

3

Application Composer: Using Sandboxes

Using Sandboxes: Overview

Read this chapter to learn about proper and recommended sandbox usage when customizing and extending your Oracle Fusion CRM 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

Maintain sandboxes using the Sandbox Manager, which you can access by selecting Manage Sandboxes... from the Administration menu.

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 Fusion CRM Applications 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 Fusion Applications 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 Fusion Applications Extensibility Guide for Business Analysts. Or, review this document to learn how to customize Oracle Fusion CRM Applications, specifically, using sandboxes.

Sandboxes let users make changes isolated from the mainline application, as well as from other sandboxes. The mainline 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. When making changes in a sandbox, your customizations will not be available to any other sandbox or to the mainline application until you have published your sandbox. When publishing a sandbox, the included application customizations overwrite the mainline 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. Instead, always use sandboxes whenever possible.

Composers

To customize applications within Oracle Fusion CRM, you can use these composers:

- Oracle Fusion CRM 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 Fusion CRM 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 Extensibility Guide for Business Analysts on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation

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.

- 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. 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.

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

Sandboxes: How They Work with Some Customizations and Features

When customizing Oracle Fusion CRM Applications using the Oracle Fusion CRM Application Composer or Page Composer within a sandbox, be aware of the exceptions listed below.

Lookup Types and Values

Lookup types and lookup values are considered seed data, and are not stored inside the MDS. 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.

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 Fusion Applications.

Note that you can also create custom subject areas, which are report subject areas that you build using the Application Composer. To create a custom subject area, however, you must be in the mainline 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.

Note

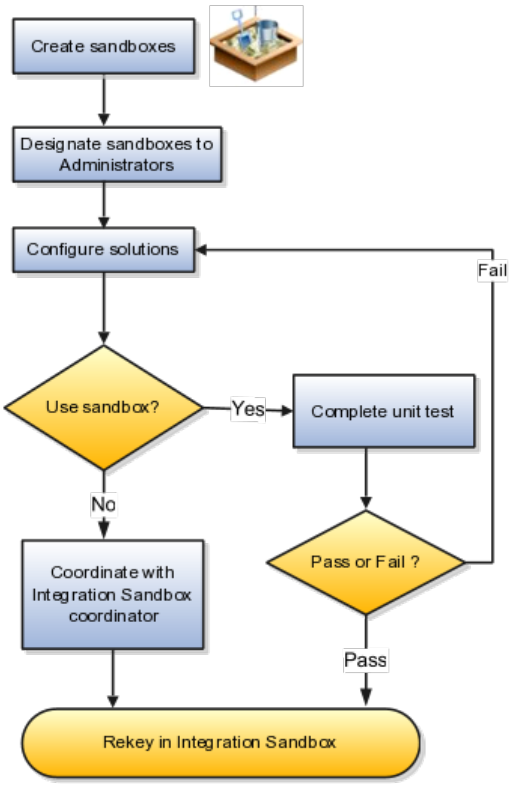
Since e-mail templates are part of the object workflow feature, you must create these templates outside a sandbox as well, in the mainline application.

Import and Export

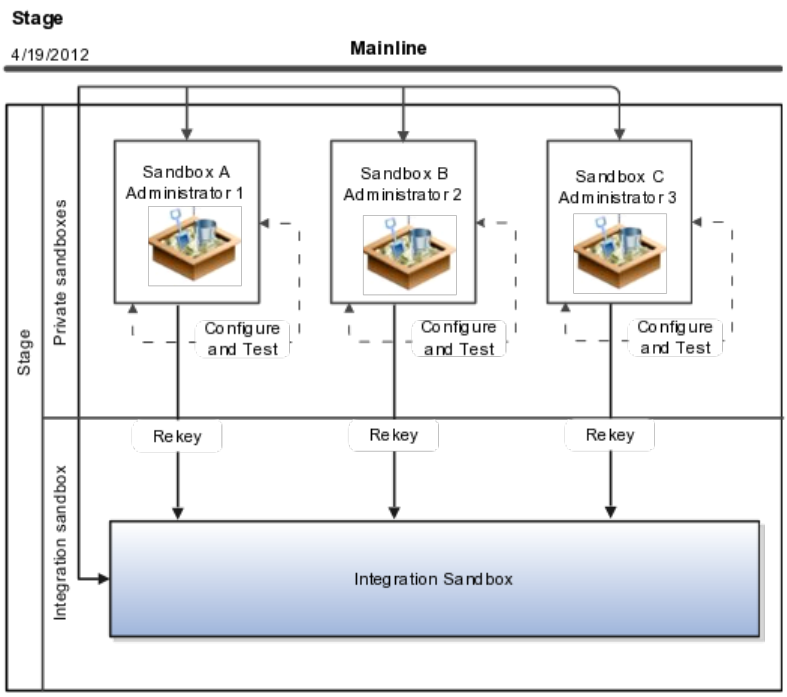
To support the importing and exporting of the custom objects that you created with the Application Composer, you must first generate the object artifacts required for both file-based import and bulk export. Note that this task is not supported from within a sandbox, and can only be completed in the mainline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

Sandbox Development Lifecycle Components: Explained

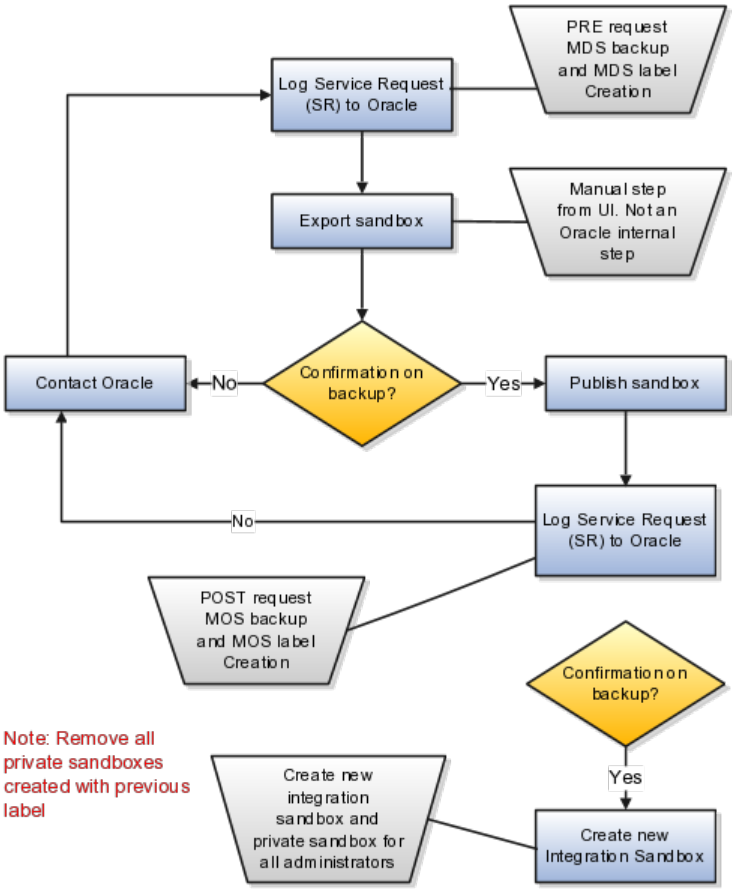
The development lifecycle and planning may look like the following for each customization cycle, usually one week:

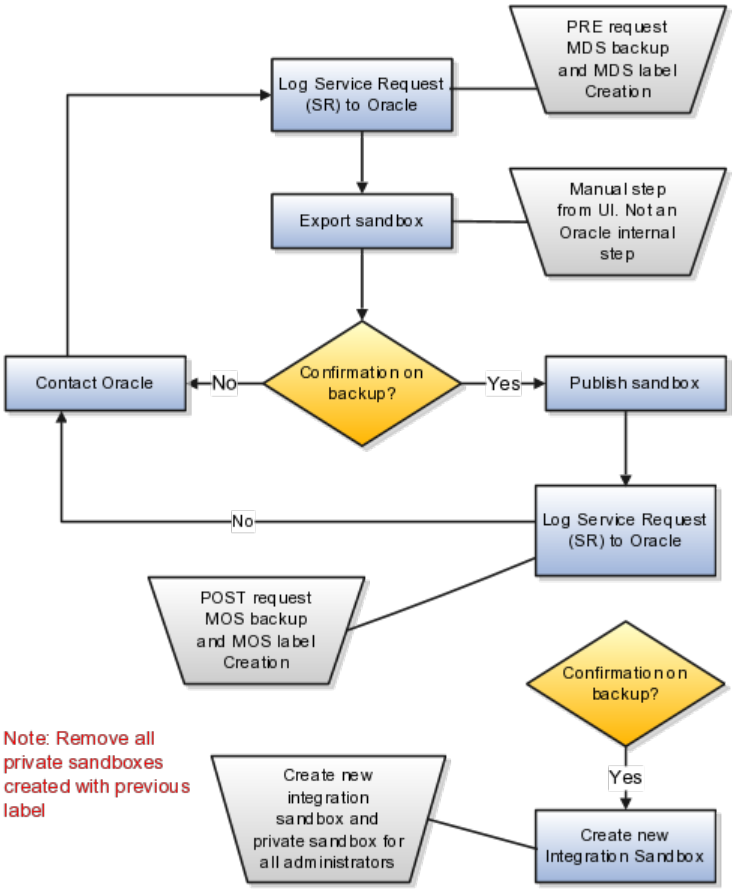


Stage



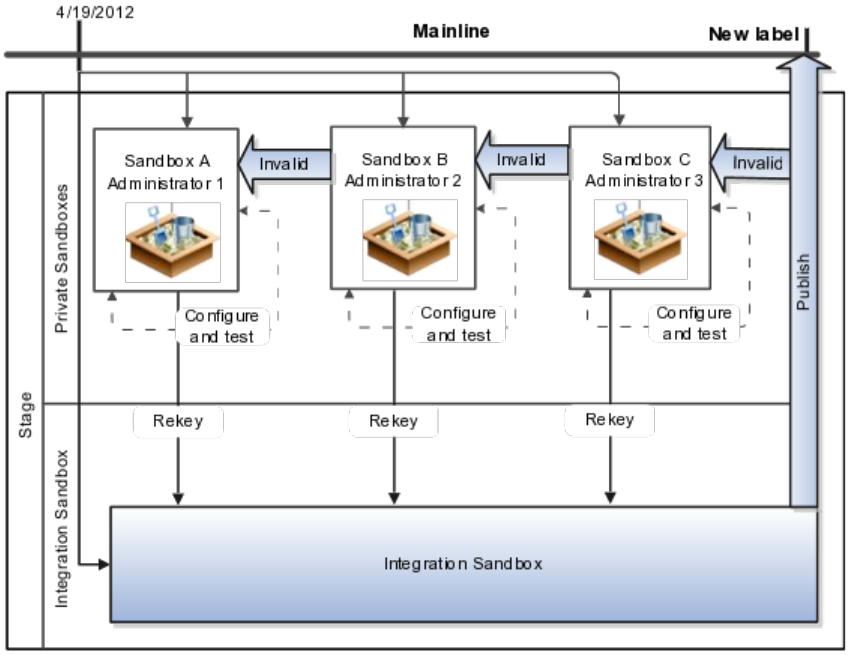
Publishing a Sandbox



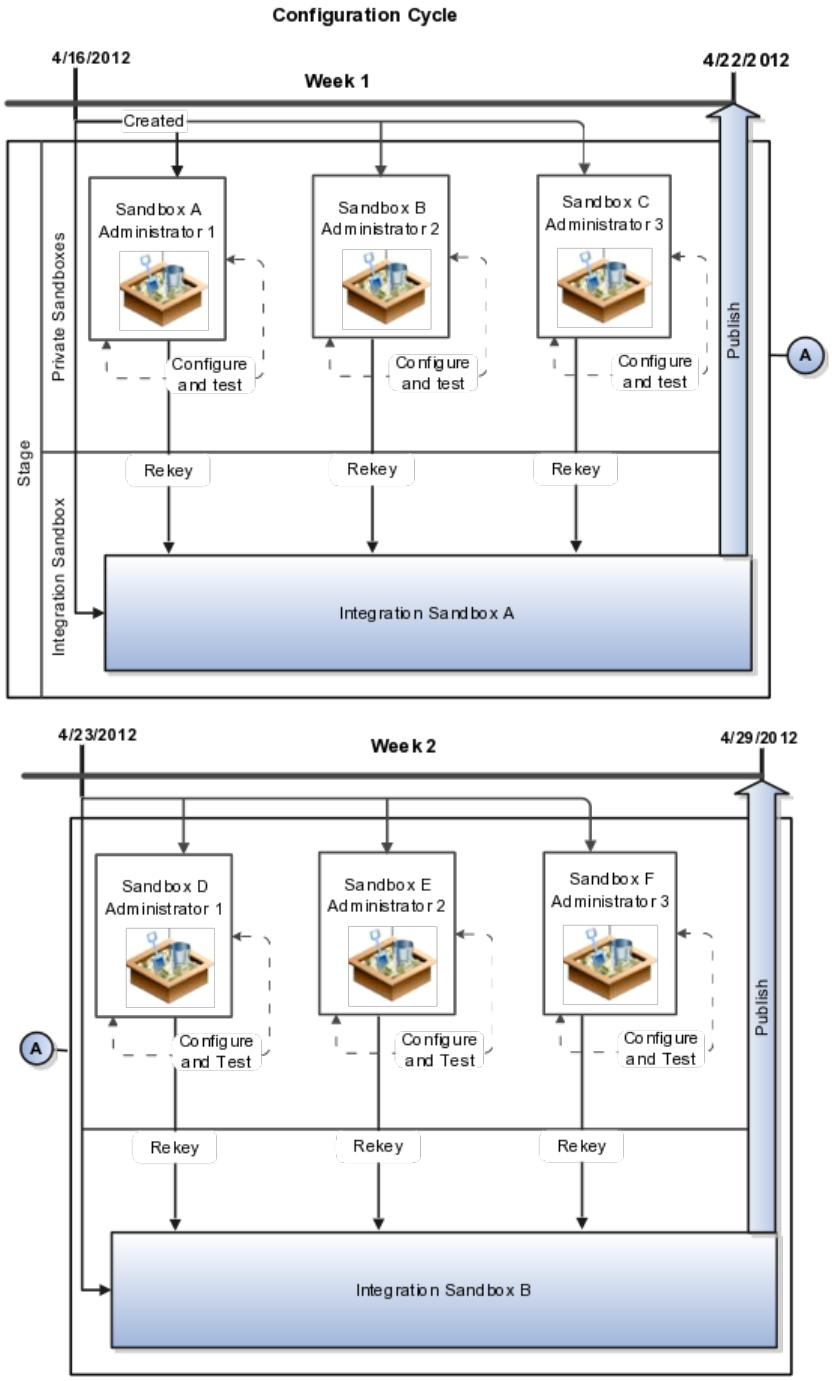


Invalid Sandbox

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.



Example of a Multi-week Cycle



Using the Sandbox Manager: Explained

Maintain sandboxes using the Sandbox Manager, which you can access by selecting Manage Sandboxes... from the Administration menu.

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 outside Oracle Fusion CRM without affecting the mainline, create a separate data security sandbox by selecting the Create data security sandbox check box when creating a new sandbox. When working within Oracle Fusion CRM, 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 mainline application. 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 Fusion Applications and log back in. This helps you to avoid conflicts by ensuring that the cache is cleared.

Note that if you log out and log back in, 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

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 mainline application. After exiting a sandbox session, you should always log out from Oracle Fusion Applications and log back in. This helps you to avoid conflicts by ensuring that the cache is cleared.

Important

Once back in the mainline application, avoid making customizations using the Oracle Fusion CRM 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 mainline application. 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.

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

The only supported sandbox operations are described below.

Supported Sandbox Operations

The only supported sandbox operations are:

- 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.
- Publish - publish a sandbox using extreme caution. Once a sandbox has been published, all existing sandboxes derived from the same mainline 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.

Importing Sandboxes

Do not import sandboxes. This operation is reserved for Oracle internal development only.

Signing Out of Oracle Fusion Applications

Sign out and sign back in when:

- Activating a sandbox for the first time
- Exiting a sandbox
- Switching between sandboxes

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 mainline application. The resulting mainline 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.)

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 mainline application.

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 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 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.

4

Application Composer: Using the Application Composer

Using Application Composer: Overview

Read this chapter to learn about the primary task flows that are available in Oracle Fusion CRM 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 CRM applications
- How to define a custom object
- 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 object and custom fields
- 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 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 CRM Applications: How It Works

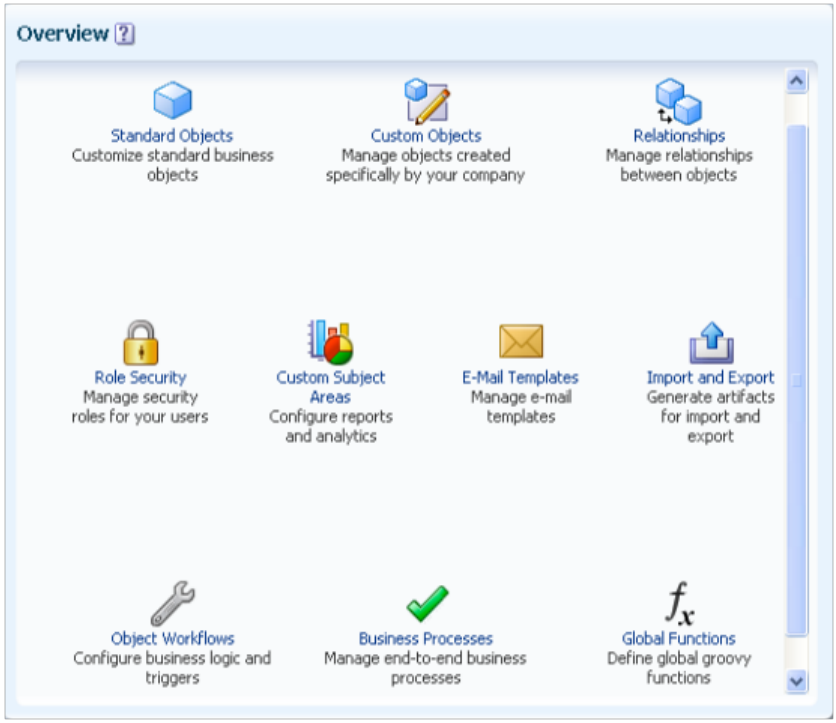
The Oracle Fusion CRM Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application

developers, to customize and extend an Oracle Fusion CRM application. Make the type of data model changes which, for non-CRM 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. The Application Composer is a design time at run time tool, which means that you can navigate to the Application Composer directly from a CRM 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.

Pattern-Based Application Design

The 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 the Application Composer creates the underlying object artifacts for you.

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



From the 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.

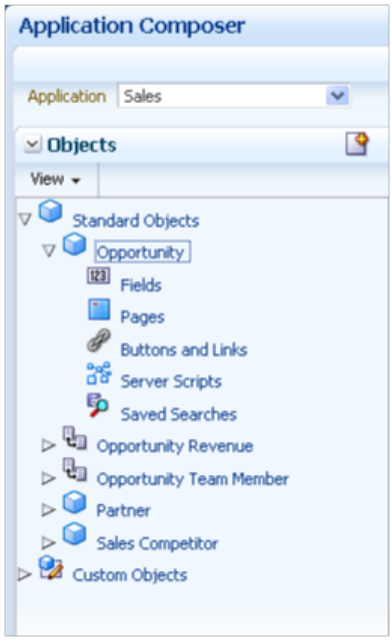
- Write application logic, such as triggers, validation rules, and workflows, for an object or for use across multiple objects.
- Implement functional and data-level security for custom objects.
- Enable objects for custom reporting.

Getting Started: The Application Composer's Overview Page

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

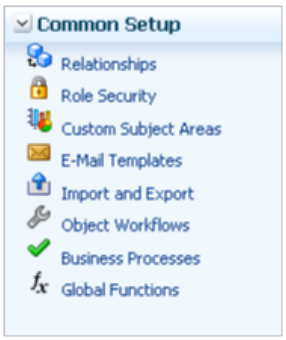
From the main Overview page, select the application you want to customize using the **Application** choice list. Then:

- 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 CRM application.

Oracle Fusion Applications Extensibility

The Application Composer is but one tool that lets you customize and extend your Oracle Fusion CRM applications. To learn more about extensibility options that are available to you across all Oracle Fusion applications, see the Oracle Fusion Applications Extensibility Guide for Business Analysts.

Customizing CRM Applications Across Application Boundaries: Explained

When you customize applications using the Oracle Fusion CRM Application Composer, you always do so within the context of a Web application, such as Oracle Fusion Sales or Oracle Fusion 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 the 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 CRM applications:

- Oracle Fusion Customer Center
- Oracle Fusion Marketing
- Oracle Fusion 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 the Oracle Fusion CRM Application Composer. The new custom object will belong only to the application that you select.

Tip

When you first open the 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 the 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 CRM applications:

- Oracle Fusion Customer Center
- Oracle Fusion Marketing
- Oracle Fusion Sales

Additionally, when customizing an object in Customer Center, Marketing, or Sales, your customizations can include the following types of objects:

- Custom top-level objects

  When customizing an object in Customer Center, Marketing, or Sales, your customizations can include any custom top-level object that was created in Customer Center, Marketing, or Sales.
- Standard top-level objects

  When customizing an object in Customer Center, Marketing, or Sales, your customizations can include any standard top-level object that exists in any Oracle Fusion CRM Web application.

Object Model Customizations

When customizing an object's model in Customer Center, Marketing, or Sales, you can include objects from other Web applications. The types of changes you can make include:

- Dynamic choice lists

  A dynamic choice list is a field that contains a list of values which are populated from the actual data of another object.

  From a cross-application perspective, you can create a dynamic choice list field for an object in one Web application that is populated by the records from an object in another Web application.

  - For example, create a dynamic choice list field for the opportunity object (Oracle Fusion Sales) which is populated with sales lead records (Oracle Fusion Marketing).
  - Or, create a dynamic choice list field for the sales account object (Oracle Fusion Customer Center) which is populated with competitor records (Oracle Fusion Sales).

- 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.
  - 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.

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 in Oracle Fusion Common CRM, 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 include objects from other Web applications. The types of changes you can make include:

- Subtabs and tree nodes

  You can display details that are related to the current object but derived from another object. You do this by adding subtabs to an object's details page, and specifying the source of subtab data, or by adding tree nodes to the object'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 include objects from other Web applications in your scripts. The types of changes you can make include:

- When writing a script, access data from any standard object in any CRM 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 Oracle Fusion CRM 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.

Oracle Fusion Common CRM Objects: Exceptions

Objects available in the Common Web application include CRM 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 CRM Applications Using the Application Composer: Explained

The Oracle Fusion CRM Application Composer provides a series of task flows which let you customize and extend an Oracle Fusion CRM 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 CRM application that you are customizing.

Available Customization Task Flows

Different sets of customization task flows are available to you, depending on the CRM application that you are customizing. See Customizing Oracle Fusion CRM Applications Using Oracle Fusion CRM 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 CRM applications:

- Oracle Fusion Common CRM

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

Defining Objects: Explained

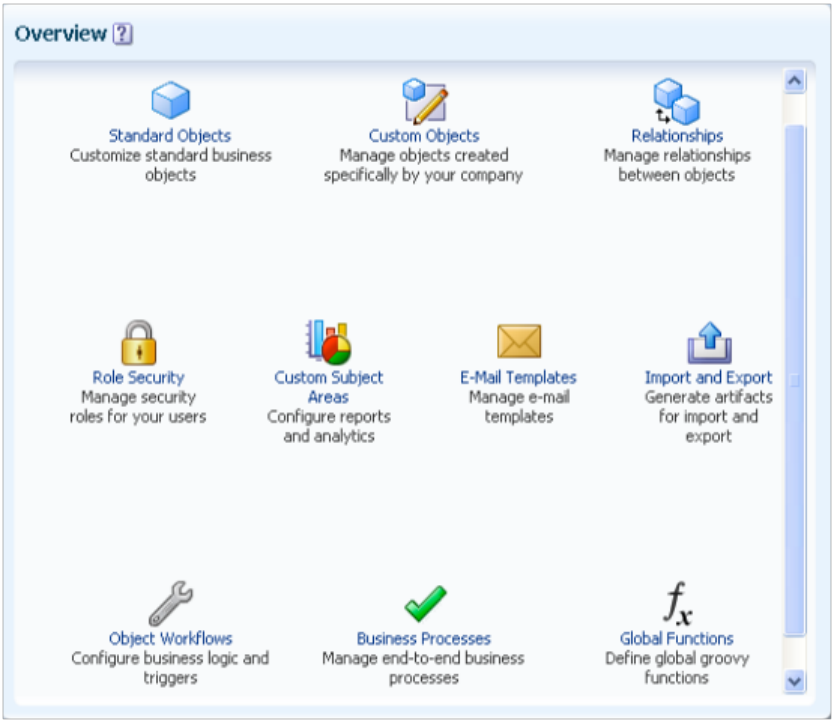
One of the primary customization options available to you when using the Oracle Fusion CRM Application Composer is the ability to extend a CRM application's object model. Customize CRM objects by adding new fields to an existing object (standard objects), or create entirely new objects (custom objects). Standard objects are objects that are delivered with a CRM application, and made available to the Application Composer for customization. Custom objects are objects that you create using the 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 the Application Composer before you begin to customize or extend your CRM 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 the Application Composer from a CRM application at run time by using the Navigator menu. The first view of the Application Composer is the main Overview page, which is the entry point into all your customization options.



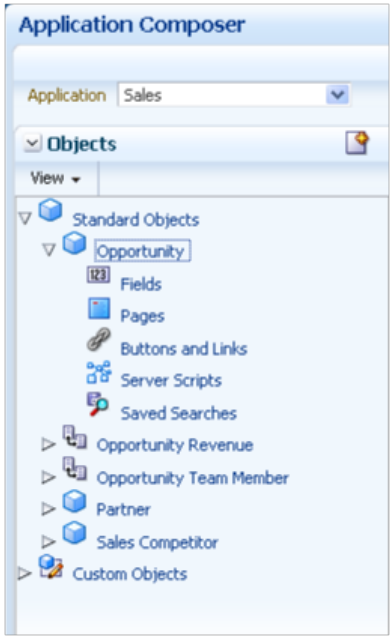
On the main Overview page, the regional pane at left displays the object tree, which lets you browse an application's existing object configuration in a tree

format. The object tree reflects the latest configuration of the application: both standard objects as well as custom objects.

To use the object tree:

1. Select **Application Composer** from the Navigator menu, under the Tools category.
2. On the main Overview page, select an application from the **Application** choice list.

3. For each object node, whether standard or custom, expand it further to view and edit object details, such as an object's fields and desktop pages where the object is exposed.



Note

At the top of the object tree, you can also click the New icon to create a new custom object.

For both standard and custom objects, you can view and edit the following details:

- Fields
  Add new fields to an object.

- Pages

  Modify the pages on which an object appears.

- Buttons and links

  Add buttons or links to desktop 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 the 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.

   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

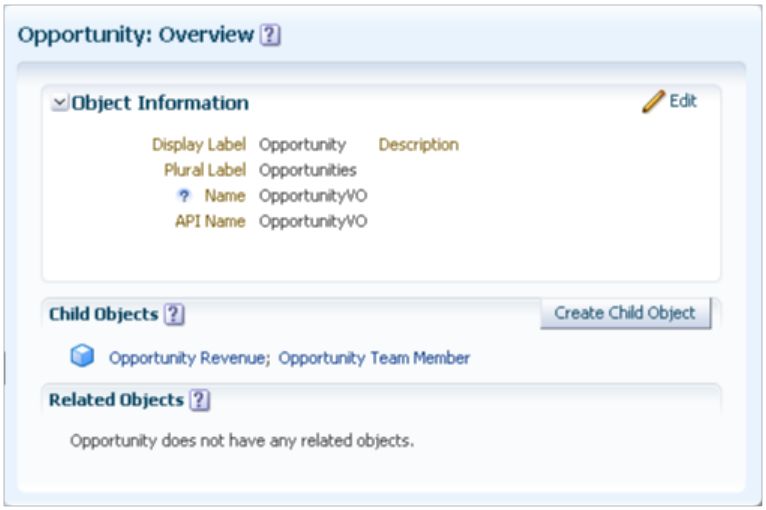
To create a custom child object, click the Create Child Object button in the standard or custom objects summary table, or from an object's Object Overview page.

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 for an object is automatically derived using the logical name followed by _c. You use the API name in Groovy expressions that you build with the expression editor, 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 the 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

The Application Composer does not support the deletion of either standard or custom objects. If you no longer need an object, 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 the 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 CRM object and a common object: resource, organization contact, organization profile, 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 Fusion CRM 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.

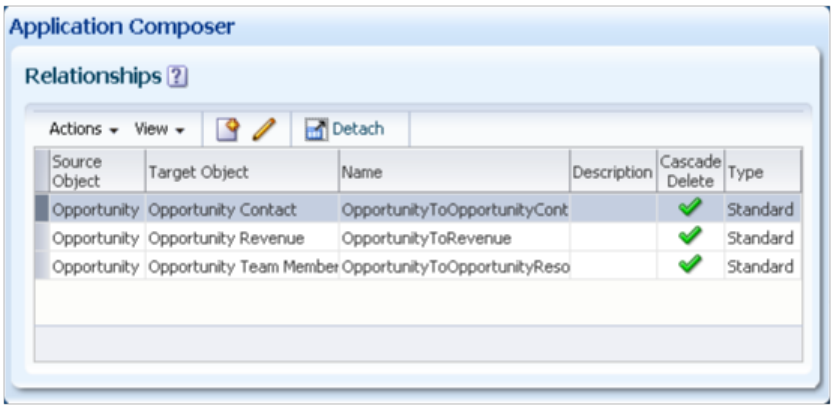
Creating Reference Relationships

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.

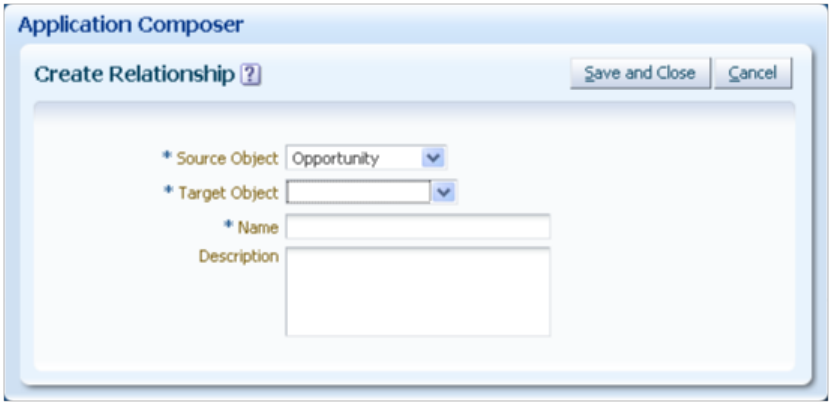
You can also create a foreign key-based, one-to-many relationship by creating child objects and dynamic choice lists. These implicit relationships are discussed in related topics.

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:

- Trading Community Resource

- Trading Community Organization Contact
- Trading Community Organization Profile
- Trading Community Address

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. The 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 the 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. The 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 the Oracle Fusion CRM Application Composer, you can extend a CRM 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 CRM application. The fields that you add to an object are custom fields. When creating a custom field, the 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, the Application Composer automatically creates all the underlying object artifacts for you, and provides full Web service support for those new fields, as well. The Application Composer also makes it easy to enable your object model extensions for importing and exporting.

Review these aspects of editing fields in the Application Composer before you begin to customize or extend your CRM 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 CRM 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 CRM 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. The Application Composer provides a set of field types that you can choose from when creating new fields:

  - Text
  - Long text
  - Number
  - Date
  - Datetime
  - Check box
  - Percentage
  - Currency

- Formula
- Fixed choice list
- Dynamic choice list

Deleting Fields

The Application Composer does not support the deletion of either standard or custom fields from objects. If you no longer need a field, optionally enter a note in the field description that the field is no longer used.

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 Trading Community Group Profile object are related objects by default and are already delivered with a join. You can display Trading Community Group Profile 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 the Oracle Fusion CRM Application Composer.

Joins are delivered by default for the Sales Account object for these Oracle Fusion Common CRM objects:

- Trading Community Group Profile
- Trading Community Organization Profile
- Trading Community Person Profile

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.

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.

Field Types and Field Properties: Explained

When you create a custom field, 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 a 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 a object, the Oracle Fusion CRM 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.

- **Text**

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

- **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.

- **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.

- **Check box**

  Users can select a check box, indicating a true or false attribute of a record.

- **Percentage**

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

- **Currency**

  Users can enter a currency amount.

- **Fixed choice list**

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

- **Dynamic choice list**

  Users can select from a list of values, populated from another object's set of records.

- **Formula**

  A formula field is a field that is calculated in the run time CRM 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.

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.

Field Property	Field Property Region	Related Field Types
Label Specify the display label for the field. A maximum length of 80 characters is recommended, although no maximum length is actually enforced.	Appearance These properties control how the field appears to your users in the run time application.	Set this property for all field types.
Help Text The help text displays when users hover over the field in the run time application. A maximum length of 80 characters is recommended, although no maximum length is actually enforced.	Appearance	Set this property for all field types.
Display Width Specify the display width for most field types at run time. The display width is the actual character width for the field on a desktop page. **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. **Warning** For fixed choice lists, note that the display width is determined by the length of the longest string in the choice list.	Appearance	Set this property for all field types except for check box, date, and datetime fields.

Name Enter a unique field name, which is for internal use only. The field name is automatically populated based on the field label you enter, but without spaces. Field names can contain only underscores and alphanumeric characters. They must begin with a letter, not include spaces, not end with an underscore, and not contain consecutive underscores. Field names are limited to 28 characters. You cannot change this property after the field is created. **Tip** It is possible to create custom fields with different names, but the same display label. Avoid this scenario, however, so that you do not see two fields with the same display label when configuring a desktop page. **Note** 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.	Name	Set this property for all field types.
Description Enter a unique field description, which is for internal use only.	Name	Set this property for all field types.

Required Indicate if the field is a required field. You can also optionally use the expression editor to write an expression that describes the conditions required for this field to be required. **Note** If you write an expression to control whether a field is required, then you must also configure the **Depends On** choice list. This choice list includes fields from the current object, and is used in the evaluation of your expression at run time.	Constraints Specify constraints, which let you control the run time behavior of the field.	Set this property for all field types except for formula fields.
Updateable Indicate if the field is an updateable field. You can also optionally use the expression editor 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. **Note** 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.	Constraints	Set this property for all field types except for formula fields.
Searchable 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.	Constraints	Set this property for all field types except for long text and formula fields.

Indexed Enable faster searching by indexing this column. Only a limited number of columns are indexed. Accordingly, use this property only on the most frequently searched fields. You cannot change this property after the field is created.	Constraints	Set this property for text, number, date, datetime, currency, and percentage field types. You cannot index long text, formula, and check box fields, or fixed and dynamic choice lists. **Note** You cannot index long text fields. Instead, your users can use the Oracle Fusion Applications search capability to search these field types.
Fixed Value Specify a literal default value for the field. **Warning** 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.	Default Value	Set this property for all field types except for formula fields and dynamic choice lists.
Expression Use the expression editor to write an expression that dynamically sets the default value for a field at run time.	Default Value	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.

How Fields Types Work With Other Components

When you create new fields for objects, the 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 CRM Extensibility Framework to provide you with the most flexibility possible when customizing and extending your CRM 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 editor, 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 the 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.

Text Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

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

Field Property	Field Property Region
Display Type 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.	Appearance
Maximum Length Indicate the maximum number of characters that a user can enter in the field. You can set a maximum length of 254 characters. If the field is a multiline field, then carriage returns are permitted and count as characters against the total.	Constraints
Minimum Length Indicate the minimum number of characters that a user can enter into the field.	Constraints

Additional Text Field Specifications

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (254 char).
- A object can have a total of 625 fields. Of those 625 fields, 350 fields are reserved for text and check box fields, and fixed and dynamic choice lists.

Long Text Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Fixed Value	Default Value
Expression	Default Value

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

Field Property	Field Property Region
Display Type 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.	Appearance

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 the Oracle Fusion CRM Application Composer, you can extend a CRM 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

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

Field Property	Field Property Region
Decimal Places 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 the 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**.	Constraints

Maximum Length 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: • **Display Width** 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. • **Decimal Places** **Maximum Length - Decimal Places** = the number of digits that can appear to the left of the decimal point. Do not set a maximum length that is shorter than the number of decimal places.	Constraints
Minimum Value Indicate the minimum numerical value that a user can enter into this field.	Constraints
Maximum Value Indicate the maximum numerical value that a user can enter into this field.	Constraints

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.

Date Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

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 the Oracle Fusion CRM Application Composer, you can extend a CRM 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.

Datetime Field Properties

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

The following properties are common across multiple field types:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Name	Name
Description	Name

Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

The following property is unique to only certain field types, including datetime fields:

Field Property	Field Property Region
Display Type Indicate if you want this datetime field to show the date or time, or both.	Appearance

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.

Check Box Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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. Of those 625 fields, 350 fields are reserved for text and check box fields, and fixed and dynamic choice lists.
- The check box field type is not supported by custom subject areas. This means that you cannot add check box fields to a custom report.

Percentage Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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. The 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

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

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

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.
- The Application Composer automatically adds the percent sign.

Currency Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Indexed	Constraints
Fixed Value	Default Value
Expression	Default Value

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

Field Property	Field Property Region
Minimum Value Indicate the minimum numerical value that a user can enter into this field.	Constraints
Maximum Value Indicate the maximum numerical value that a user can enter into this field.	Constraints
Exchange Date Optionally specify the exchange date to use to calculate the currency conversion rate. **Tip** To use the system date when the record was created as the exchange date, specify the field's creation date as the exchange date.	Exchange Date

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 CRM object includes the following fields to assist with currency conversion. These fields are automatically added to a CRM object, provided that the object's CRM application allows the creation of currency fields, and are derived from the CRM 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 CRM 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 CRM application recalculates the currency conversion rate for the record.

Fixed Choice Lists: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width **Note** The size of the field depends on the longest value of the strings in the choice list.	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints
Fixed Value **Tip** If this choice list allows multiple values, you can still write an expression to preselect multiple values by default. For example, if the field is comprised of three lookup codes with (Code,Label) of (S,Small), (M,Medium),(L,Large),(XL,Extra Large), then to preselect the Small and Extra Large selections by default, set the default value to the literal string (without quotes): **S,XL**.	Default Value

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

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

Using the List of Values Region

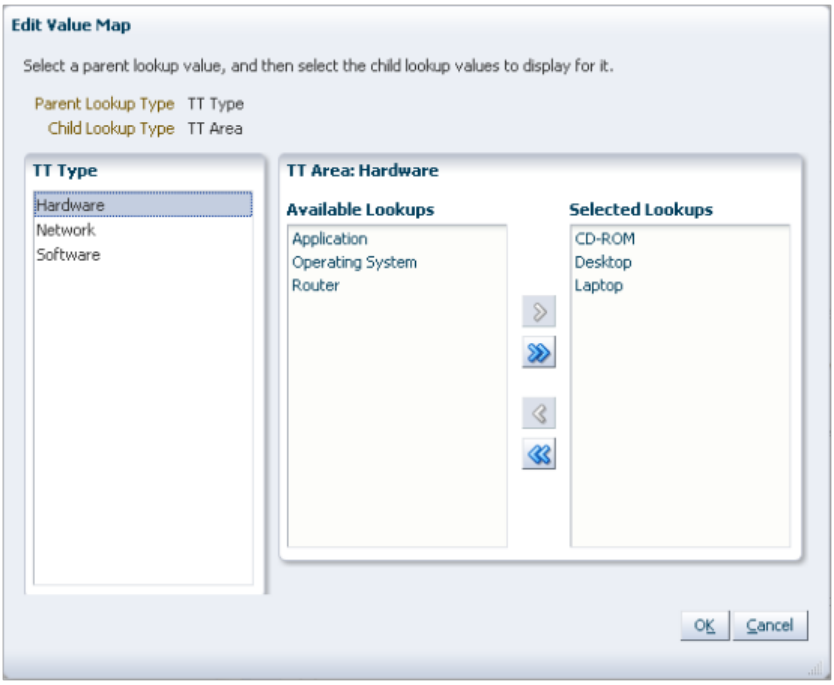
The values in a fixed choice list are populated from FND lookup types. Select the lookup type whose values you want to display in this choice list. Selecting the lookup type is possible only during field creation.

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 CRM 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.

Additional Fixed Choice List Specifications

Additional specifications for this field type include the following details:

- Data type is VARCHAR2 (1500).

Dynamic Choice Lists: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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 desktop 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:

Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name
Required	Constraints
Updateable	Constraints
Searchable	Constraints

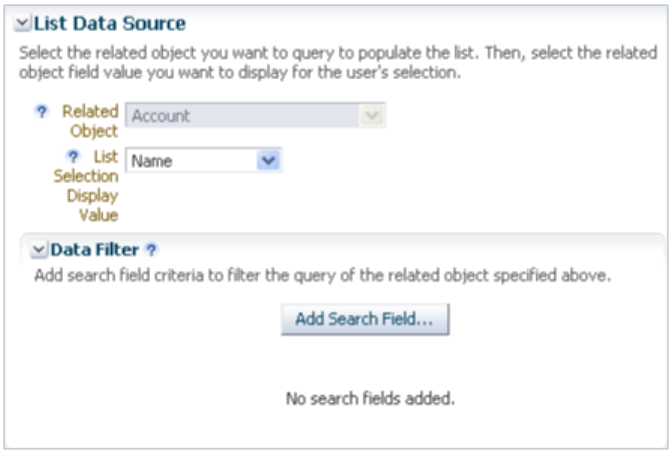
The following properties are unique to only certain field types, including dynamic choice lists:

Field Property	Field Property Region
Related Object	List Data Source
List Selection Display Value	List Data Source
Data Filter	List Data Source
Additional List Display Values	Additional List Display Values
Additional List Search Fields	Additional List Search Fields

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



- 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, 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

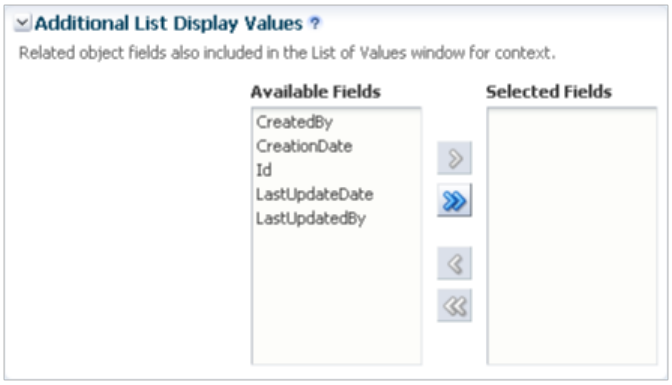
  The **List Selection Display Value** choice list is the related object's field that is displayed within the dynamic choice list as the first column at runtime. This is the primary field on the related object that your users will use to make the appropriate selection. In our example, the field might be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

- 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



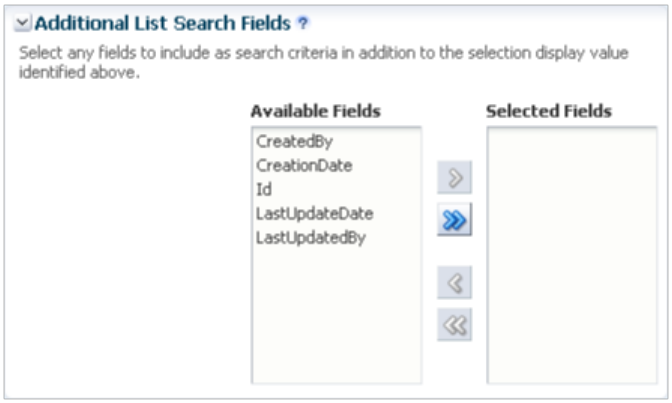
You can further refine the look and feel of the dynamic choice list by selecting additional fields to display in the choice list.

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 CRM object and a common object: resource, organization contact, organization profile, 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).

Formula Fields: Explained

Using the Oracle Fusion CRM Application Composer, you can extend a CRM 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 CRM 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:

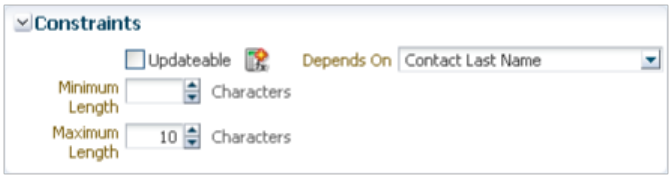
Field Property	Field Property Region
Label	Appearance
Help Text	Appearance
Display Width	Appearance
Name	Name
Description	Name

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

Field Property	Field Property Region
Formula Type Specify the field's data type, such as text, number, or date. The type can be specified only during field creation.	Field Value Type
Display Type If the formula type is **Text**, then indicate if you want this formula field to render in the run time application as a simple text box. Or, indicate if the field should allow multiple lines where text can wrap.	Appearance
Depends On	Constraints

Using the Expression Editor and the Depends On Choice List

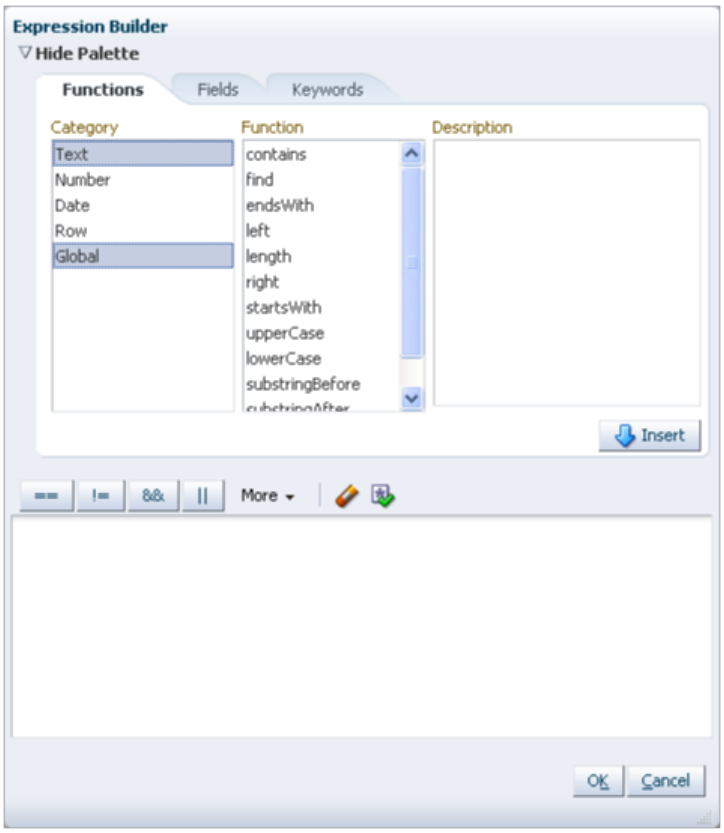
Use the **Depends On** choice list to indicate if the field should be automatically recalculated (using the expression you write) if another field's value changes.



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 editor 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.

Defining Pages: Explained

After you extend a CRM application's object model using the Oracle Fusion CRM Application Composer, your next step is to expose those new objects and fields to users. To expose new objects and fields to users, use the Desktop Pages tab. This tab lets you create new pages and customize existing pages. Customizing and creating desktop pages in CRM 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 CRM 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 Enterprise 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 Fusion CRM 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 CRM 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 CRM 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 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 desktop 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 desktop 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.

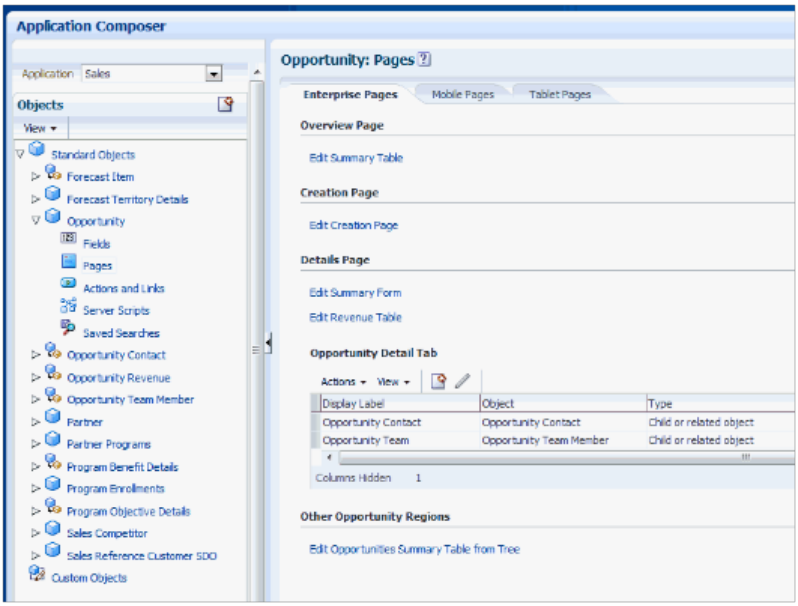
The details page can also display information related to the object record in subtabs. For example, the details page for an opportunity could include a subtab that lists customer contacts or previous orders.

Note

Some CRM objects, also known as common objects, do not have a standard work area. These include common components (note, interaction, task) and common objects (resource, organization contact, organization profile, address).

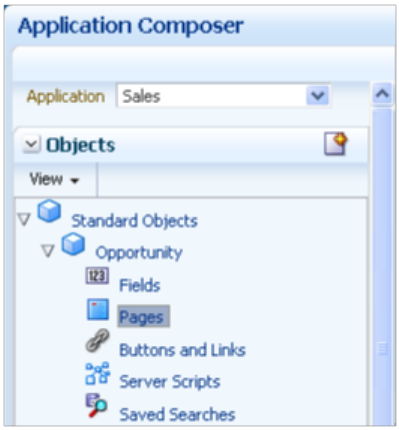
Using the Pages Overview Page

The Pages overview page in Application Composer provides an overview of the set of standard desktop 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

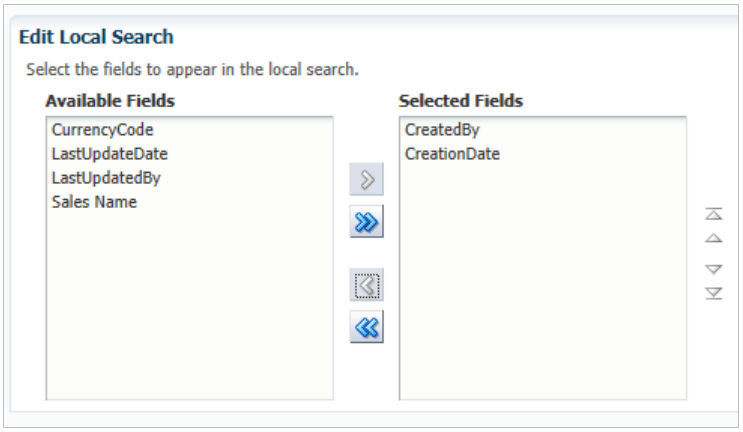
  Select which panes to display in the regional area. You can also specify if you want the regional area and individual panes to be expanded (or collapsed) by default.

- 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 fields selection, one of the work area configuration pages in the Application Composer.



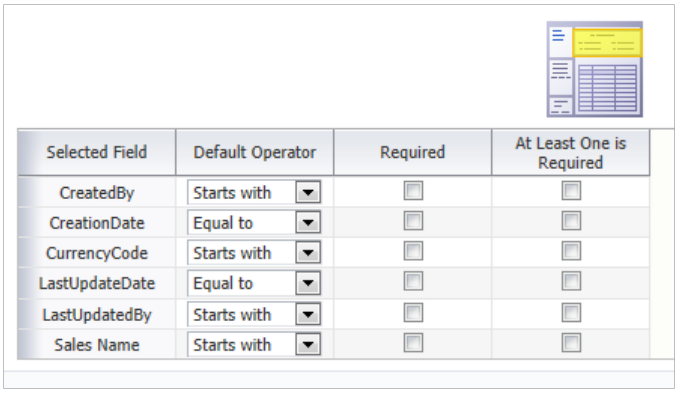
1. Select the fields that will be used to search in the local search region.

   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 fields operators, one of the work area configuration pages in the Application Composer.

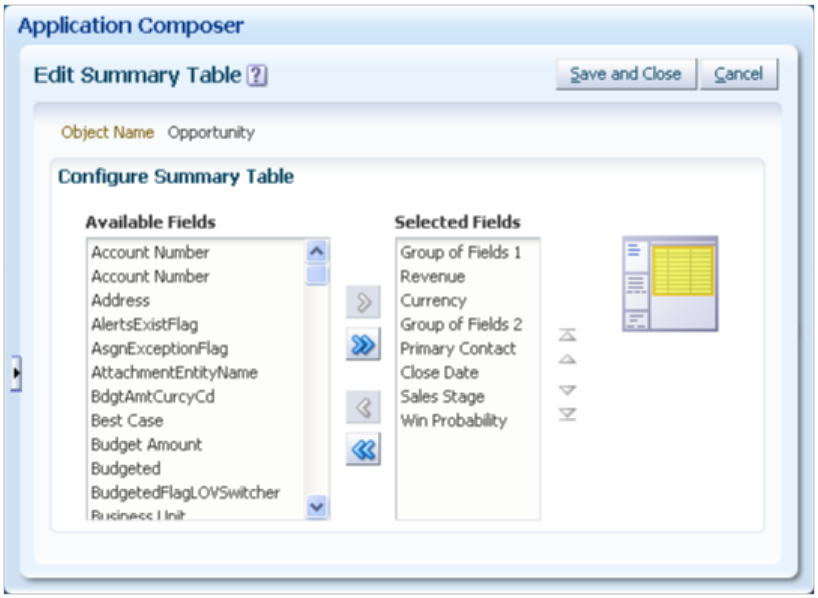


Selected Field	Default Operator	Required	At Least One is Required
CreatedBy	Starts with		
CreationDate	Equal to		
CurrencyCode	Starts with		
LastUpdateDate	Equal to		
LastUpdatedBy	Starts with		
Sales Name	Starts with		

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

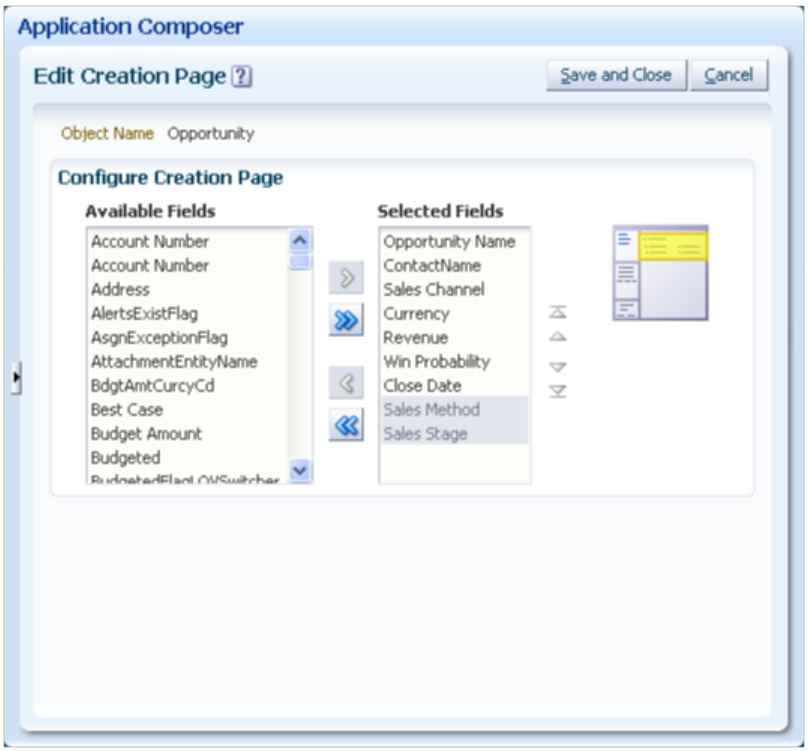
   - Required: The user must include this field in a search at run time.
   - At Least One is Required: The user must include at least one of the selected fields in a search at run time.
   - Default Operator: The user can define each search field value by using one of the listed operator options. For example, 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 buttons to the summary table, if you previously created them.

- Creation Page

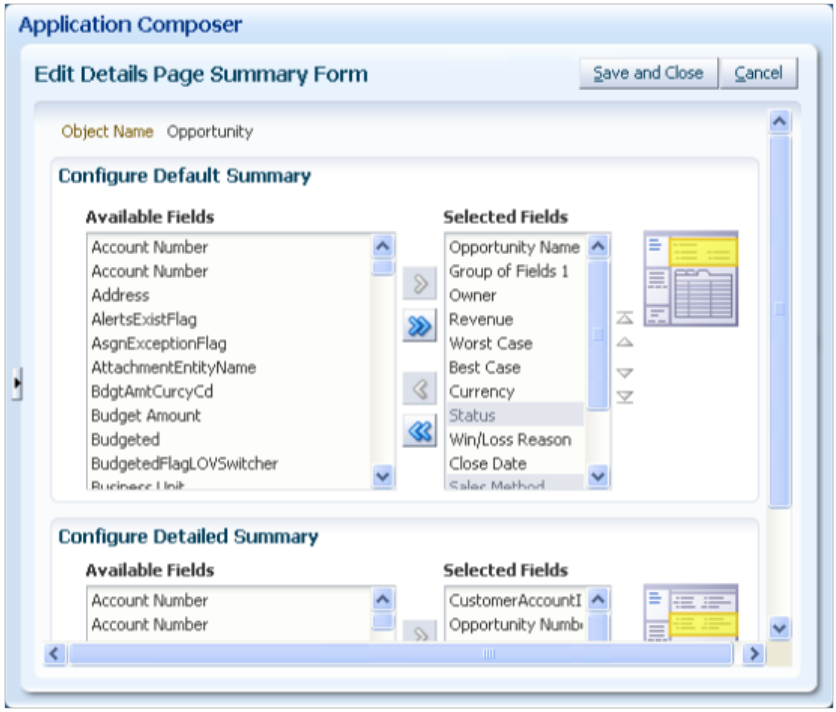


a. Select the fields that you want to display on the object's creation page.

   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



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 Fusion CRM Applications. 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. In the Pages 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.

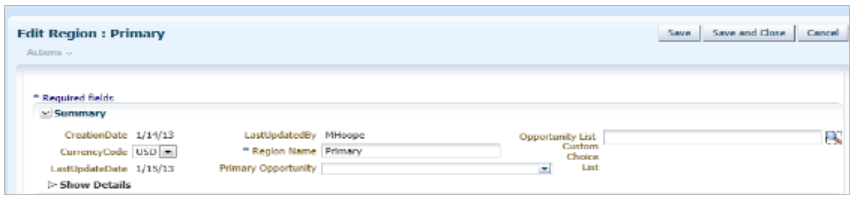
d. In the Configure Picker Table list, select the fields that you want to appear in the search results table.

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.

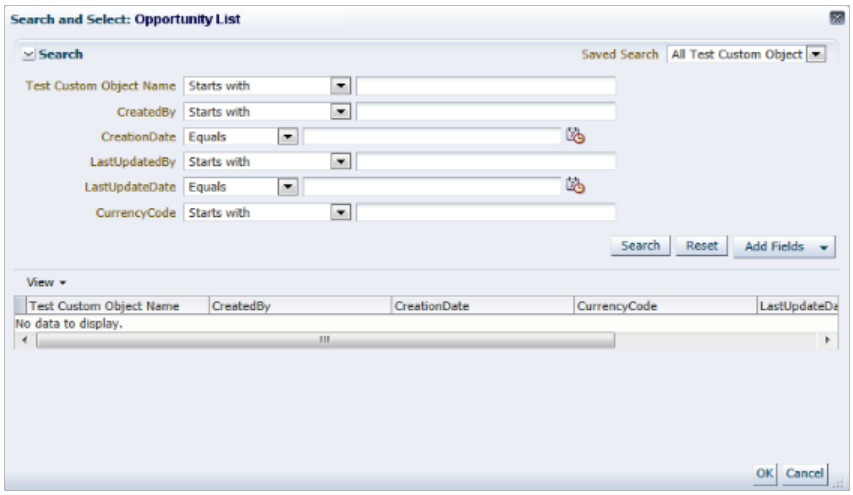
b. In the Overview page, click the object name.

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



c. In the Edit object page, 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.



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**.

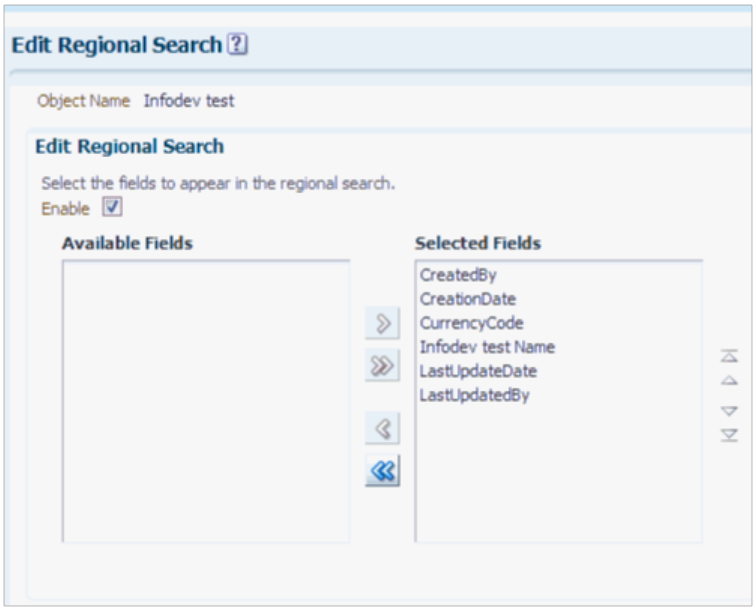
c. Select the record.

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 CreationDate 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 CRM 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 Trading Community address object has configuration pages for customizing the overview page, creation page, and address details form. The Trading Community organization profile 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 CRM 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.

The 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 desktop 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 the 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 CRM application, you cannot use Page Composer to edit those pages again.

Note

The exception is the customer profile in Oracle Fusion 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 CRM 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 CRM object can be displayed on a work area, which consists of an overview page, a creation page, and a details page. The Oracle Fusion CRM 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.

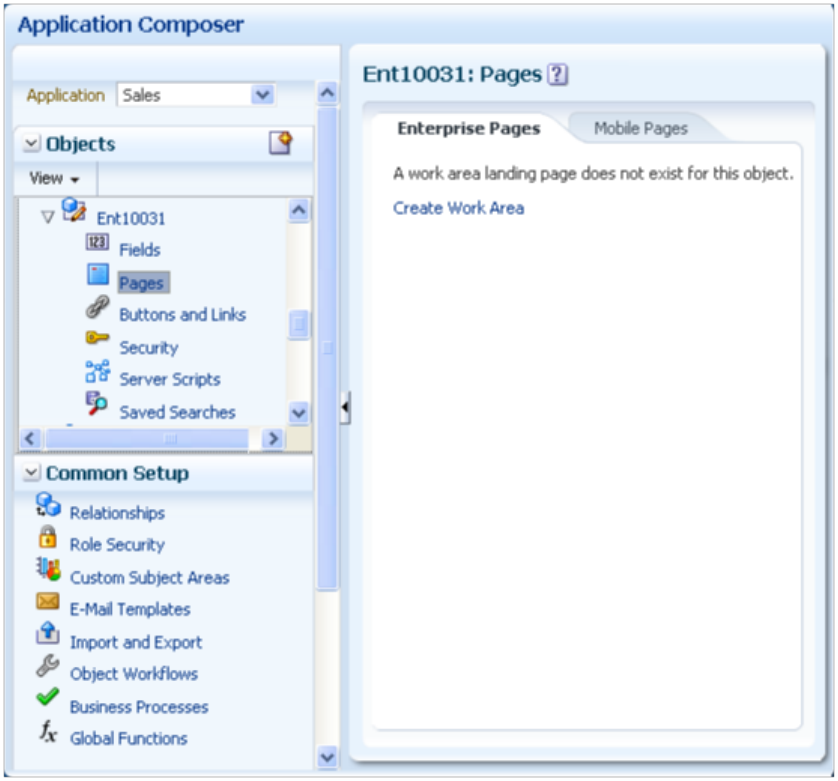
Review these aspects of the work area creation process in the Application Composer before you create a new work area for a custom object:

- 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.

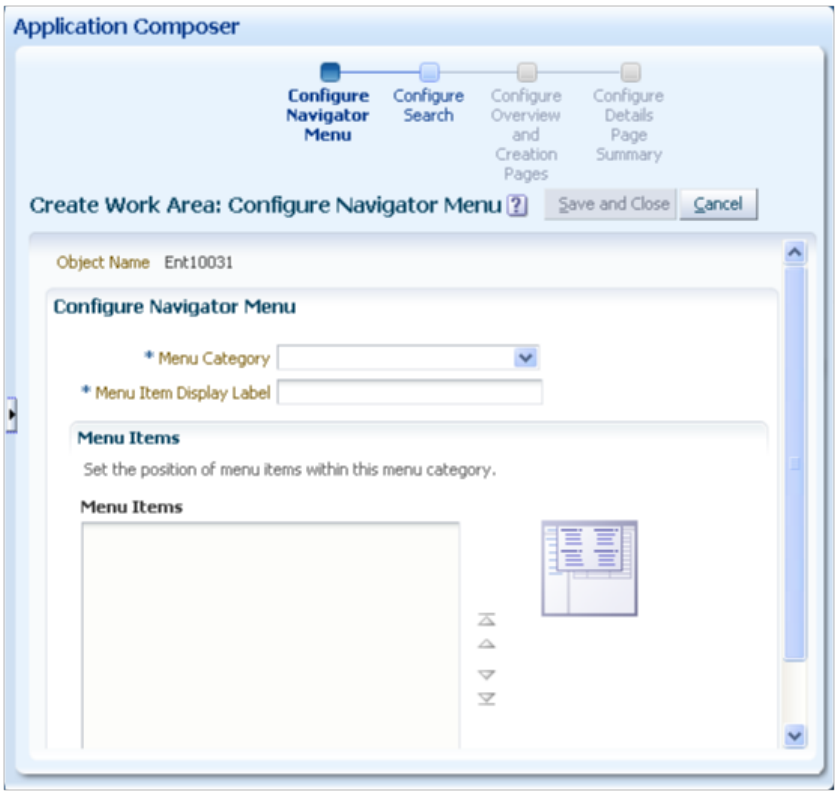
Note

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 the 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

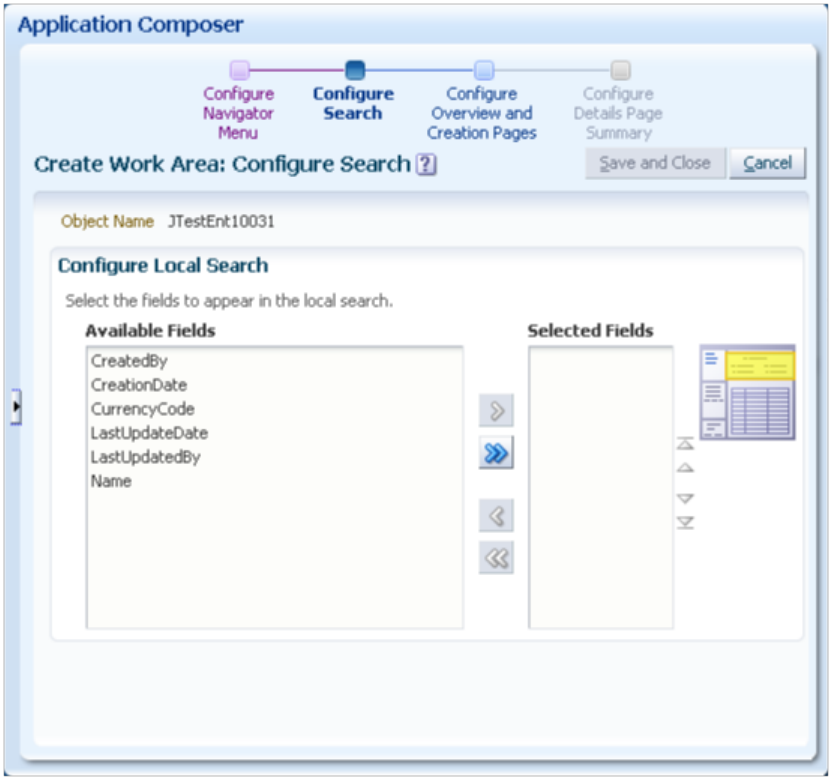
Extending the Regional search is part of the first step in creating a work area. By adding custom parameters, you can expand the search parameters of the Regional area for both custom and out-of-the-box objects in a Fusion application at run time.

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

Select the fields that you want to display as search criteria in the local search region. The local search region appears above the summary table on an object's overview page. Adding fields to this region is optional.



1. Select the fields that you want to display as search criteria fields in the local search region.

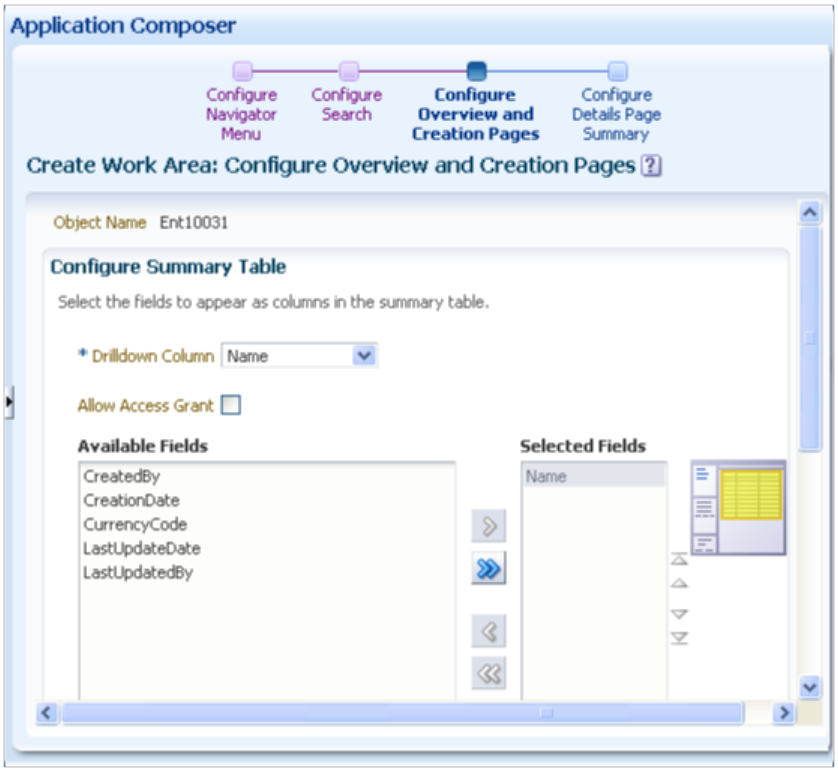
   The list of fields available for selection is displayed to you in a single column, although the local search region is formatted as a region with two columns. The first field you select is displayed in the first column, the second field you select is displayed in the second column, the third field you 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.

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. Select the **Allow Access Grant** check box if you want users to be able to grant access to a record in the summary table to another user, at run time.
4. Add custom buttons to the summary table, if you previously created them.
5. 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.

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.

   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 CRM 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 Fusion CRM 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 Oracle Fusion CRM Application Composer.

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

- Using the Details page
- Adding subtabs
- 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 Fusion CRM 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 desktop 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.

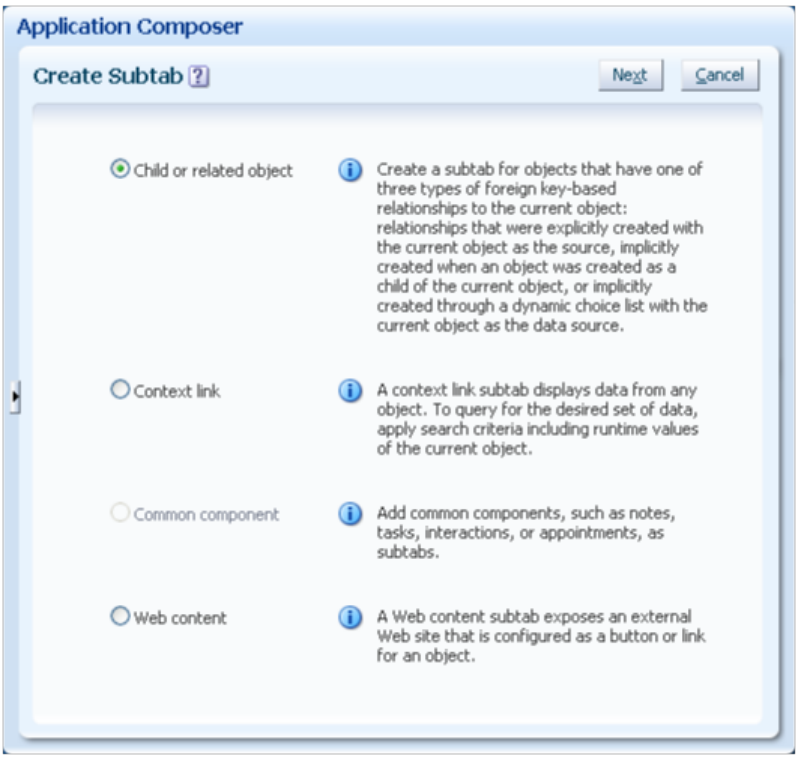
To add a subtab to an existing details page:

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.



Tip

To hide subtabs that you previously added using Application Composer, use Page Composer.

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.

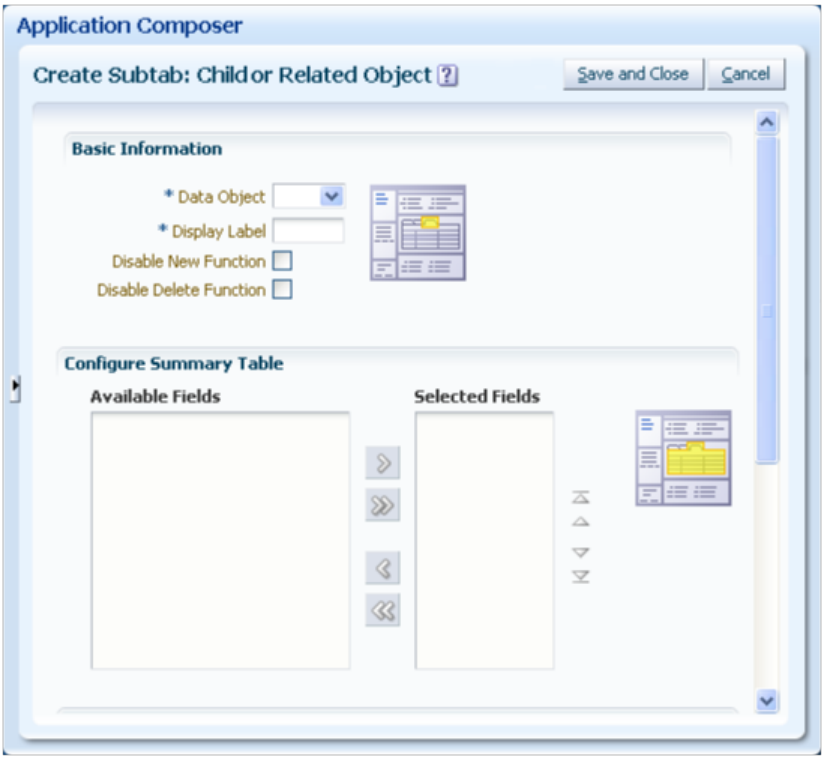
  Using our previous example, perhaps you don't need to display an HR Representative choice list on a department desktop page, but you still want to add a department subtab to an employee's details page. In this case, manually create a reference relationship between the employee and department objects 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 Fusion CRM 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. For child object subtabs only, specify if you want to display the **Manage Permission** button on the subtab at run time.

      At run time, users can select an object record and click that button to specify the level of access another user should have to the selected record.

   d. 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.

   e. 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.

---

   f. 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.

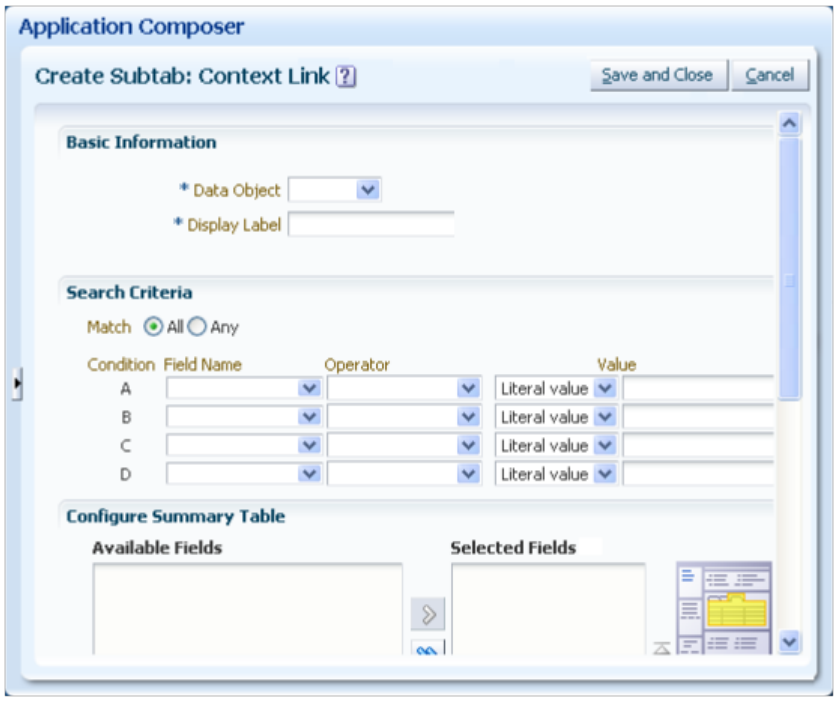
If the subtab's object is a related object, then users can associate an existing record of the subtab object to the master object of the page. However, to create new related object records, users must do so in the object's own creation page.

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 Fusion CRM 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 an 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.

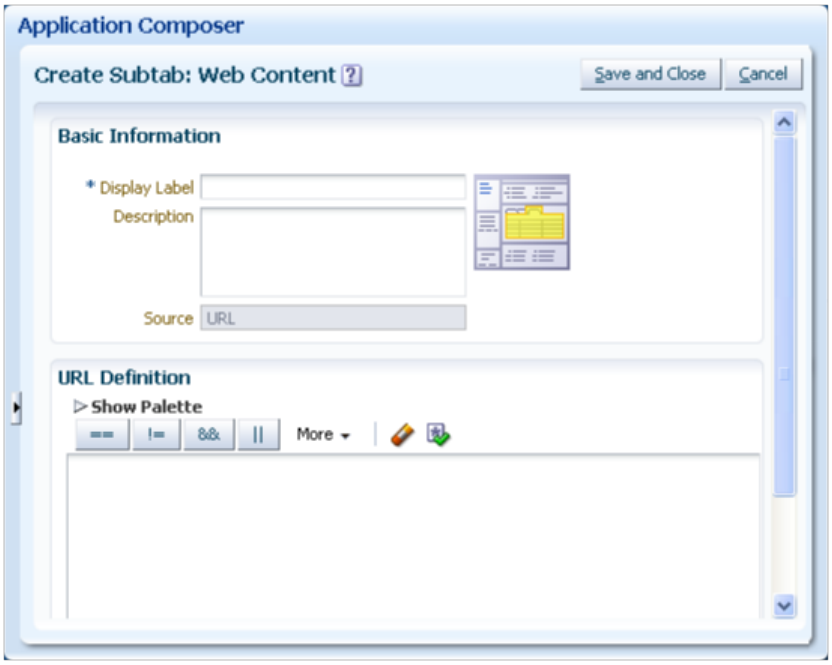
Web Content Subtabs

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 subab 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 editor 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 CRM standard objects, such as the Sales Account Profile 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 Oracle Fusion CRM application altogether. Add a tree node to a standard object's tree from that object's Pages Overview page in the Oracle Fusion CRM Application Composer.

Review these aspects of the tree node creation process in the 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 Fusion CRM 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 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:

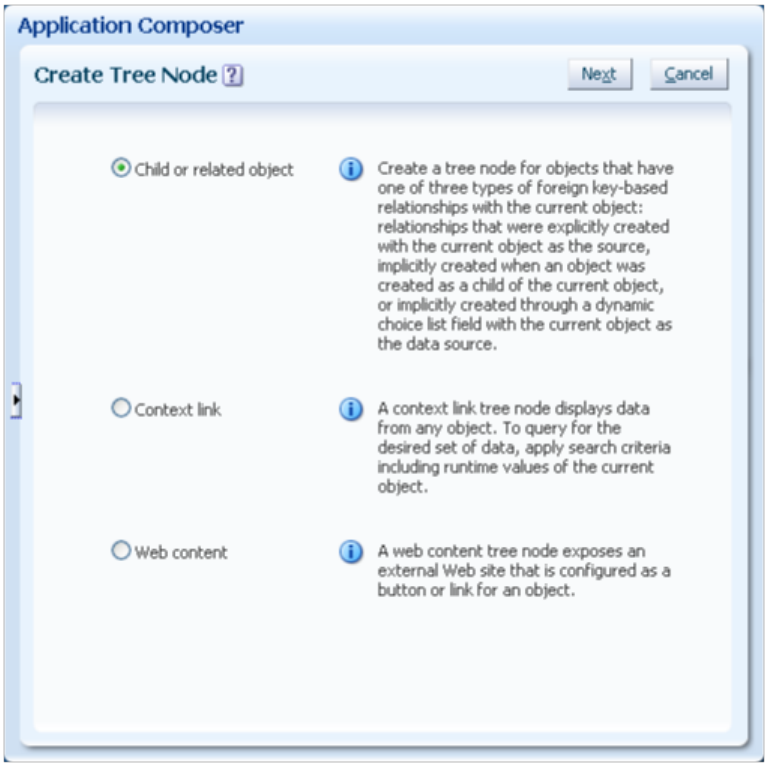
1. Select an application on the main Overview page.
2. Select a standard object, either the Sales Account Profile or Partner object, in the object tree.
3. Select the **Pages** node.

Note

Only the top-level objects, Sales Account Profile 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 the 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.

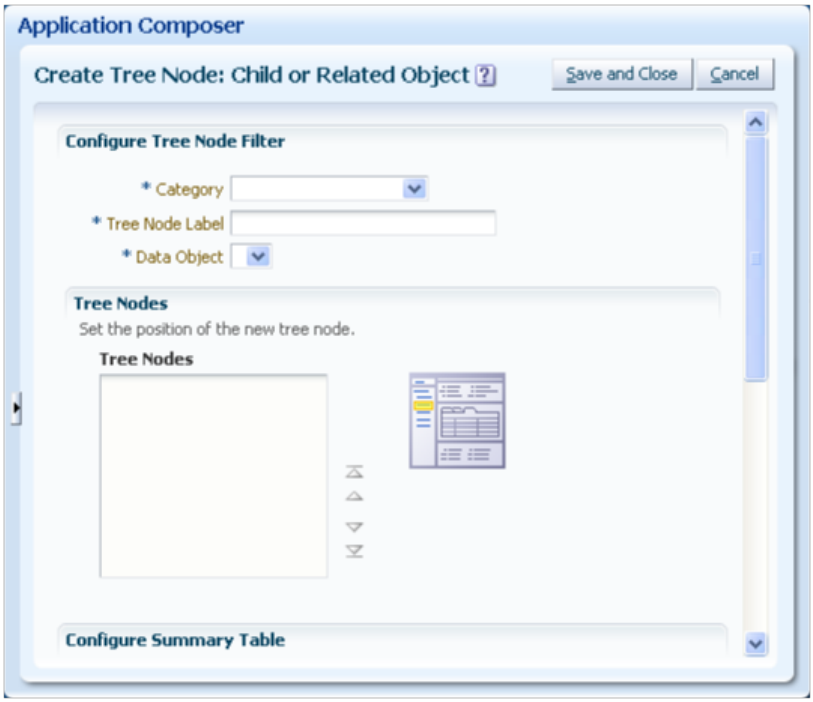
The Application Composer lets you add a tree node to an object's tree for either a child object or for three types of related objects. These objects exist in four types of one-to-many relationships, which are described in detail in the related topic about object relationships:

- 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. For child object tree node pages only, specify if you want to display the **Manage Permission** button on the tree node page's summary table at run time.

      At run time, users can select an object record and click that button to specify the level of access another user should have to the selected record.
   f. 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.

g. Select which buttons you want to display on the tree node page at run time.

Note

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

You cannot add buttons to a tree node page for the Sales Account Profile object.

---

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

You can configure fields for the detail form that appears under the summary table. If the tree node's object is a child object, then users can enter child object data into this detail form at run time. Always include required fields in this section.

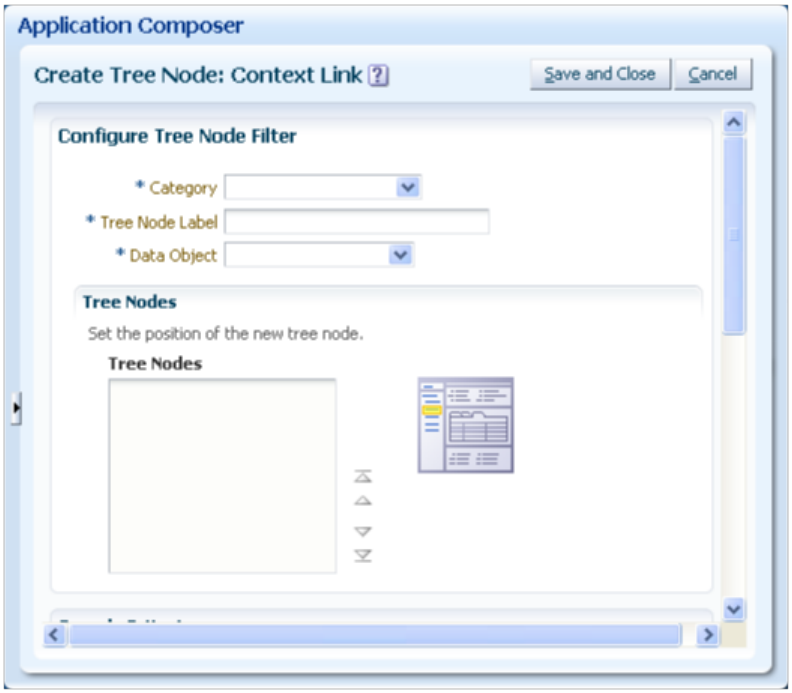
If the tree node's object is a related object, then users can associate an existing record of the tree node object to the master object of the page. However, to create new related object records, users must do so in the object's own creation page.

Context Link Tree Nodes

A context link tree node page displays a filtered list of records from any top-level object, where the filter is often based on the run time values from the current object. The object does not have to be related to the current object. Context link tree node pages are read only.

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

1. On the Pages Overview page, click the **Create Tree Node** icon.
2. Select **Context Link**.



3. On the Context Link tree node configuration page:

   a. Select the tree node category and enter the tree node label.

   b. Enter the name of the tree node filter.

   c. Select the object that is to be exposed on the tree node page.

   d. Set the position of the new tree node.

   e. 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.

---

   f. 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.

   g. 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.

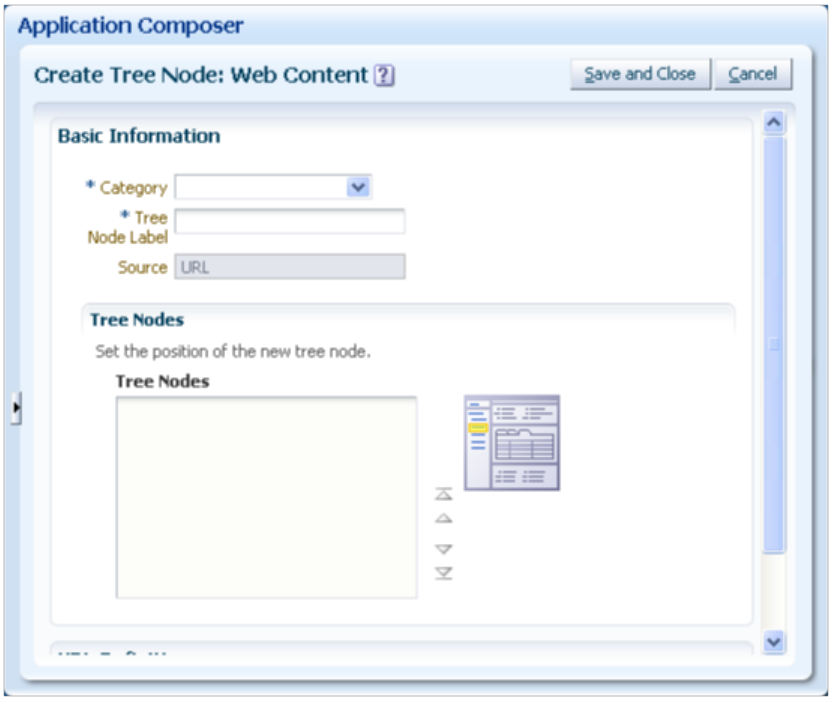
Web Content Tree Nodes

A Web content tree node page exposes an external Web site on desktop page. 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 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 editor 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)
```

Actions and Links: Explained

In Oracle Fusion CRM 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.

Actions

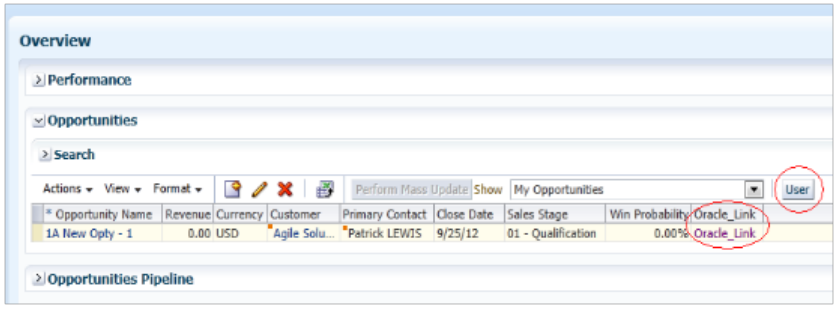
An action can be based on a script (a Groovy method that is defined on the object) or a URL. When configuring the work area for a standard or custom object, you can add custom links or actions to a page-level or task-level Actions menu or as a toolbar button. You can also manage the Actions menu by hiding 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. A button can perform an action or navigate the user to another page in the runtime application, or to another web site. For example, you might want to 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 runtime to create a new type of record from a selected row, such as escalating an existing "trouble ticket" to a more severe "case" that can be managed separately. After you create an action, it can be exposed as a button or an option on the Action menu. After you create a link, it can be selected as a field.

Adding Actions or Links

You add actions or links in two steps:

1. Define an action or link for an object.
2. Use the Oracle Fusion CRM 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.



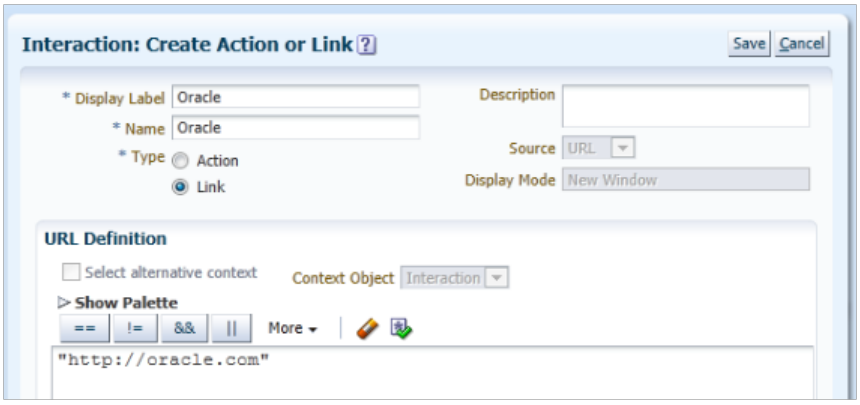
To define an action or link for an object:

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..

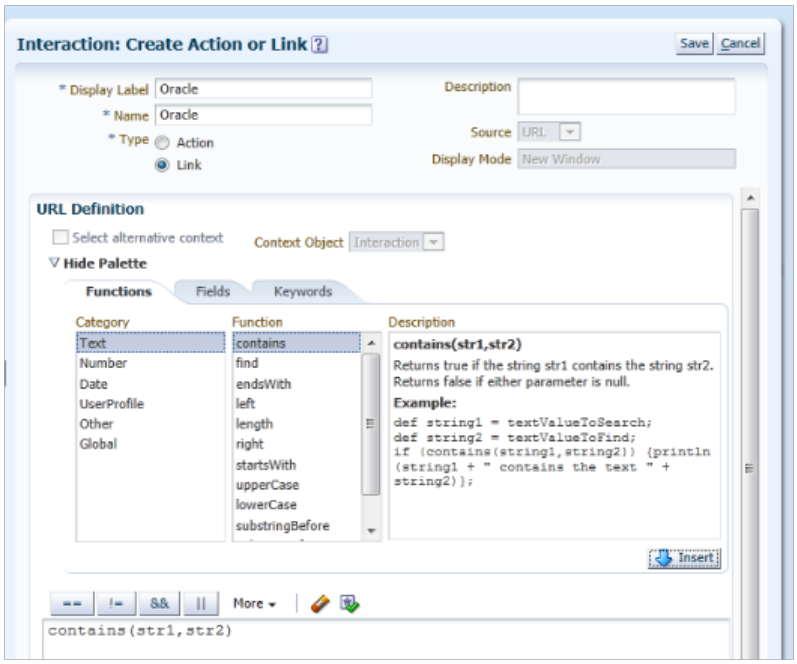


If the source is a URL, you can enter a static URL, enclosed in double quotation marks. Or, you can define the URL by using the expression editor, which provides access to this object's fields to assist you in constructing the URL. If this object has a parent or relationship with a source object, then 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 content** check box.

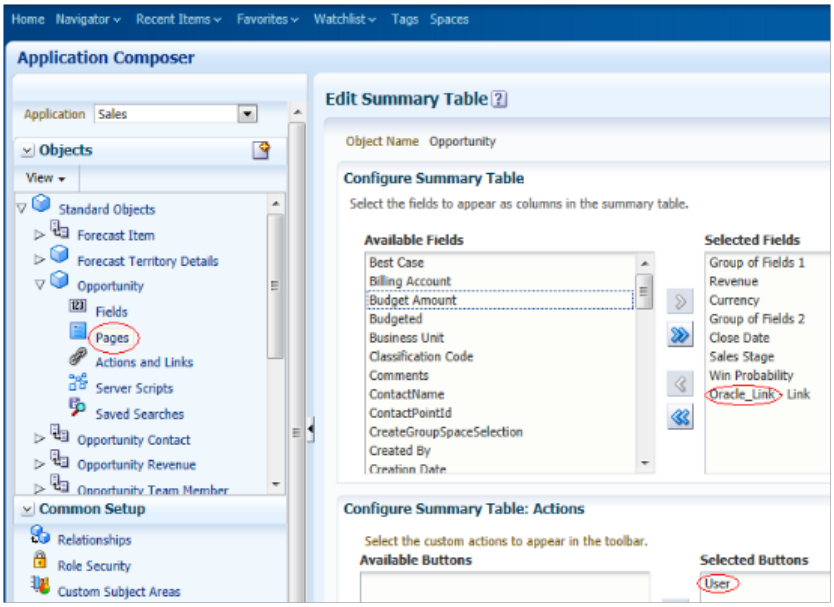
The following figure shows a script in the URL Definition window.



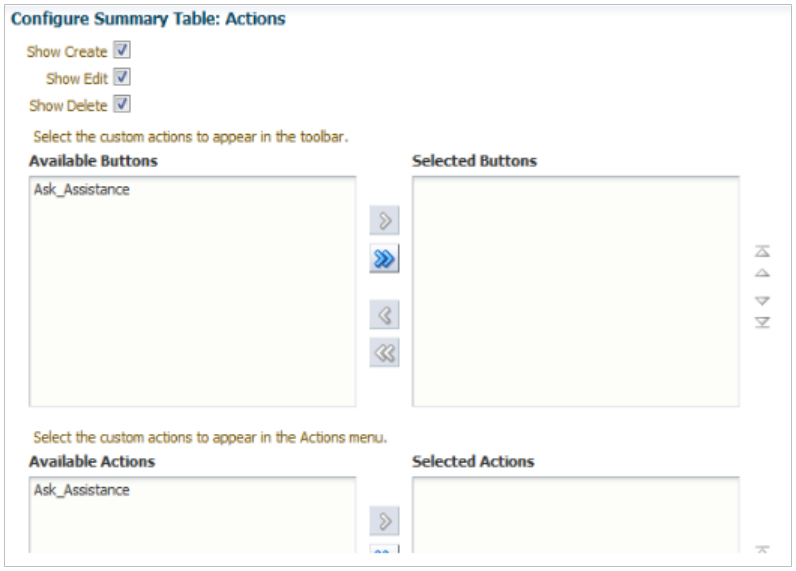
If the source is a script, you can either select a predefined object function from the **Method Name** choice list, or create a new object function using the expression editor.

After you save actions or links, you can expose them on UI pages by configuring the Applications Composer options available in the Edit Summary Table page in the Pages node of an object.

The following figure shows a selected button and fields in the Edit Summary Table page in the Pages node of Applications Composer.



For example, the Edit Summary Table could have links in the fields selector with standard or custom fields because at runtime 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 the actions and links, in the Edit Summary Table you could see both the available and selected buttons and actions, or one, or none.



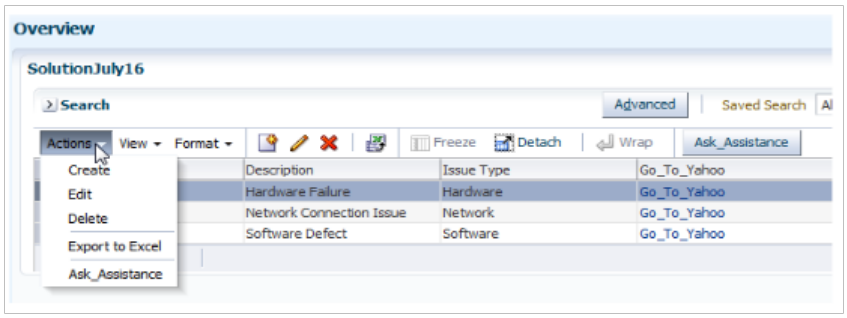
The figure above shows the Configure Summary Table: Actions with options checked for the Show Create, Show Edit, and Show Delete options on the Action menu. It also shows a custom button and a custom action.

After you define an action, you can then expose it as a button or an Action 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.



The figure above shows the 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 CRM 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 a CRM 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, Oracle Fusion CRM 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 the Oracle Fusion CRM Application Composer before you begin to view or edit saved searches for CRM 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 CRM 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 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.

Field Type	Available Operators	Available Values
• Number • Check box • Percentage	• Equal to • Greater than • Greater than or equal to • Is blank • Is not blank • Less than • Less than or equal to • Not equal to	Enter a literal value. **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 150percent in your saved search, you must enter 1.5.
• Dynamic choice list • Text • Fixed choice list • Currency	• Contains • Does not contain • Ends with • Equal to • Is blank • Is not blank • Not equal to • Starts with	Enter a literal value. **Tip** For fixed choice list fields, you must enter the lookup code, not the lookup meaning.

• Date • Date time	• After • Before • Equal to • Is blank • Is not blank • Not equal to • On or after • On or before	Enter: • Literal value • Date function Specify the number of days, months, or years after or before the current date.

- Value

  The values that you specify are applied as the search criteria against the object records at run time.

  See the previous table.

Creating Saved Searches Using Page Composer: Highlights

A saved search contains specific search criteria and settings that are captured for running the same search again later. You save the visible search fields and entered criteria, selected conditions, and search mode, either basic or advanced. Your end users can create their own saved searches that only they can see. Or, you can use Page Composer to create a saved search that can be used by a set of users, such as all users at the site or only a group of users with a specific job role.

Creating a Saved Search using Page Composer

- If you have access to customize a user interface page using Page Composer, then you can customize saved searches for other users by working on saved searches while Page Composer is open. You can also delete, rename, or change the search options for a saved search.

  See: Task: Create a Site-Level Saved Search for All Users

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 the Oracle Fusion CRM 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 the 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

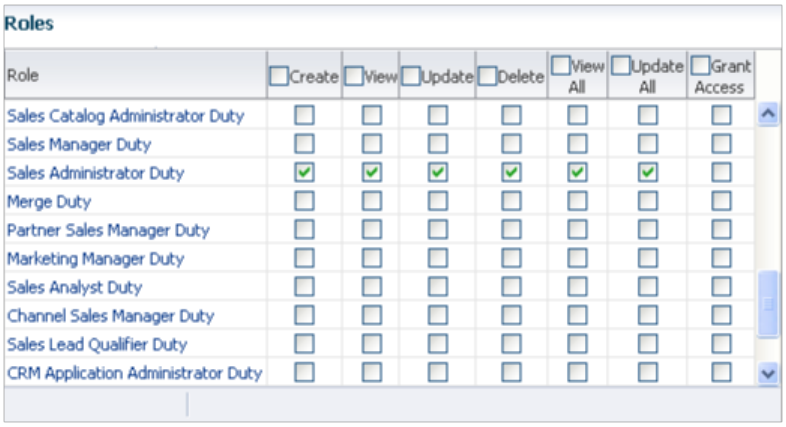
Securing Objects

The object-centric Define Policies page displays a list of the enterprise-level duty roles which map to a CRM 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.



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

- Enable function security for a role.
- Enable data security for a role.
- Allow users to grant access to a specific record at run time, to users with a given role.

Securing Roles

The Role Security page displays a list of the enterprise-level duty roles, which map to a CRM job role. Select a role and click the **Define Policies** button to

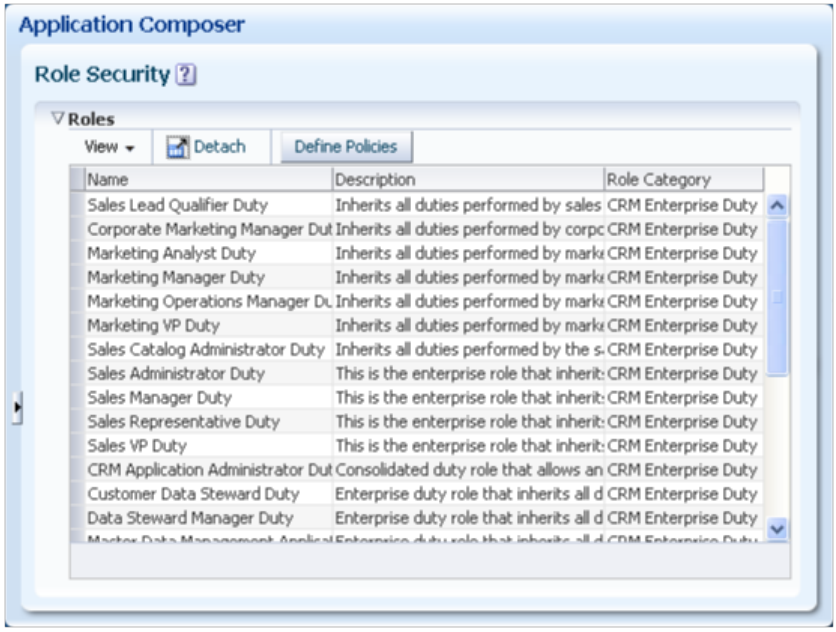
navigate to the role-centric Define Policies page, which displays a list of the custom objects for your CRM implementation. Use this page to manage access for users with the corresponding role by specifying a security policy for one or more top-level or child custom objects. When you do this, users with 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:

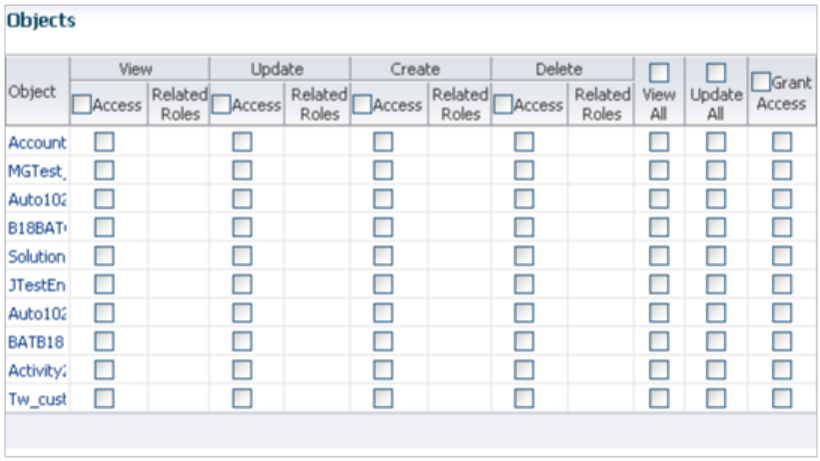
1. Select an application on the main Overview page.
2. Select the **Role Security** node from the Common Setup pane.

   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.
- Allow users to grant access to a specific record at run time, to users with a given role.
- 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.

The last column contains the **Grant Access** check box. Selecting this check box controls the display of the **Manage Permission** button on the object's summary table at run time. At run time, users with the corresponding role (or an

administrative role) can select an object record and click that button to specify the level of access another user should have to the selected record.

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. To let users delete records, you must grant users instance-level delete privileges using the **Grant Access** check box and the run time **Manage Permission** button.

Application Composer and the Oracle Authorization Policy Manager (APM)

Across Oracle Fusion applications, 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 the 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 the 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 CRM application. You must manually grant additional access to other duty roles, if desired. For example, if you create a custom object in Oracle Fusion Sales, 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 CRM applications:

Application	Default Duty Role
Oracle Fusion Customer Center	Sales Administrator Duty Marketing Operations Manager Duty
Oracle Fusion Marketing	Marketing Operations Manager Duty
Oracle Fusion Sales	Sales Administrator Duty

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:

Application	Child Objects of This Parent Object	Access Granted to These Duty Roles
Oracle Fusion Customer Center	Sales Account Profile	Sales Administrator Duty Marketing Operations Manager Duty

Oracle Fusion Marketing	Sales Lead	Marketing Operations Manager Duty Sales Administrator Duty Channel Operations Manager Duty
Oracle Fusion Marketing	Marketing Campaign Marketing Response Marketing List Marketing Treatment Marketing Event Activity Marketing Advertising Activity	Marketing Operations Manager Duty
Oracle Fusion Marketing	Marketing Budget Marketing Claim Marketing Budget Entry Marketing Budget Fund Request	Marketing Operations Manager Duty Channel Operations Manager Duty
Oracle Fusion Sales	Opportunity Sales Competitor	Sales Administrator Duty
Oracle Fusion Sales	Partner	Channel Operations Manager Duty
Oracle Fusion Common CRM	Trading Community Org Contact Trading Community Resource Profile Trading Community Organization Profile Trading Community Address	Customer Data Steward Duty

Importing and Exporting Custom Objects: Explained

To support the importing and exporting of the custom objects that you created with the Oracle Fusion CRM Application Composer, you must first generate the object artifacts required for both file-based import and bulk export.

Oracle Fusion Import and Export Processes

In Oracle Fusion, 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 CRM 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 the 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, 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 Fusion CRM product-specific documentation for additional details on how Oracle Fusion CRM products enable 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 an Oracle Fusion tool you can use to modify Oracle Fusion 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 CRM 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 CRM, 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).

Use Cases	Design View - Standard mode	Design View - Direct Selection mode	Page Type
Add content (Business Intelligence reports, CRM portlets such as Calendar)	Yes	No	Dashboard and other select pages
Delete region	Yes	No	Dashboard and other select pages
Move region	Yes	No	Dashboard and other select pages
Change page layout (for example, change a two column layout to three column layout)	Yes	No	Dashboard and other select pages
Default region state (open or close)	Yes	No	Transactional pages (all non-dashboard pages)
Manage save queries (create and edit)	Yes	No	Transactional pages (all non-dashboard pages)
Hide or show field	No	Yes	Transactional pages (all non-dashboard pages)
Change field label	No	Yes	Transactional pages (all non-dashboard pages)
Make field required or not	No	Yes	Transactional pages (all non-dashboard pages)
Make field read-only or updateable	No	Yes	Transactional pages (all non-dashboard pages)
Reorder fields in a Form	No	Yes	Transactional pages (all non-dashboard pages)
Reorder table columns	Yes	Yes	Transactional pages (all non-dashboard pages)
Hide or show table columns	Yes	Yes	Transactional pages (all non-dashboard pages)
Set table column width with the mouse	Yes	No	Transactional pages (all non-dashboard pages)
Set table column width and 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)

The Oracle Fusion CRM Application Composer also lets you make UI changes at run time. However, the types of UI changes that you can make using the Application Composer are quite different. Specifically, your primary focus when using the 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 ability to make these types of object model extensions is available only in Oracle Fusion CRM applications. Also, using the Application Composer, you cannot access the Resource Catalog to add new content to a page. With the Application Composer, administrators can make customizations at the site level only.

The following table describes some of the primary differences between Page Composer and the Application Composer.

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)
Select the MDS layer where you want to author customizations, such as at the site layer or job role layer	Yes	No
View results of customizations immediately	Yes, in the Page Composer design interface	Yes, in the CRM application that you are customizing

5

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 Fusion CRM. Certain standard objects offer a set of simplified pages, some of which you can customize using Oracle Fusion CRM 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
- What you can extend on the simplified pages for customers
- What you can extend on the simplified pages for leads
- What you can extend on the simplified pages for opportunities

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 Fusion CRM Simplified Pages: Explained

Oracle Fusion CRM 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 using the Oracle Fusion CRM Application Composer and also Page Composer. For example, you can show or hide fields, rearrange fields, and add custom fields to simplified pages.

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 Fusion CRM 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.

Object	Available Simplified Pages	Extensible?
Opportunity	Opportunity work area Opportunity Contact subtab Opportunity Team subtab	Yes
Lead	Lead work area Lead Contact subtab Lead Team subtab	Yes
Customer and Consumer	Customer work area Consumer work area	Yes
Contact	Contact work area	Yes
Tasks	Tasks page	No
Interactions	Interactions page	Yes
Appointments	Appointments page	No
Notes	Notes page	The Create Note and Update Note pages are extensible. The Notes overview page is not extensible.

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 add new subtabs and the ability to customize the search region is not available.

Simplified pages are available only for a select group of standard Oracle Fusion CRM 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 Fusion CRM simplified pages.

- Hide or show standard fields
- Extend form regions
- Extend table regions
- Reorder fields (tables and pages)
- Change field labels

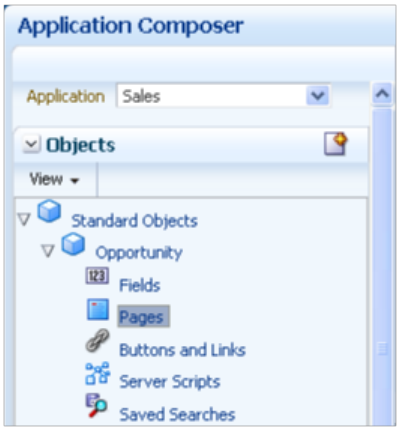
Use Page Composer to:

- 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.

Extending Simplified Pages for Contacts: Explained

Using Oracle Fusion CRM 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 Trading Community Person Profile 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:

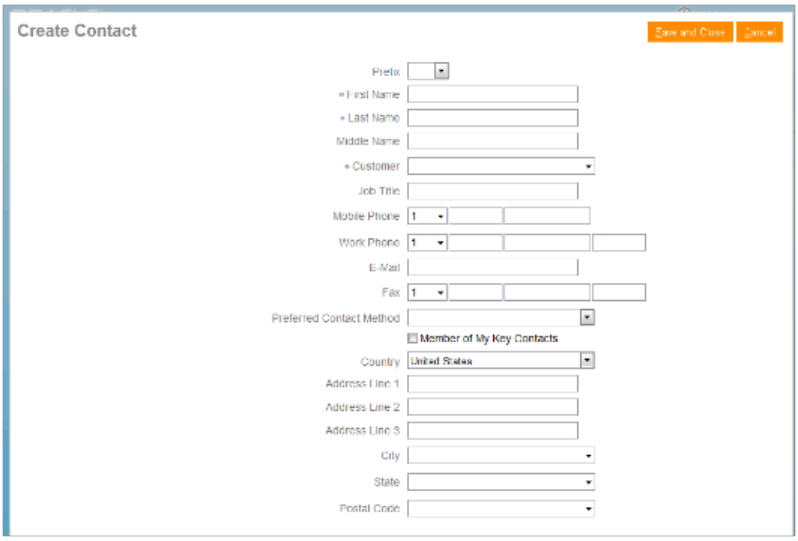
- Customer Contacts overview table
- Contact Overview page

Create Contact Page

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

- Hide or show existing fields that belong to the Trading Community Person Profile object.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).

  Custom fields that you add to the Trading Community Person Profile 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.

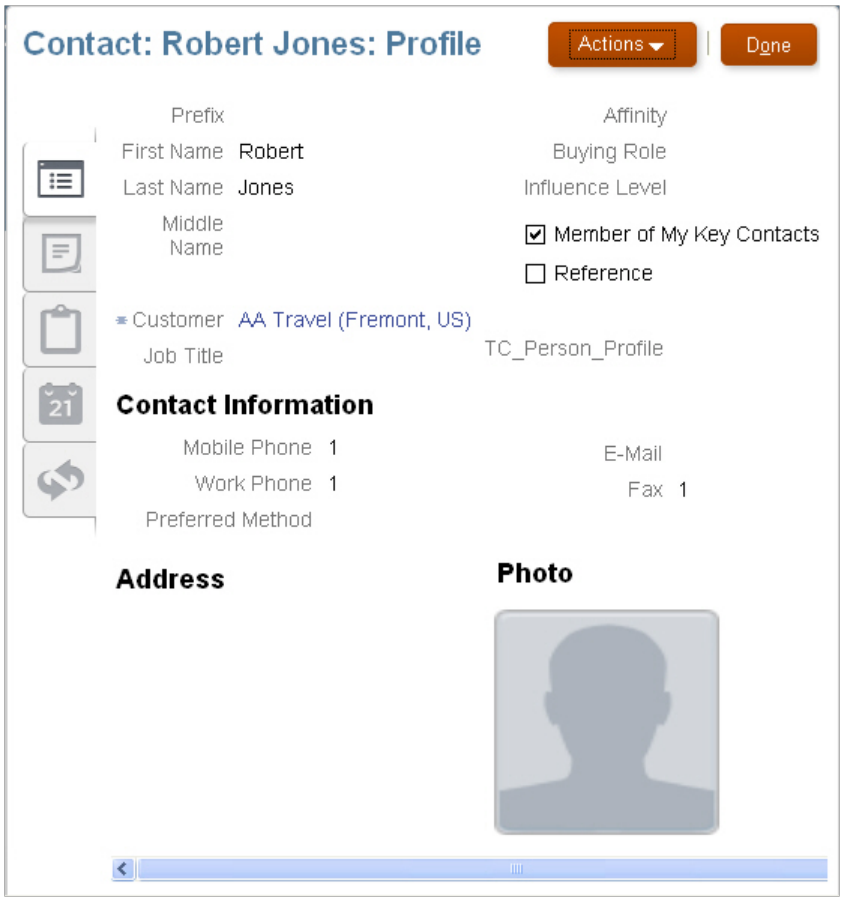
Contact Detail Page (Profile Page)

You can extend the following items on the Contact detail page (Profile page):

- Hide or show existing fields that belong to the Trading Community Person Profile object.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

  Custom fields that you add to the Trading Community Person Profile 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 Oracle Fusion CRM 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 Trading Community Person Profile and Trading Community Organization Profile 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.

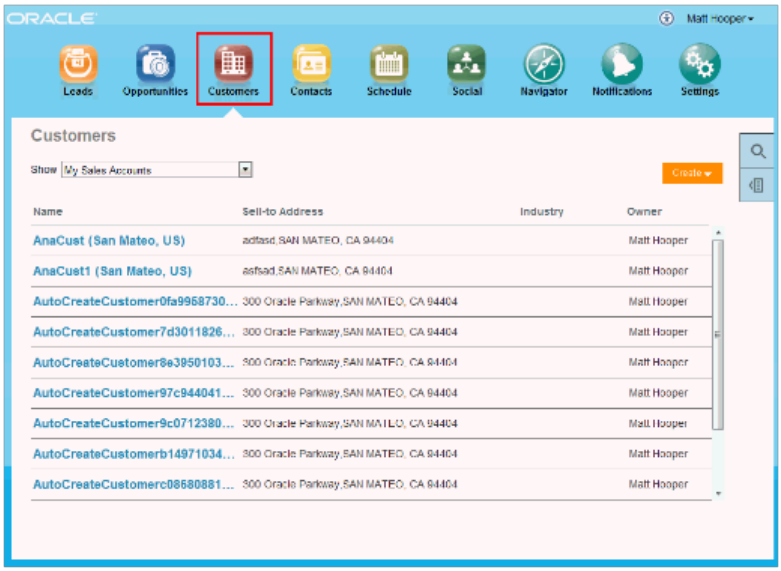
Note that these pages are not extensible:

- 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.



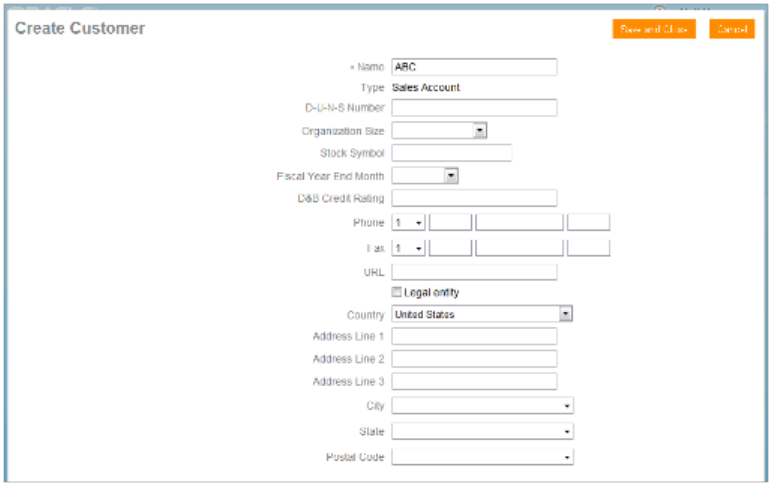
Note that you cannot make changes to these components on the page:

- 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 Trading Community Organization Profile and Sales Account objects.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
  - Custom fields that you add to the Trading Community Organization Profile 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 Trading Community Organization Profile object in Application Composer to extend this region.

- Sales Account region

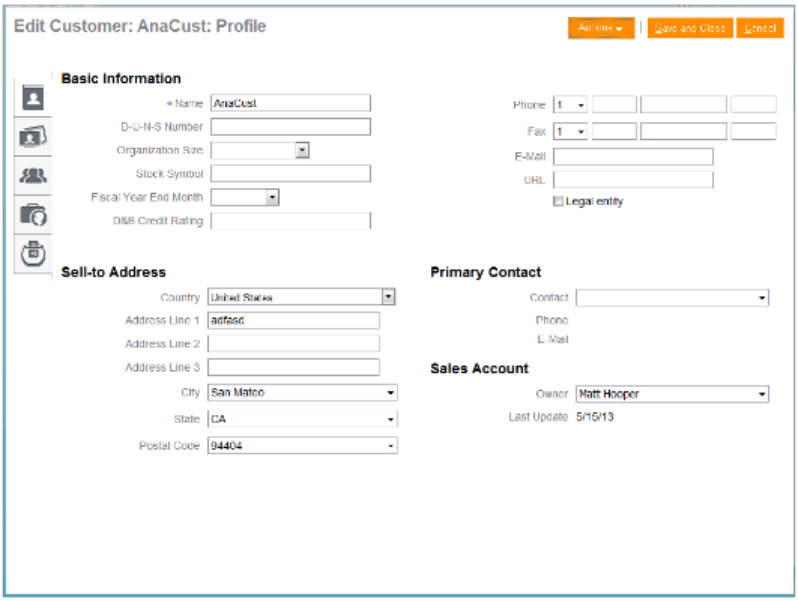
Use the Sales Account object in Application Composer to extend this region.

- 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:

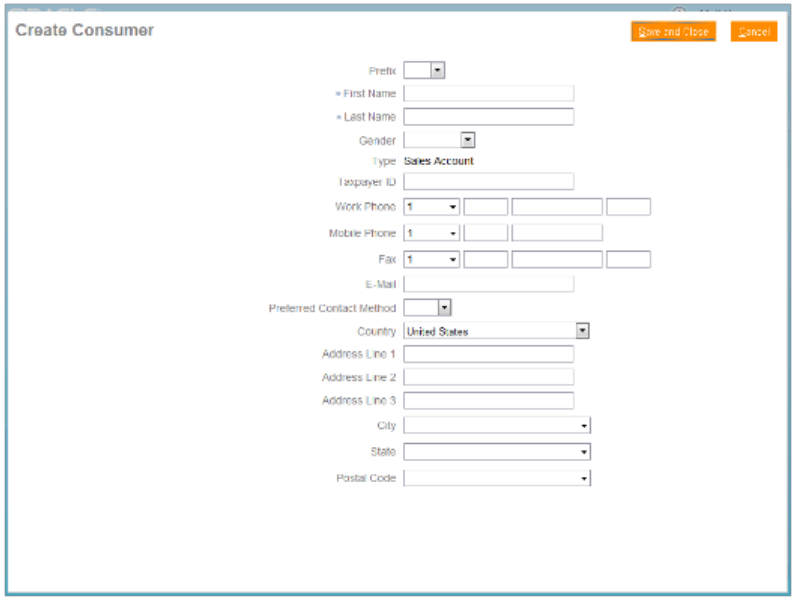
- Subtab regions
  - Contacts
  - Opportunities
  - Leads
- Actions button
- Custom buttons and actions
- You cannot create new subtabs.

Create Consumer Page

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

- Hide or show existing fields that belong to the Trading Community Person Profile and Sales Account objects.
- Reorder fields.
- Change field labels.

- Add custom fields (all types).
  - Custom fields that you add to the Trading Community Person Profile 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 Trading Community Person Profile 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 Oracle Fusion CRM 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 overview table
- Create Lead page
- Edit Lead page (Summary page)

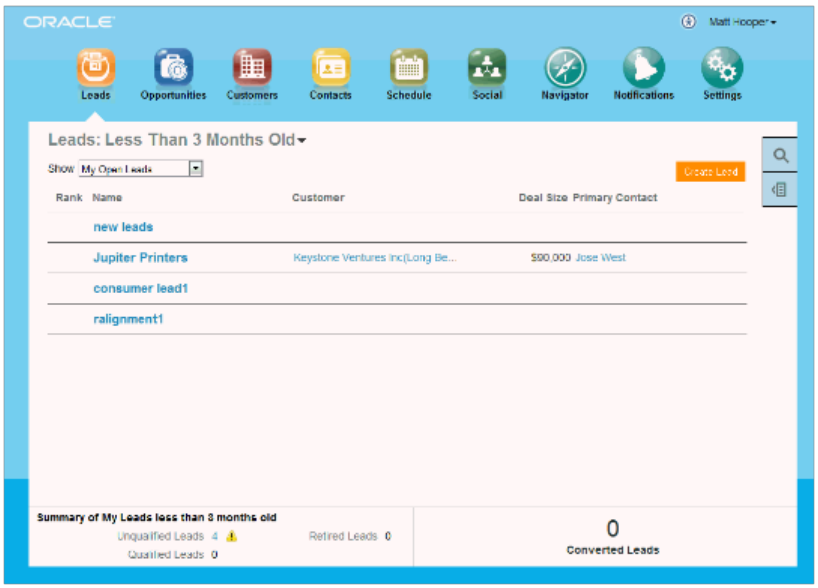
Note

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 Overview Table

You can extend the following items in the Leads overview table:

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



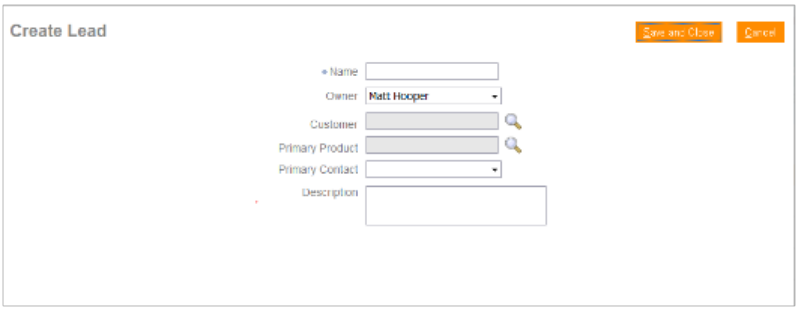
Note that you cannot make changes to these components on the page:

- 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

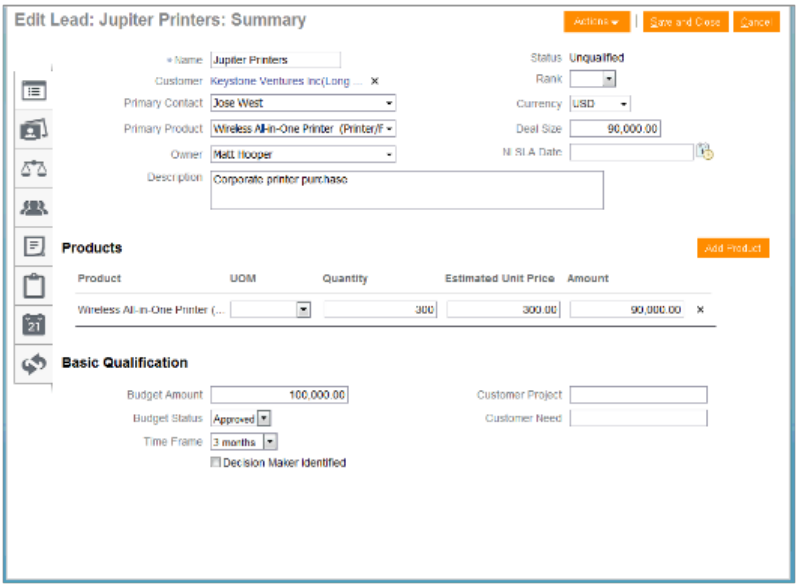
In Application Composer, add custom fields to the Sales Lead Resorces object. Then, navigate to the Pages node for the Sales Lead Version 2 object, and edit the Sales Team subtab in the Leads tabs component region.

- 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 Opportunities: Explained

Using Oracle Fusion CRM 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.

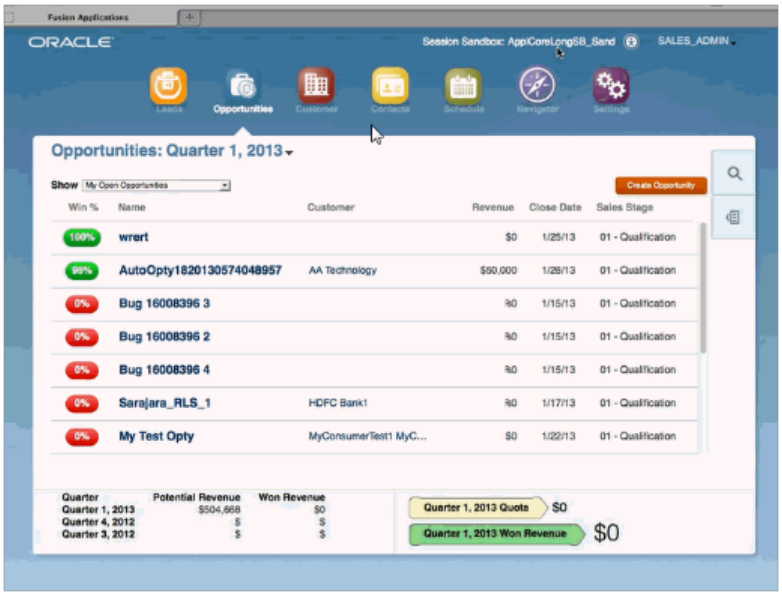
- Opportunity overview table
- Create Opportunity page
- Edit Opportunity page

Opportunity Overview Table

You can extend the following items in the Opportunity overview table:

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

This figure shows the Opportunities overview 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

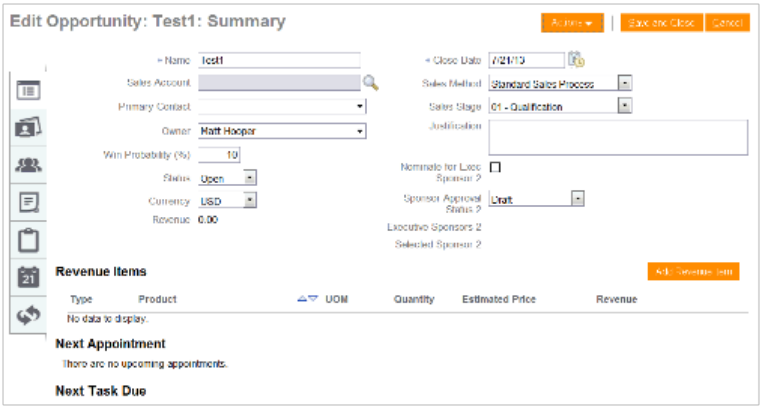
  In Application Composer, add custom fields to the Opportunity Team object. Then, navigate to the Pages node for the Opportunity object, and edit the Team subtab in the Opportunity Detail Tab region.
- Notes subtab

  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 Leads Using Application Composer: Worked Example

You can extend simplified pages using Oracle Fusion CRM 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:

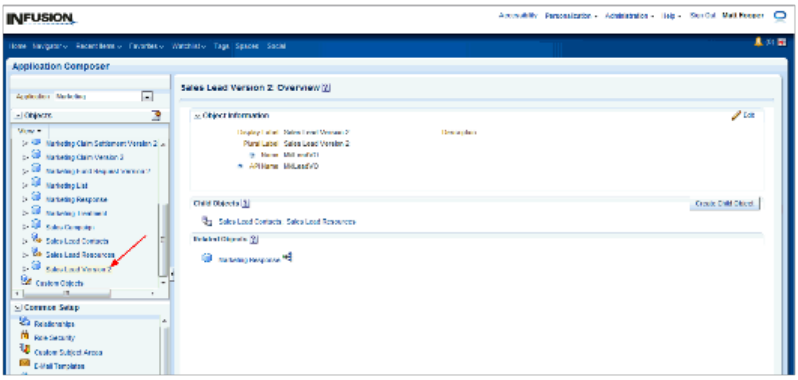
- Adding a check box to identify strategic leads
- Adding a Groovy script that checks if the size of the deal is greater than 1000 USD when a user saves a lead with this check box selected. If the deal is smaller than this amount, then users are prevented from saving the lead and receive an error message 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:

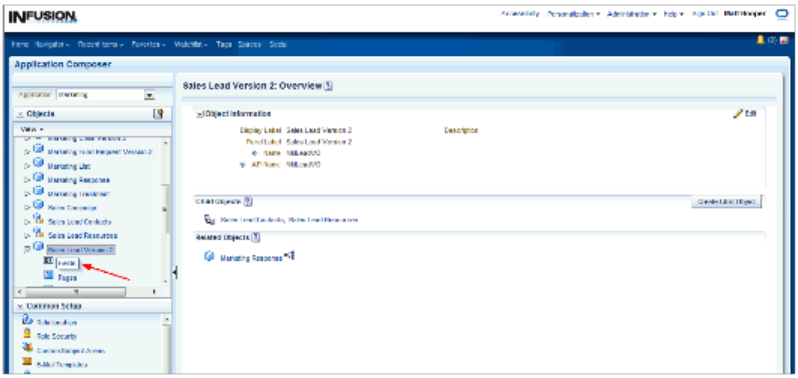
1. Navigate to Application Composer and select **Marketing** from the **Application** list. Select **Lead** from the **Objects** panel.

   This figure shows the Sales Lead Overview page in Application Composer.



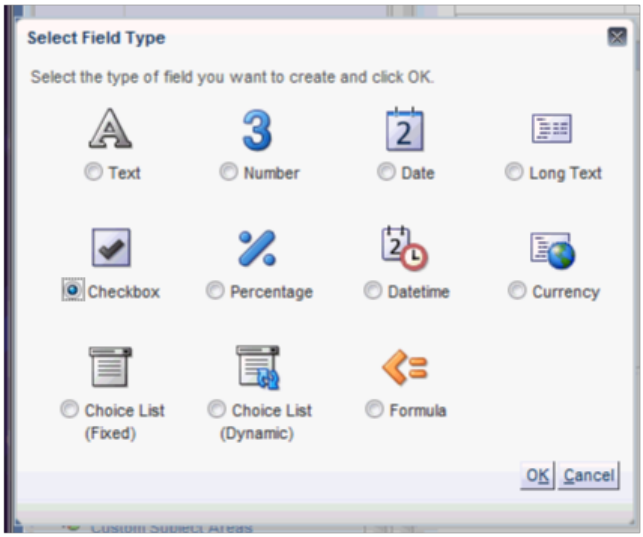
2. Under the Lead object, select the **Fields** link.

   This figure shows the Sales Lead Overview page with the **Fields** object selected for a Lead.



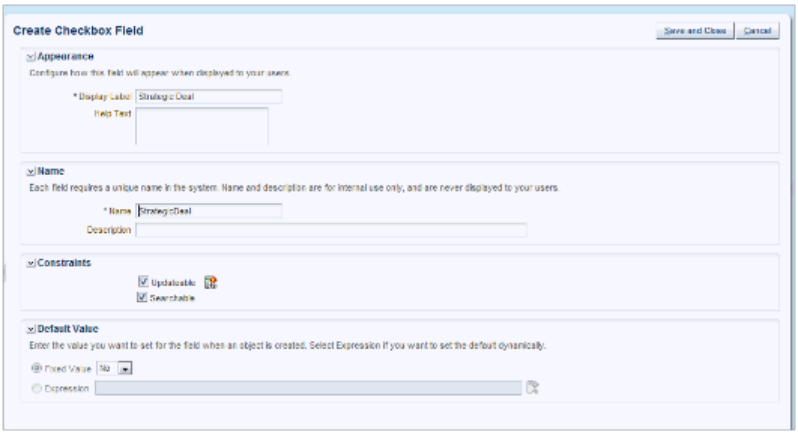
3. Click the **New** button and, in the Select Field Type window, select **Checkbox**.

   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:

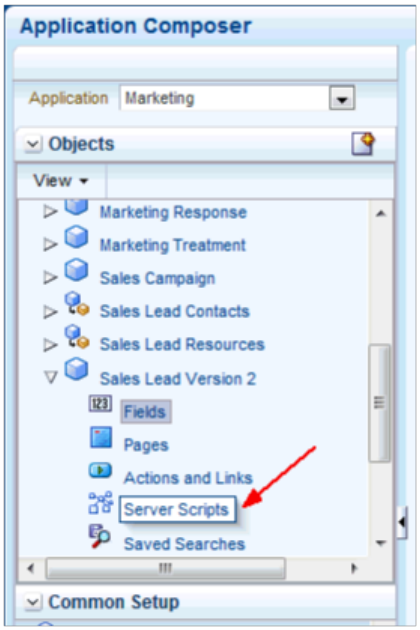
```
def retVal
if (StrategicDeal_c == 'Y')
{
 if (DealAmount >= 1000)
 {
 retVal = true
 }
 else
```

```
 {
 retVal = false
 }
}

if (StrategicDeal_c
 == null || StrategicDeal_c
 == 'N')
{
 retVal = true
}
 return(retVal)
```

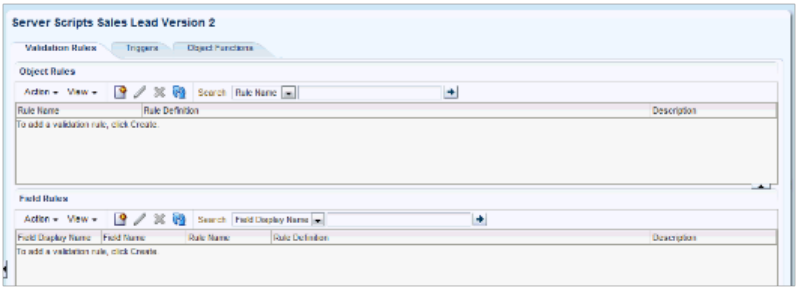
7. In the Objects panel, select **Server Scripts** under the **Sales Lead** object.

   This figure shows the Server Scripts object selected for a lead.



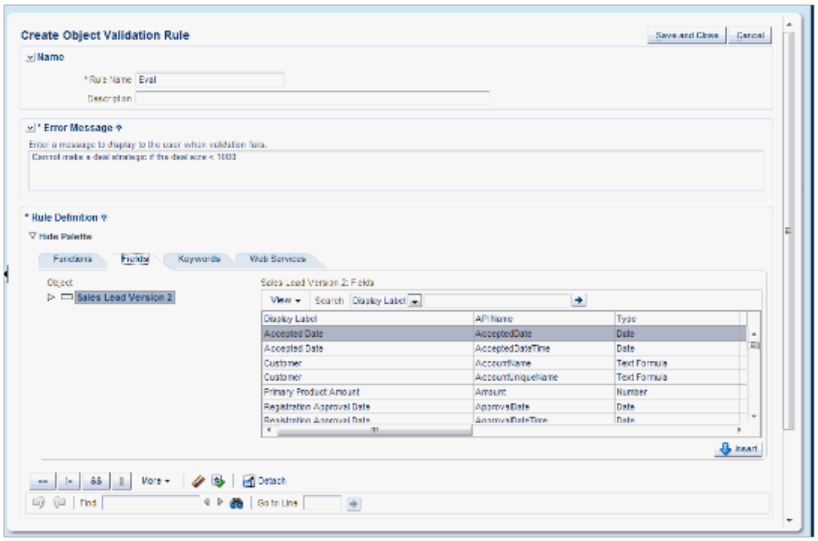
8. In the Server Scripts Lead page, select the Validation Rules tab and click the **New** button to create a new object rule.

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



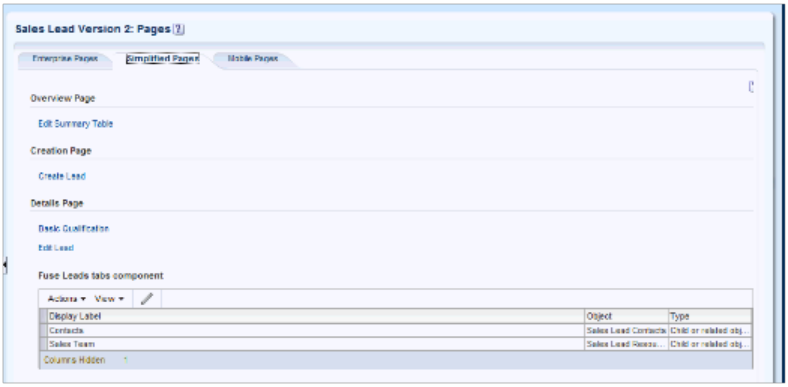
9. In the Create Object Validation Rule page, enter the rule name as "Eval" and enter a simple error message in the Error Message text region.

This figure shows the Create Object Validation Rule page.



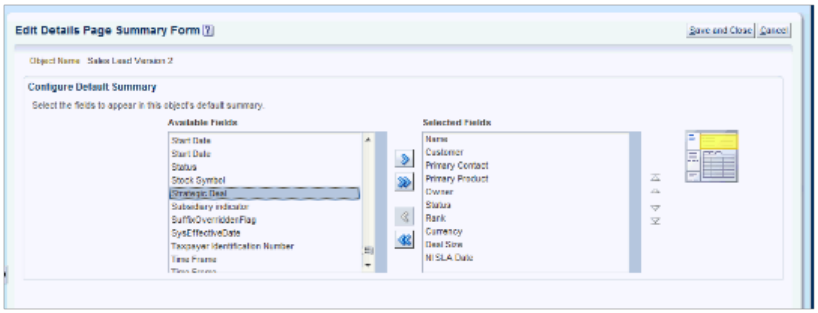
10. Cut and paste the script that you have written to validate the condition.
11. Now you need to add this field to the simplified pages. In Application Composer, select the **Pages** link under the Sales Lead object and then select the Simplified Pages tab.

    This figure shows the Simplified Pages tab.

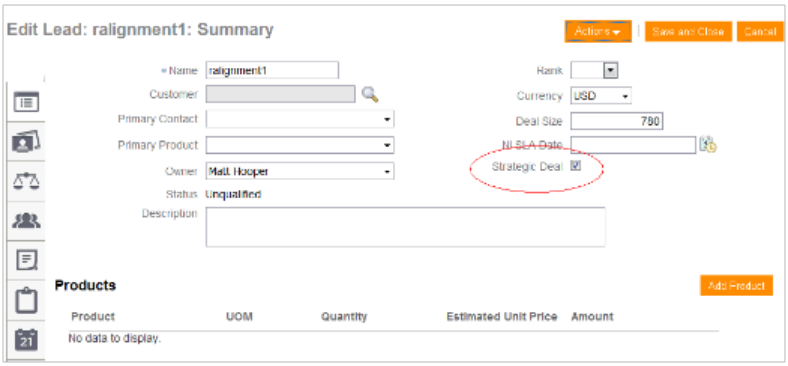


12. In the Details Page, click **Edit Lead**.
13. In the Edit Details Page Summary Form, select the **Strategic Deal** field in the Available Fields list and double click on this field to move it to the Selected Fields list.

    This figure shows the Edit Details Page Summary Form page.



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.



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 Using Application Composer: Worked Example

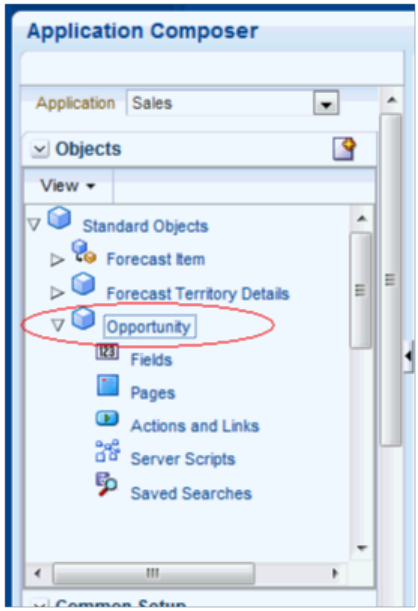
This example demonstrates how you can customize simplified pages for Opportunities using Oracle Fusion CRM Application Composer:

Extending Simplified Pages for Opportunities Using the Application Composer

1. Select the Sales application on the main Overview page.

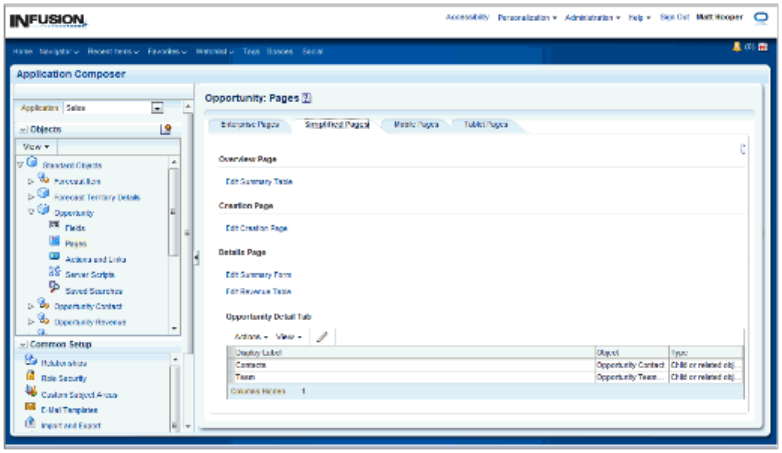
2. In the object tree, select a standard object that includes a set of simplified pages, such as Opportunity.

   This figure shows the Objects tree with the Opportunity 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.

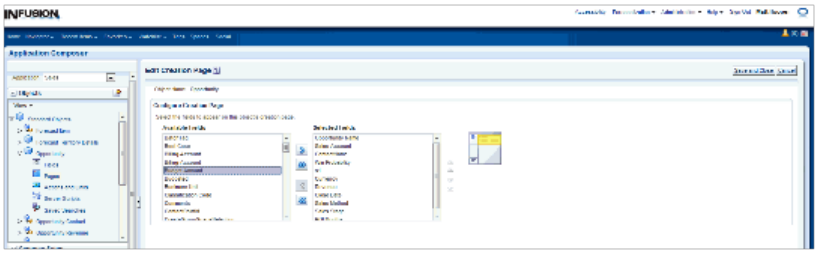
   This figure shows the simplified page for Opportunities



6. For example, if you want to add the field 'Budget Amount' to the Create Opportunity page, click the Edit Creation Page link in the Simplified Pages tab.

7. In the Available Fields list on the Configure Creation page, select Budget Amount.

   This figure shows the Configure Creation page.



8. 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.

9. Click Save and Close.

Extending Simplified Pages for Opportunities using Page Composer: Worked Example

In Oracle Fusion CRM, 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.

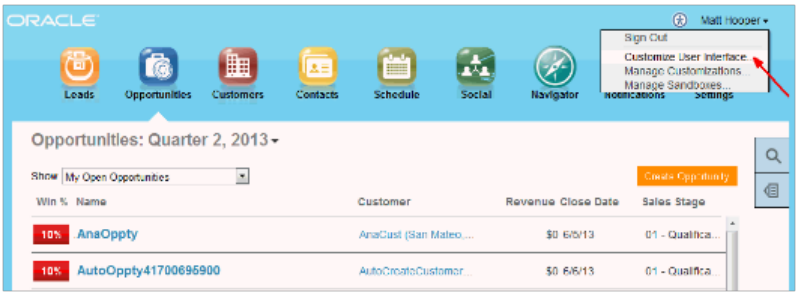
This example demonstrates how you can hide the **Include in Forecast** field for a sales representative in the Opportunities page and leave it available for the rest of the roles.

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 Fusion 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.



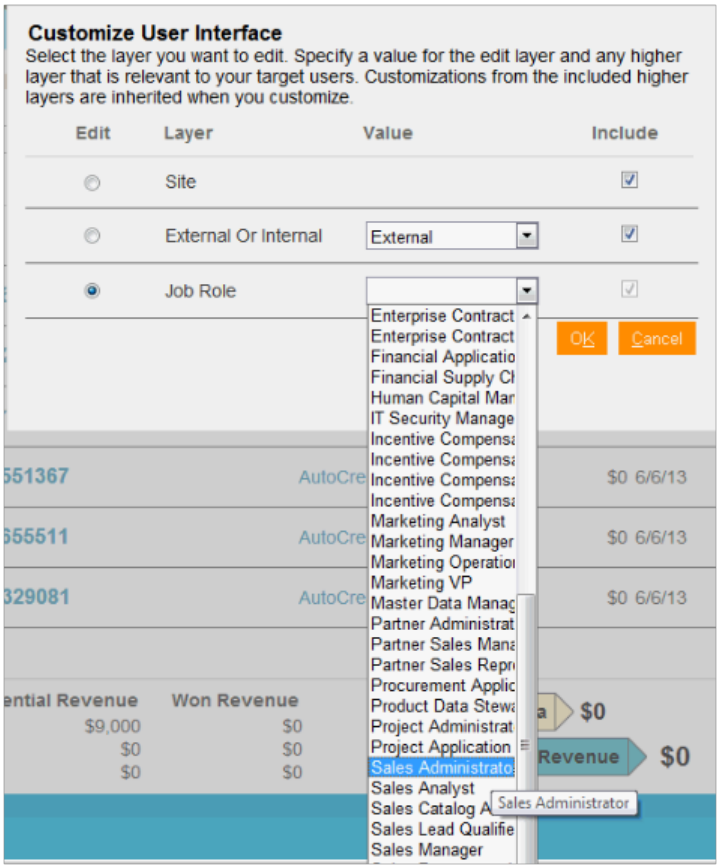
3. In the Customize Fusion Application Pages, select the layer that you want to edit.

   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.

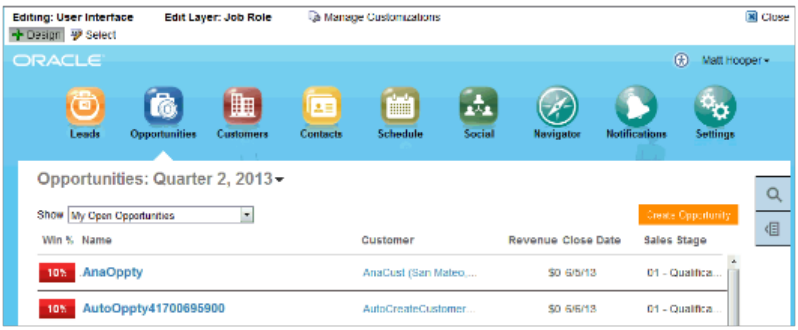


5. After you invoke the Page Composer in a particular layer (in this case, the Job Role layer) two buttons are displayed.

The two buttons are:

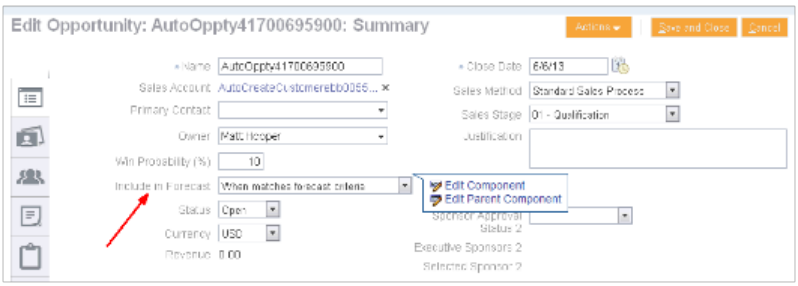
- Design: Represents design mode used to extend transactional pages.
- Select: Allows you to select components in a page to edit.

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



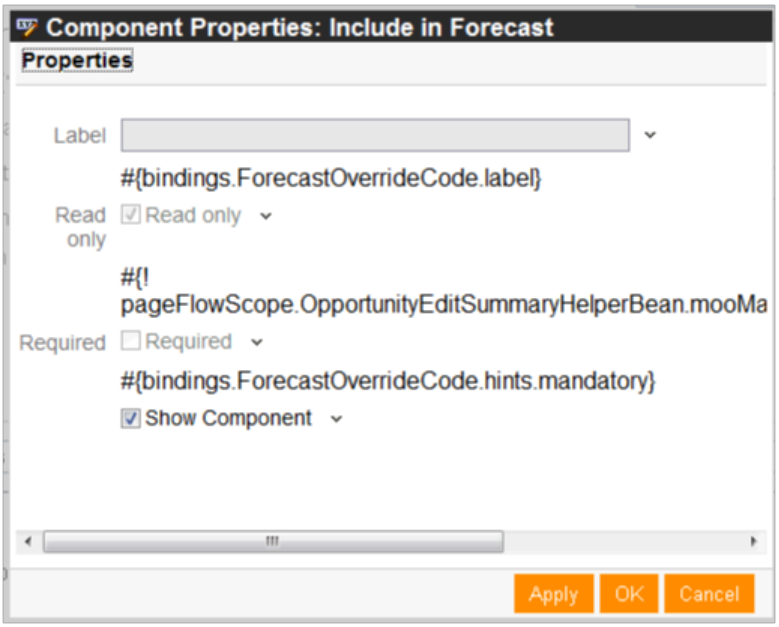
6. Drill into an opportunity and click Select.
7. In the Edit Opportunity page that appears, select the **Include in Forecast** field that you want to hide.

   This figure shows the Edit Opportunity page.



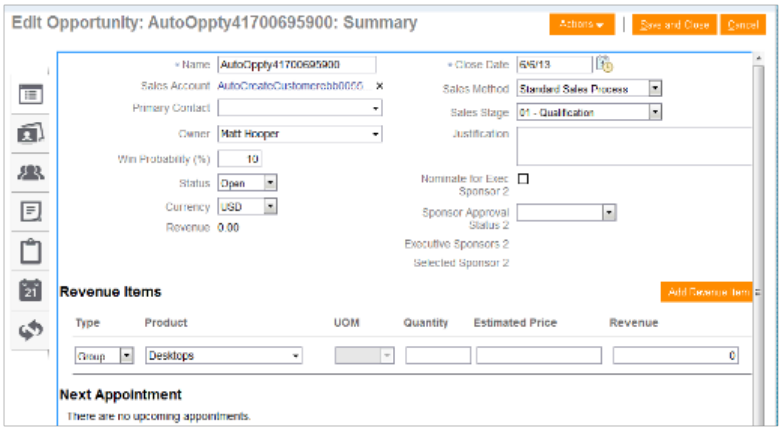
8. After you select the field, two options appear: **Edit Component** and **Edit Parent Component**.
9. Select **Edit Component**.
10. In the Component Properties window, notice that the **Show Component** check box is selected, which means that this field will be visible in the Opportunity page.

    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.



6

Application Composer: Using Groovy Scripts

Using Groovy Scripts: Overview

Read this chapter to learn about how and where you can use Groovy scripting in Oracle Fusion CRM Application Composer.

Important

To fully understand all the scripting features available to you in Application Composer, you should also review the Oracle Fusion CRM Application Composer Scripting Guide (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 the Oracle Fusion CRM Application Composer Scripting Guide (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 the Oracle Fusion CRM 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 Oracle Fusion CRM 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 Fusion CRM 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 Oracle Fusion CRM Application Composer Scripting Guide (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 Oracle Fusion CRM 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 Oracle Fusion CRM Application Composer Scripting Guide (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:

Field-Level Validation Rule Component	Value
Field Name	Priority
Rule Name	Validate_Priority_Range
Rule Body	`newValue == null || (1..5).contains(newValue as Integer)`
Error Message	The priority must be in the range from 1 to 5.

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 Oracle Fusion CRM Application Composer Scripting Guide (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:

Object-Level Validation Rule Component	Value
Rule Name	Validate_High_Priority_Ticket_Has_Owner

Rule Body	`// Rule depends on two fields, so must be written as object-level rule`
	`if (Priority_c <= 2 && AssignedTo_Id_c == null) {`
	`// Signal to highlight the AssignedTo field on the UI as being in error`
	`adf.error.addAttribute('AssignedTo_c')`
	`return false;`
	`}`
	`return true;`

Error Message	A Help Request of priority 1 or 2 must have a staff member assigned to it.

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 Oracle Fusion CRM Application Composer Scripting Guide (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 Oracle Fusion CRM Application Composer Scripting Guide (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.

Global Function Component	Value
Function Name	logStart
Return Type	void
Parameters	• Name: scriptName • Type: String
Function Definition	`// Log the name of the script` `println("[In: ${scriptName}]")`

This table describes how to set up a global function to log a diagnostic message.

Global Function Component	Value
Function Name	log
Return Type	void
Parameters	• Name: message • Type: String
Function Definition	`// Log the message, could add other info` `println(message)`

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 Oracle Fusion CRM Application Composer Scripting Guide (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 Oracle Fusion CRM Application Composer Scripting Guide (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:

Object Function Component	Value
Function Name	updateOpenTroubleTicketCount
Return Type	void
Parameters	None

Function Definition	`adf.util.logStart('updateOpenTroubleTicketCount')`
	`// Access the view object for TroubleTicket`
	` programmatic access`
	`def tickets = newView('TroubleTicket_c')`
	`// Use a global function to apply a query`
	` filter to Help Request`
	`// view object where Contact_Id = Id of`
	` current Contact and`
	`// Status is either 'Working' or 'Waiting'`
	`adf.util.applyFilter(tickets,[Status_c:`
	`['Working','Waiting'],`
	` Contact_Id__c:Id])`
	`// Update OpenTroubleTickets field value`
	`setAttribute('OpenTroubleTickets_c',tickets.getEstimatedRowCount(`

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.

Trigger Component	Value
Trigger Object	TroubleTicket
Trigger	After Insert In Database
Trigger Name	After_Insert_Set_Open_Trouble_Tickets

Trigger Definition	`adf.util.logStart('After_Insert_Set_Open_Trouble_Tickets')`
	`// Get the related contact for this Help Request`
	`def relatedContact = Contact_Obj_c`
	`// Update its OpenTroubleTickets field value`
	`relatedContact?.updateOpenTroubleTicketCount()`

This table describes how to set up a second trigger on the **TroubleTicket** object.

Trigger Component	Value
Trigger Object	TroubleTicket
Trigger	After Update In Database
Trigger Name	After_Update_Set_Open_Trouble_Tickets

Trigger Definition	`// Get the related contact for this Help Request`
	`def relatedContact = Contact_Obj_c`
	`// Update its OpenTroubleTickets field value`
	`relatedContact?.updateOpenTroubleTicketCount()`

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:

```
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 thatit 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.

Supported Classes and Methods for Use in Groovy Scripts: Explained

Groovy is a standard, dynamic scripting language for the Java platform for which the Oracle Fusion CRM 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()

Class Name	Allowed Methods	Package
ADFContext	getLocale() getSecurityContext() getUserRoles() isUserInRole()	oracle.adf.share
Array	Any constructor Any method	java.sql
Array	getArray() getElemType() getList()	oracle.jbo.domain
ArrayList	Any constructor Any method	java.util
Arrays	Any constructor Any method	java.util
AttributeDef	getAttributeKind() getIndex() getJavaType() getName() getPrecision() getProperty() getScale() getUIHelper() getUpdateableFlag() isMandatory() isQueriable()	oracle.jbo

AttributeHints	getControlType() getDisplayHeight() getDisplayHint() getDisplayWidth() getFormat() getFormattedAttribute() getFormatter() getFormatterClassName() getHint() getLocaleName() parseFormattedAttribute()	oracle.jbo
AttributeList	getAttribute() getAttributeIndexOf() getAttributeNames() setAttribute()	oracle.jbo
BaseLobDomain	closeCharacterStream() closeInputStream() closeOutputStream() getInputStream() getLength() getOutputStream() getcharacterStream()	oracle.jbo.domain
BigDecimal	Any constructor Any method	java.math
BigInteger	Any constructor Any method	java.math
BitSet	Any constructor Any method	java.util
Blob	Any constructor Any method	java.sql
BlobDomain	Any constructor getBinaryOutputStream() getBinaryStream() getBufferSize()	oracle.jbo.domain

Boolean	Any constructor Any method	java.lang
Byte	Any constructor Any method	java.lang
Calendar	Any constructor Any method	java.util
Char	Any constructor bigDecimalValue() bigIntegerValue() booleanValue() doubleValue() floatValue() getValue() intValue() longValue()	oracle.jbo.domain
Clob	Any constructor Any method	java.sql
ClobDomain	Any constructor toCharArray()	oracle.jbo.domain
Collection	Any constructor Any method	java.util
Collections	Any constructor Any method	java.util
Comparator	Any constructor Any method	java.util
Currency	Any constructor Any method	java.util
DBSequence	Any constructor getValue()	oracle.jbo.domain
Date	Any constructor Any method	java.util
Date	Any constructor Any method	java.sql

Date	Any constructor compareTo() dateValue() getValue() stringValue() timeValue() timestampValue()	oracle.jbo.domain
Dictionary	Any constructor Any method	java.util
Double	Any constructor Any method	java.lang
Enum	Any constructor Any method	java.lang
EnumMap	Any constructor Any method	java.util
EnumSet	Any constructor Any method	java.util
Enumeration	Any constructor Any method	java.util
EventListener	Any constructor Any method	java.util
EventListenerProxy	Any constructor Any method	java.util
EventObject	Any constructor Any method	java.util
Exception	Any constructor Any method	java.lang
ExprValueErrorHandler	addAttribute() clearAttributes() raise() raiseLater() warn()	oracle.jbo
Float	Any constructor Any method	java.lang

Formattable	Any constructor Any method	java.util
FormattableFlags	Any constructor Any method	java.util
Formatter	Any constructor Any method	java.util
GregorianCalendar	Any constructor Any method	java.util
HashMap	Any constructor Any method	java.util
HashSet	Any constructor Any method	java.util
Hashtable	Any constructor Any method	java.util
IdentityHashMap	Any constructor Any method	java.util
Integer	Any constructor Any method	java.lang
Iterator	Any constructor Any method	java.util
JboException	getDetails() getErrorCode() getErrorParameters() getLocalizedMessage() getMessage() getProductCode() getProperty()	oracle.jbo
JboWarning	Any constructor getDetails() getErrorCode() getErrorParameters() getLocalizedMessage() getMessage() getProductCode() getProperty()	oracle.jbo

Key	toStringFormat()	oracle.jbo
LinkedHashMap	Any constructor Any method	java.util
LinkedHashSet	Any constructor Any method	java.util
LinkedList	Any constructor Any method	java.util
List	Any constructor Any method	java.util
ListIterator	Any constructor Any method	java.util
ListResourceBundle	Any constructor Any method	java.util
Locale	Any constructor Any method	java.util
Long	Any constructor Any method	java.lang
Map	Any constructor Any method	java.util
Math	Any constructor Any method	java.lang
MathContext	Any constructor Any method	java.math
NClob	Any constructor Any method	java.sql
NameValuePairs	Any constructor getAttribute() getAttributeIndexOf() getAttributeNames() setAttribute()	oracle.jbo
NativeTypeDomainInterface	getNativeObject()	oracle.jbo.domain

Number	Any constructor bigDecimalValue() bigIntegerValue() booleanValue() byteValue() doubleValue() floatValue() getValue() intValue() longValue() shortValue()	oracle.jbo.domain
Observable	Any constructor Any method	java.util
Observer	Any constructor Any method	java.util
PriorityQueue	Any constructor Any method	java.util
Properties	Any constructor Any method	java.util
PropertyPermission	Any constructor Any method	java.util
PropertyResourceBundle	Any constructor Any method	java.util
Queue	Any constructor Any method	java.util
Random	Any constructor Any method	java.util
RandomAccess	Any constructor Any method	java.util
Ref	Any constructor Any method	java.sql
ResourceBundle	Any constructor Any method	java.util

Row	getAttribute() getAttributeHints() getKey() getLookupDescription() getOriginalAttributeValue() getPrimaryPostState() isAttributeChanged() isAttributeUpdateable() remove() revertRow() revertRowAndContainees() setAttribute() validate()	oracle.jbo
RowId	Any constructor Any method	java.sql
RowIterator	createAndInitRow() createRow() findByKey() findRowsMatchingCriteria() first() getAllRowsInRange() getCurrentRow() getEstimatedRowCount() hasNext() hasPrevious() insertRow() last() next() previous() reset()	oracle.jbo

RowSet	avg() count() createAndInitRow() createRow() executeQuery() findByKey() findRowsMatchingCriteria() first() getAllRowsInRange() getCurrentRow() getEstimatedRowCount() hasNext() hasPrevious() insertRow() last() max() min() next() previous() reset() sum()	oracle.jbo
Scanner	Any constructor Any method	java.util
SecurityContext	getUserName() getUserProfile()	oracle.adf.share.security
Session	getLocale() getLocaleContext() getUserData()	oracle.jbo
Set	Any constructor Any method	java.util
Short	Any constructor Any method	java.lang
Short	Any constructor Any method	java.lang

SimpleTimeZone	Any constructor Any method	java.util
SortedMap	Any constructor Any method	java.util
SortedSet	Any constructor Any method	java.util
Stack	Any constructor Any method	java.util
StackTraceElement	Any constructor Any method	java.lang
StrictMath	Any constructor Any method	java.lang
String	Any constructor Any method	java.lang
StringBuffer	Any constructor Any method	java.lang
StringBuilder	Any constructor Any method	java.lang
StringTokenizer	Any constructor Any method	java.util
Struct	Any constructor Any method	java.sql
Struct	getAttribute() setAttribute()	oracle.jbo.domain
StructureDef	findAttributeDef() getAttributeIndexOf()	oracle.jbo
Time	Any constructor Any method	java.sql
TimeZone	Any constructor Any method	java.util
Timer	Any constructor Any method	java.util
TimerTask	Any constructor Any method	java.util
Timestamp	Any constructor Any method	java.sql

Timestamp	Any constructor compareTo() dateValue() getValue() stringValue() timeValue() timestampValue()	oracle.jbo.domain
TreeMap	Any constructor Any method	java.util
TreeSet	Any constructor Any method	java.util
UUID	Any constructor Any method	java.util

UserProfile	getBusinessCity()	oracle.adf.share.security.identitymanagment
	getBusinessCountry()	
	getBusinessEmail()	
	getBusinessFax()	
	getBusinessMobile()	
	getBusinessPOBox()	
	getBusinessPager()	
	getBusinessPhone()	
	getBusinessPostalAddr()	
	getBusinessPostalCode()	
	getBusinessState()	
	getBusinessStreet()	
	getDateofBirth()	
	getDateofHire()	
	getDefaultGroup()	
	getDepartment()	
	getDepartmentNumber()	
	getDescription()	
	getDisplayName()	
	getEmployeeNumber()	
	getEmployeeType()	
	getFirstName()	
	getGUID()	
	getGivenName()	
	getHomeAddress()	
	getHomePhone()	
	getInitials()	
	getJpegPhoto()	
	getLastName()	
	getMaidenName()	
	getManager()	
	getMiddleName()	
	getName()	
	getNameSuffix()	
	getOrganization()	
	getOrganizationalUnit()	
	getPreferredLanguage()	

ValidationException	getDetails() getErrorCode() getErrorParameters() getLocalizedMessage() getMessage() getProductCode() getProperty()	oracle.jbo
Vector	Any constructor Any method	java.util
ViewCriteria	createAndInitRow() createRow() createViewCriteriaRow() findByKey() findRowsMatchingCriteria() first() getAllRowsInRange() getCurrentRow() getEstimatedRowCount() hasNext() hasPrevious() insertRow() last() next() previous() reset()	oracle.jbo
ViewCriteriaItem	getValue() makeCompound() setOperator() setUpperColumns() setValue()	oracle.jbo

ViewCriteriaItemCompound	ensureItem() getValue() makeCompound() setOperator() setUpperColumns() setValue()	oracle.jbo
ViewCriteriaRow	ensureCriteriaItem() getConjunction() isUpperColumns() setConjunction() setUpperColumns()	oracle.jbo

ViewObject	appendViewCriteria() avg() count() createAndInitRow() createRow() createViewCriteria() executeQuery() findByKey() findRowsMatchingCriteria() first() getAllRowsInRange() getCurrentRow() getEstimatedRowCount() getMaxFetchSize() hasNext() hasPrevious() insertRow() last() max() min() next() previous() reset() setMaxFetchSize() sum()	oracle.jbo
WeakHashMap	Any constructor Any method	java.util

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 Oracle Fusion CRM 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 Oracle Fusion CRM Application Composer Scripting Guide (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.

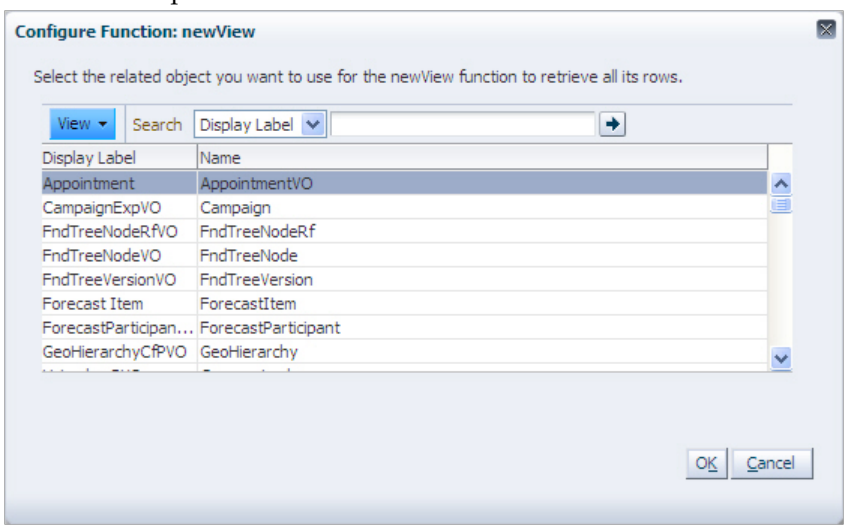
This topic:

- 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 Oracle Fusion Common CRM 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 Fusion CRM 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 Oracle Fusion Common CRM 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.

1. Go to https://fusionappsoer.oracle.com.
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.

Standard View Object	Description	Typical Attributes
Address	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 Organization, Person, or Group object.	Refer to the Trading Community Address object in Application Composer, and review the descriptions provided for all attributes.
CodeAssignment	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 Organization or Person profile objects.	Refer to the Trading Community Classification Code Assignment in OER ADF Service Data Object.
CommonLookup	Access application common lookups in scripting.	LookupType, LookupCode, Tag, EnabledFlag, StartDateActive, EndDateActive, Meaning, Description
Contact	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 Organization, Person, or Group objects.	Refer to the Trading Community Customer Contact Profile object in Application Composer, and review the descriptions provided for all attributes.
FndTreeVersion	Use this object in scripting to access tree versions. The customer hierarchy and party hierarchy are modeled as trees.	TreeStructureCode, TreeCode, TreeVersionID, Status, EffectiveStartDate, EffectiveEndDate, TreeVersionName
FndTreeNode	Use this object to determine the parent/child relationships for a given hierarchy. The hierarchy for a given version is stored in this object.	TreeStructureCode, TreeCode, TreeVersionID, TreeNodeID, ParentTreeNodeID, Depth, ChildCount, ParentPk1Value

FndTreeNodeRf	Use this object in scripting to easily access the flattened version of the given hierarchy version.	TreeStructureCode, TreeCode, TreeVersionID, TreeNodeID, Pk1Value, AncestorPk1Value, Distance, IsLeaf
Location	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.	Refer to the Trading Community Location SDO in OER ADF Service Data Object.
OrganizationParty	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 Organization Profile object.	Refer to the Trading Community Organization Details in OER ADF Service Data Object.
OrganizationProfile	Access this Organization Profile extensible object as a child of an OrganizationParty row or directly get the profile if you have a PartyID.	Refer to the Trading Community Organization Profile object in Application Composer, and review the descriptions provided for all attributes.
OriginalSystemReference	Use this object to get the ID for given TCA object based on the source system and source system reference information.	Refer to the Trading Community Original System Reference in OER ADF Service Data Object.
PersonParty	Use this object to get the Person Party and all of its children when you have the person PartyID in your script, and you do not have a view link from the current object to Organization Profile object.	Refer to the Trading Community Person Details in OER ADF Service Data Object.
PersonProfile	Access this Person Profile extensible object as a child of a PersonParty row or directly get the profile if you have a PartyID.	Refer to the Trading Community Person Profile object in Application Composer, and review the descriptions provided for all attributes.
Relationship	Use this object in scripting if you have a RelationshipID on the current object and that object does not have a view link to this object. Access this Relationship extensible object as a child of the Organization, Person, or Group object.	Refer to the Trading Community Relationship object in Application Composer, and review the descriptions provided for all attributes.

Resource	Use this Resource extensible object in scripting to get the resource object details if you have a user or resource PartyID, and the current object ID does not expose a view link to this object.	Refer to the Trading Community Resource Profile in OER ADF Service Data Object.

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 Oracle Fusion CRM Application Composer. Some Oracle Fusion CRM applications provide global functions ready for your use, or you can define new global functions.

For more information on global functions, see Oracle Fusion CRM 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.

Global Function	Description
adf.util.getUserPartyId()	Returns the logged-in user's Party_ID.
adf.util.getUserPartnerCompanyId()	Returns the partner company's party_ID for the logged-in user, if the user is a partner user.
adf.util.getUserRootResourceOrgId()	Returns the organization_ID for the logged-in user's organization hierarchy root resource organization.

Calling Web Services from Groovy Scripts : Explained

You can call Web services from your Groovy scripts in Oracle Fusion CRM 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)

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:

```
def newEmp = [:]
```

Then, you can add properties to the map using the explicit put() method like this:

```
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:

```
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:

```
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:

```
def dependent1 = [Name : "Dave",
 BirthYear : 1996]
def dependent2 = [Name : "Jenna",
 BirthYear : 1999]
def listOfDependents = []
listOfDependents.add(dependent1)
listOfDependents.add(dependent2)
```

To save a few steps, the last three lines in the preceding example can be done in a single line by constructing a new list with the two desired elements in one line like this:

```
def listOfDependents = [dependent1, dependent2]
```

You can also create the list of maps in a single operation using a combination of list constructor syntax and map constructor syntax:

```
def listOfDependents = [[Name : "Dave",
 BirthYear : 1996],
 [Name : "Jenna",
 BirthYear : 1999]]
```

If the employee object in the previous codes examples has a property named Dependents that is a list of objects representing dependent children, you can assign the property using the same syntax as shown above (using a list of maps as the value assigned):

```
newEmp.Dependents = [[Name : "Dave",
 BirthYear : 1996],
 [Name : "Jenna",
 BirthYear : 1999]]
```

Lastly, note that you can also construct a new employee with nested dependents all in a single statement by further nesting the constructor syntax:

```
def newEmp = [Empno : 1234,
 Ename : "Sean",
 Sal : 9876,
 Hiredate : date(2013,8,11),
 Dependents : [
 [Name : "Dave",
 BirthYear : 1996],
```

```
 [Name : "Jenna",
 BirthYear : 1999]]
 ]
```

For more information on maps and lists, see the section called Working with Maps in 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 Oracle Fusion CRM 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.

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.

Web Service Calls in Groovy Scripts: Examples

You can call Web services from your Groovy scripts in Oracle Fusion CRM 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 Oracle Fusion CRM 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.

```
// 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.

```
// 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.

```
// 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 Oracle Fusion CRM 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:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws

What is the URL of the Web Services Description Language (WSDL) file that you will use?	http://external-hosted:7101/MathsWS-Model-context-root/NoSecurity?wsdl **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.

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

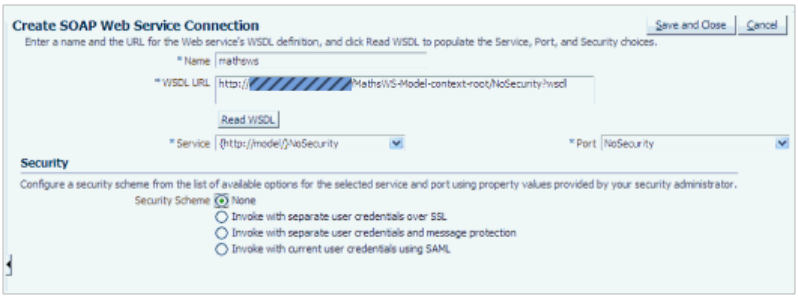
When you create a Web service connection, you specify a name for the Web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of Web services in the Expression Builder in Application Composer.

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.
4. Enter `http://external-hosted:7101/MathsWS-Model-context-root/NoSecurity?wsdl` in the **WSDL URL** field, and click **Read WSDL**.

   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**.

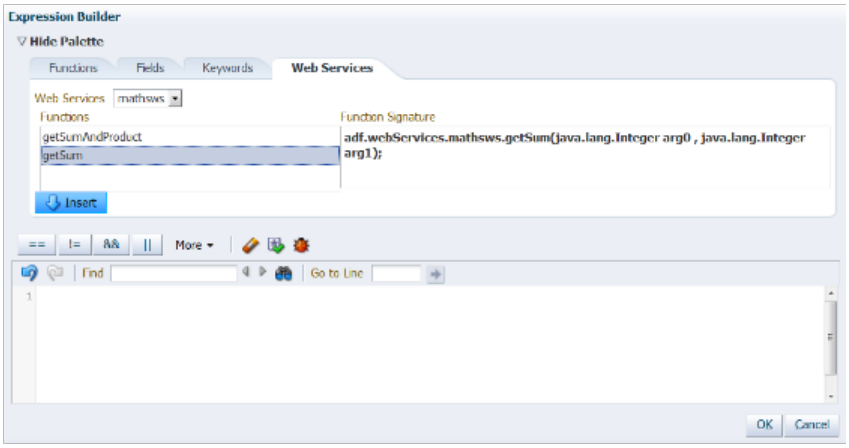
   The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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 Oracle Fusion CRM Application Composer. The Web service is secured with message protection. For this example, the Web service is used to calculate a custom field's default value.

The following table summarizes key decisions for this scenario:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use?	http://external-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl This WSDL file specifies the desired message protection security scheme. **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Which credential key will you use?	mylogin
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.
What will the server encryption alias name be?	serverenckey

To call a Web service from a Groovy script that is secured with message protection, complete the following tasks:

1. Create the Web service connection.
2. Add the Web service call to the Groovy script, and check whether the call succeeds.
3. Contact the administrator to resolve runtime exceptions
4. Re-create the Web service connection.
5. Verify that the Web service call succeeds.

Prerequisites

Verify that you have completed the following prerequisite steps:

1. Get details of the WSDL URL and the user credentials to use from the Web service provider.
2. Get the server encryption certificate and the Certificate Authority (issuer) certificate from the Web service provider.
3. Create a custom field for an object that has a calculated default value.
4. Prepare the Groovy script for the expression used to calculate the field's default value. The Groovy code must prepare the argument values, which in this example are two values that are summed.

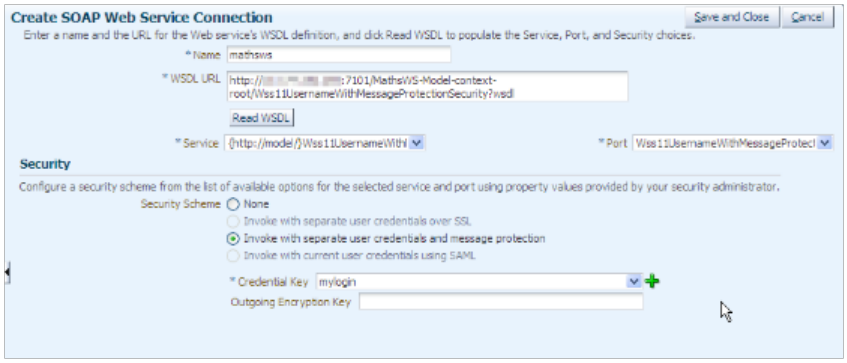
Creating the Web Service Connection

When you create a Web service connection, you specify a name for the Web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of Web services in the Expression Builder in Application Composer.

1. In Application Composer, select **Web Services** in the Common Setup pane.
2. On the Web Services page, click the New icon.
3. On the Create SOAP Web Service Connection page, enter `mathsws` in the **Name** field.

   The name must not include periods.
4. Enter `http://external-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl`l in the **WSDL URL** field, and click **Read WSDL**.

   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. Click **Save and Close**.

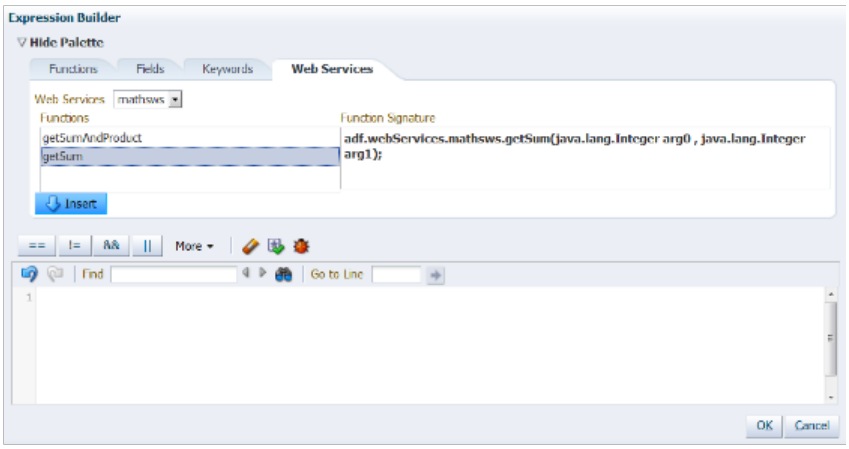
   The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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.
5. Enter `http://external-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl`l in the **WSDL URL** field, and click **Read WSDL**.
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. On the Create SOAP Web Service Connection page, enter `serverenckey` in the **Outgoing Encryption Key** field.
9. 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 Oracle Fusion CRM 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:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use?	https://external-hosted:7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecurity?wsdl This WSDL file specifies the desired SSL security scheme. **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Which credential key will you use?	mylogin
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.

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.

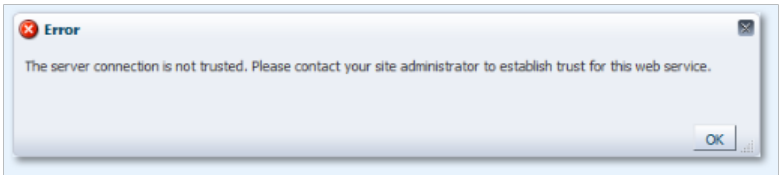
Creating the Web Service Connection

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.
4. Enter `https://external-hosted:7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecurity?wsdl` in the **WSDL URL** field, and click **Read WSDL**.

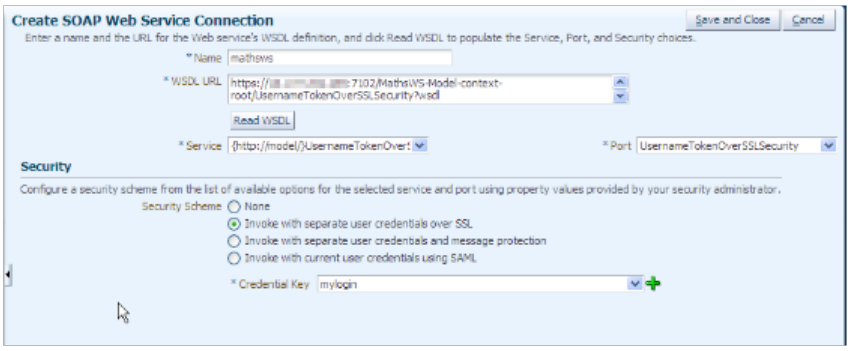
   The following figure shows the error that is displayed after you click **Read WSDL**.



   You must create a service request for your administrator to resolve the issue.
5. Create a service request for your administrator:
   a. Retrieve the server's Certificate Authority (CA) SSL certificate from the Web service provider.
   b. Attach the SSL certificate to the service request, and include the WSDL location, and error details
   c. Submit the service request.

   The administrator will add the SSL certificate into the Oracle Fusion CRM trust store.
6. Wait until your administrator informs you that the SSL certificate has been imported, and close the service request.
7. Repeat steps 1 through 4.

   The following figure shows what happens after you click **Read WSDL**. The **Service** and **Port** fields are filled according to values in the WSDL file. Under **Security Scheme**, the **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. Click **Save and Close**.

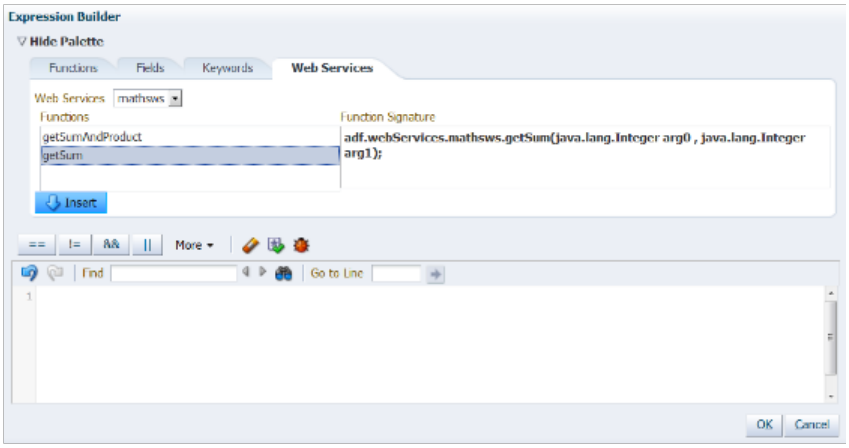
    The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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 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 Oracle Fusion CRM 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:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use?	https://internal-hosted:7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecurity?wsdl This WSDL file specifies the desired SSL authentication scheme. **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Which credential key will you use?	mylogin
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.

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.

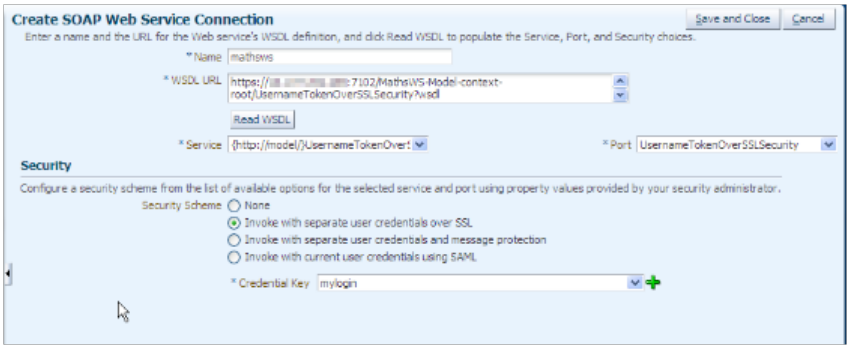
Creating the Web Service Connection

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.
4. Enter `https://internal-hosted:7102/MathsWS-Model-context-root/UsernameTokenOverSSLSecurity?wsdl` in the **WSDL URL** field, and click **Read WSDL**.

   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. Click **Save and Close**.

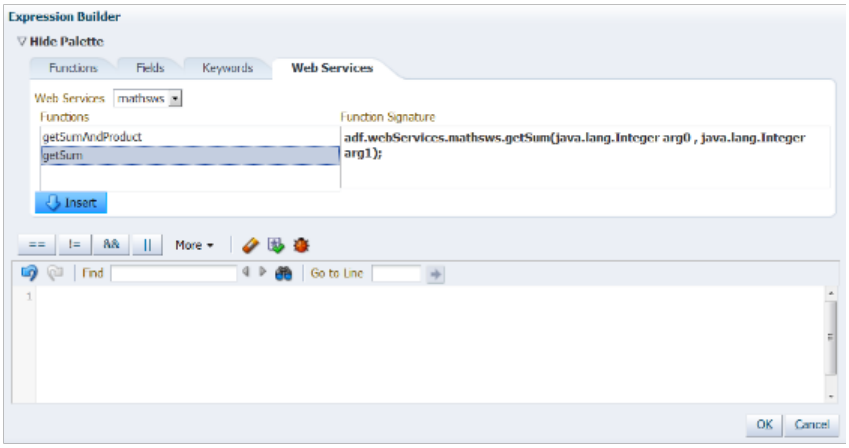
   The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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 Oracle Fusion CRM Application Composer. The Web service is secured with message protection. For this example, the Web service is used to calculate a custom field's default value.

The following table summarizes key decisions for this scenario:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use?	http://internal-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl This WSDL file specifies the desired message protection security scheme. **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Which credential key will you use?	mylogin
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.

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.

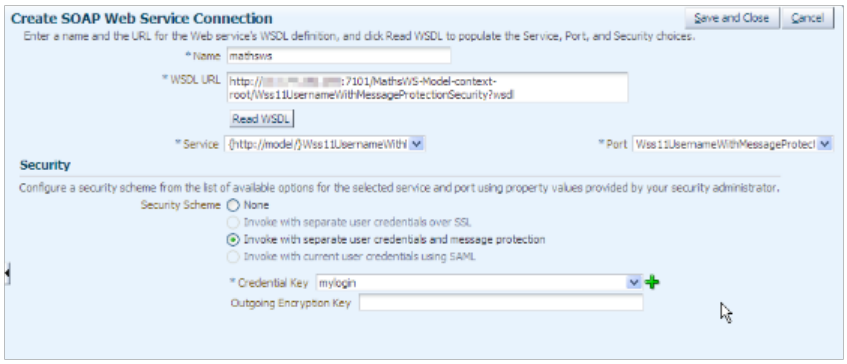
Creating the Web Service Connection

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.
4. Enter `http://internal-hosted:7101/MathsWS-Model-context-root/Wss11UsernameWithMessageProtectionSecurity?wsdl`l in the **WSDL URL** field, and click **Read WSDL**.

   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. Click **Save and Close**.

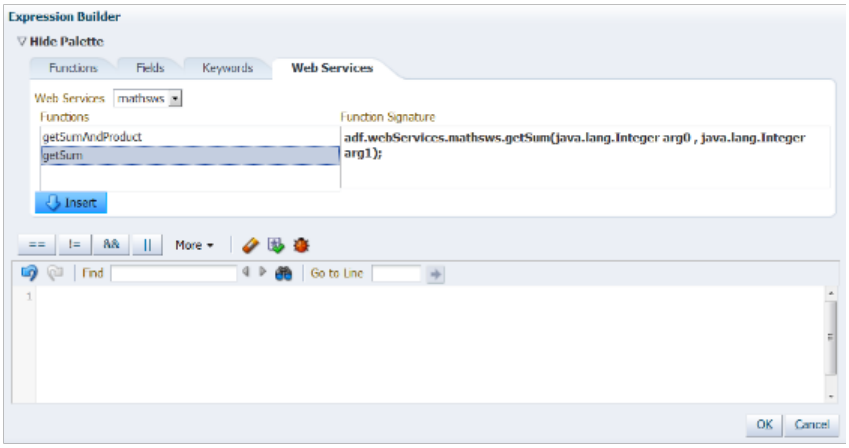
   The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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 Oracle Fusion CRM 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:

Decisions to Consider	In This Example
What name will you use for the Web service connection?	mathsws
What is the URL of the Web Services Description Language (WSDL) file that you will use?	https://internal-hosted:7102/MathsWS-Model-context-root/SamlOrUsernameTokenWithMessageProtection?wsdl **Note** The URL shown here is an arbitrary example. You must obtain the real WSDL URL from the service provider,
Where will the Web service be called from?	From a Groovy script expression used to calculate a custom field's default value.
Which Web service method will be called from the Groovy script?	getSum This method returns the sum of two integer argument values.

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

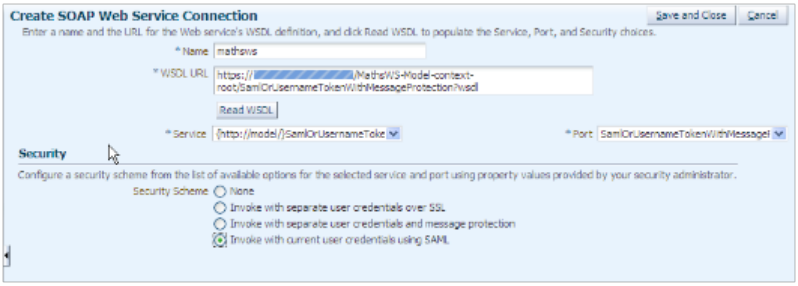
When you create a Web service connection, you specify a name for the Web service, the URL of the WSDL file, and the security scheme settings. The name is simply an identifier that is used in the list of Web services in the Expression Builder in Application Composer.

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.
4. Enter `https://internal-hosted:7102/MathsWS-Model-context-root/SamlOrUsernameTokenWithMessageProtection?wsdl`l in the **WSDL URL** field, and click **Read WSDL**.

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. Click **Save and Close**.

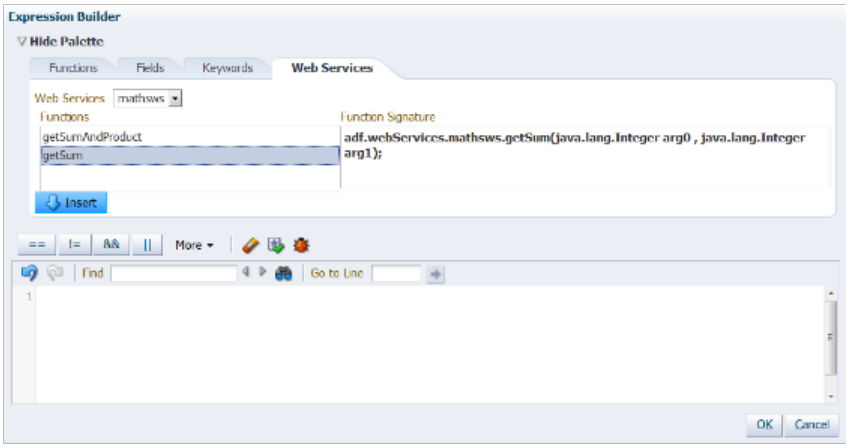
   The Web service connection is created and the name and WSDL URL are listed on the Web Services page.

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.

7

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:

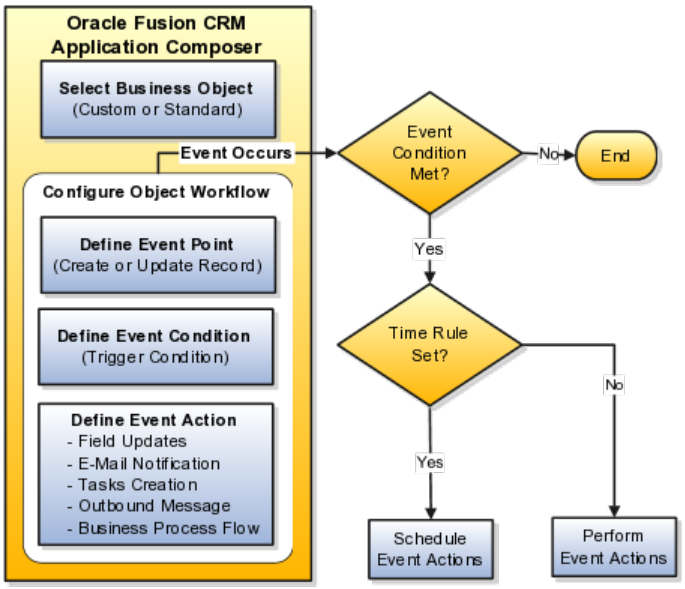
- The concepts and terminology used in object workflows.
- How to set trigger conditions for object workflows using groovy scripts.
- Configuring event actions such as tasks, e-mail notifications, field updates, business process flows, and outbound messages using object workflows.
- Creating an approval flow using business process composer and then triggering the approval flow through object workflows.

Object Workflows: Explained

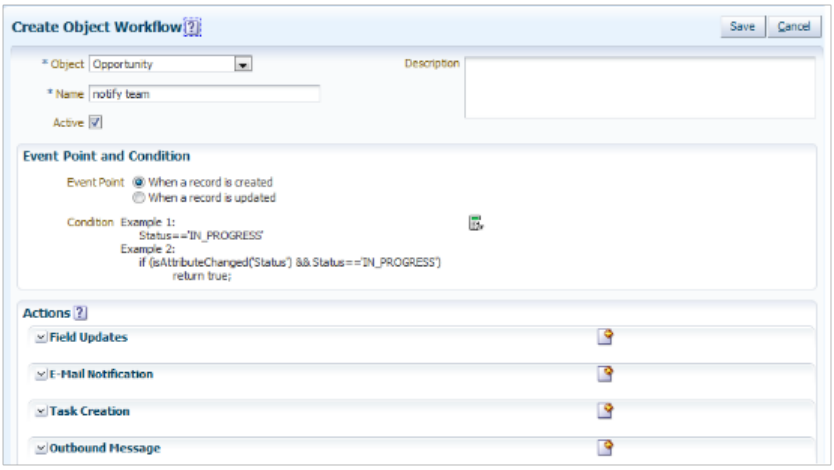
The object workflows represent orchestrated business processes that are triggered based on the conditions that you set while defining the workflows. You must have administrative privileges to configure object workflows.

You configure object workflows using the main page of the Application Composer.

This figure illustrates how you configure and trigger object workflow.



This figure illustrates the create object workflow page.



Some of the aspects or common terminology used for object workflows is as follows:

- Business Object: A business object or an object can be a standard object delivered with the product or a custom object that you define based on your business need.
- 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: An event condition is a trigger for object workflows. It 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. See: Expression Editor in this topic.
- 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.

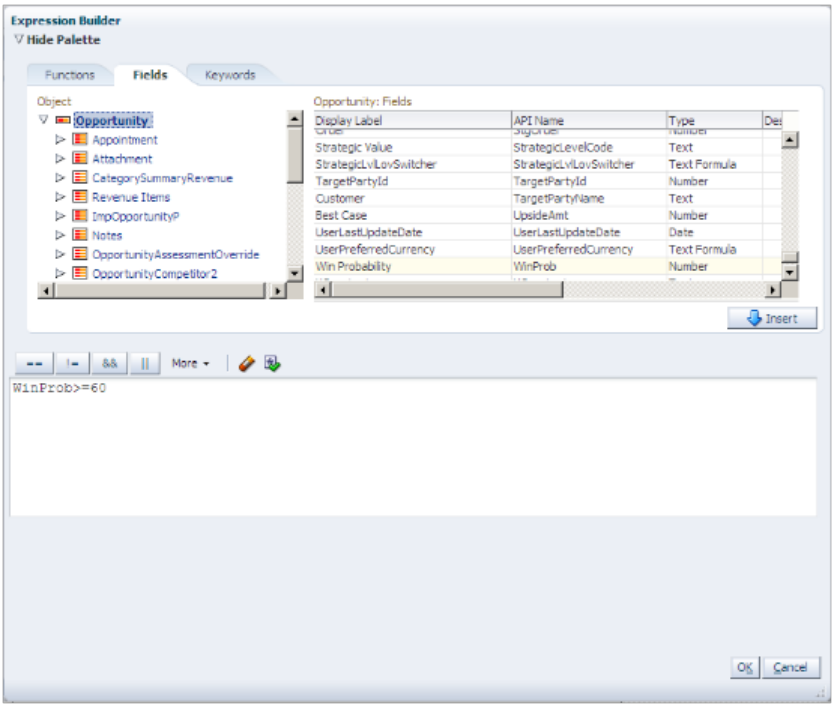
Application Composer

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 Editor

An expression editor supports building logical and math operations, including field lookups that you can optionally use to define trigger condition. Fields in the expression editor are populated based on the object for which you are defining a workflow. The expression editor 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 editor using which you can define groovy conditions for object workflows.



Use the expression editor 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:

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

Example 3:

```
WinProb>10 || WinProb<50
```

In addition, keep these points in mind when using the expression editor to build raise conditions:

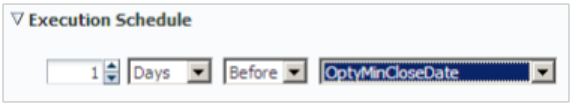
- Ensure that you return a valid boolean as part of your raise condition. Returning a non-boolean value could lead to run-time errors.
- Use `return true` or `return false` to explicitly return the boolean value and code indentation when the evaluation logic is complex, to minimize risk of run-time errors.
- 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 Fusion CRM Application Composer Scripting Guide (ID 1354807.1) on My Oracle Support at https://support.oracle.com.

Execution Schedule

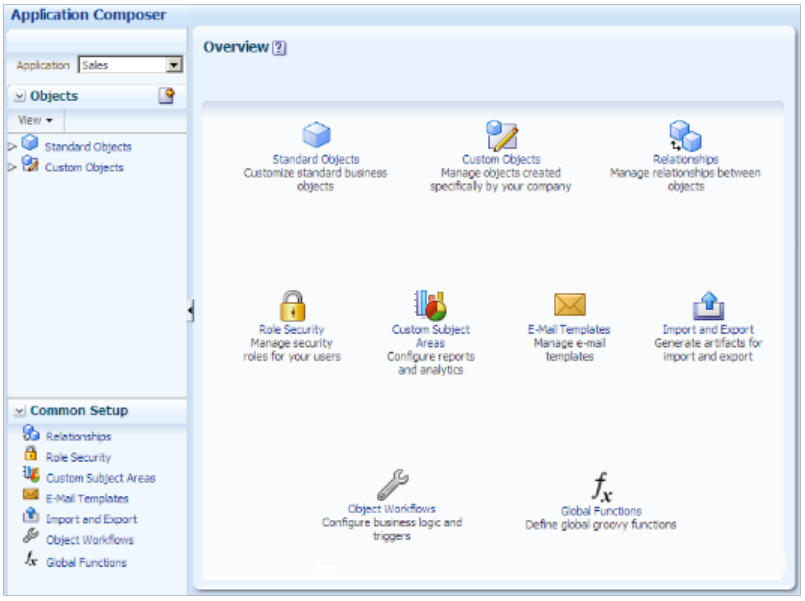
While defining an event action, you can optionally set an execution schedule that governs when an event action should happen. Else, the event actions are run immediately. When multiple event actions are configured, the Field Updates happen first before any other event action is triggered. For event actions 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.

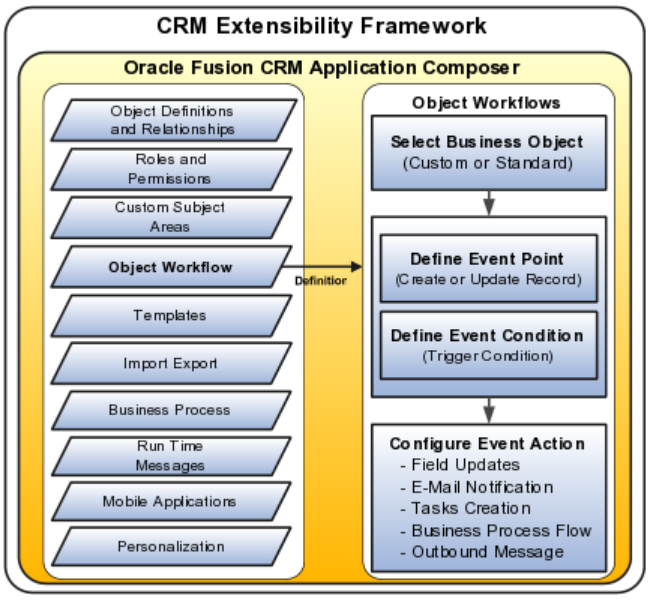


Object Workflows and Application Composer: How They Work Together

Oracle Fusion CRM Application Composer is a browser-based configuration tool. It is part of the CRM 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.

For more information on customizing Oracle Fusion CRM Applications using Application Composer, see Oracle Fusion Applications CRM Extensibility Guide on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.

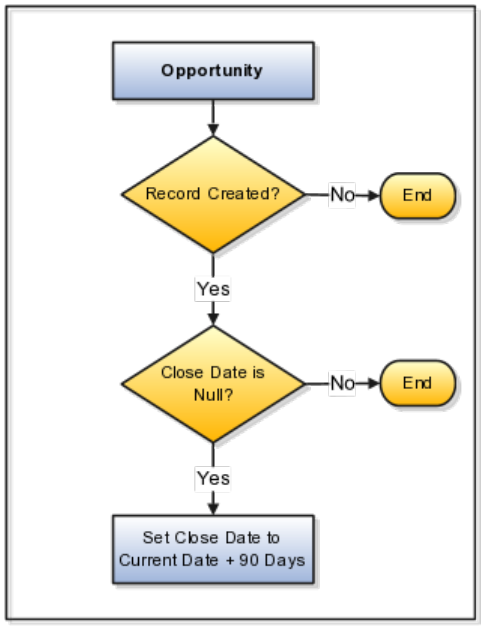
Object Workflows: Examples

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.

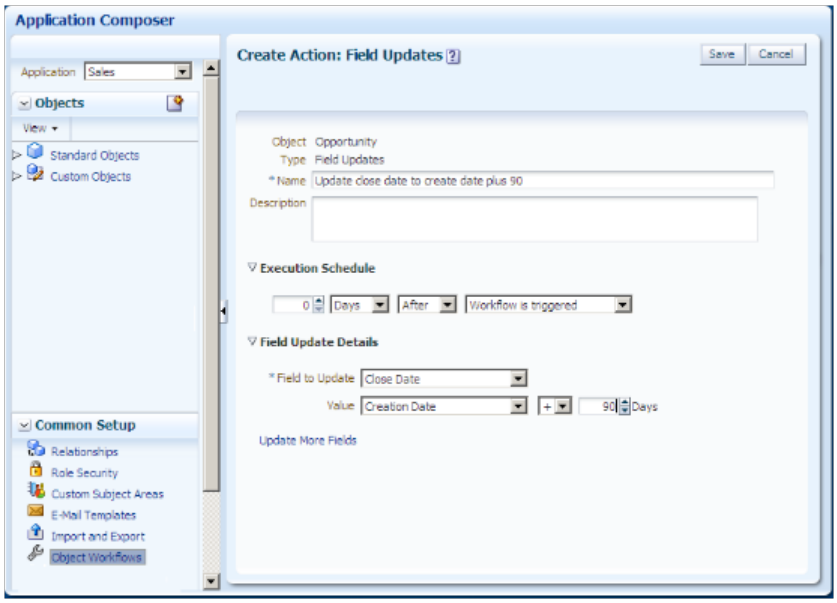
This figure illustrates the triggering of the event action for the business scenario.



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 editor 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.



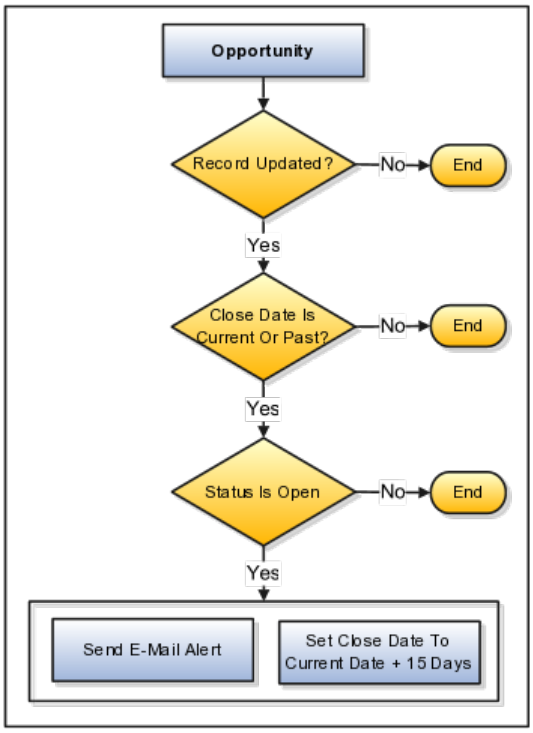
6. Provide the **Name** and **Description** for the field updates action and optionally set the **Execution Schedule**.
7. Under Field Update Details, select **Close Date** and set its value to **Creation Date** plus 90 days.
8. Save the event action.

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 editor 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.

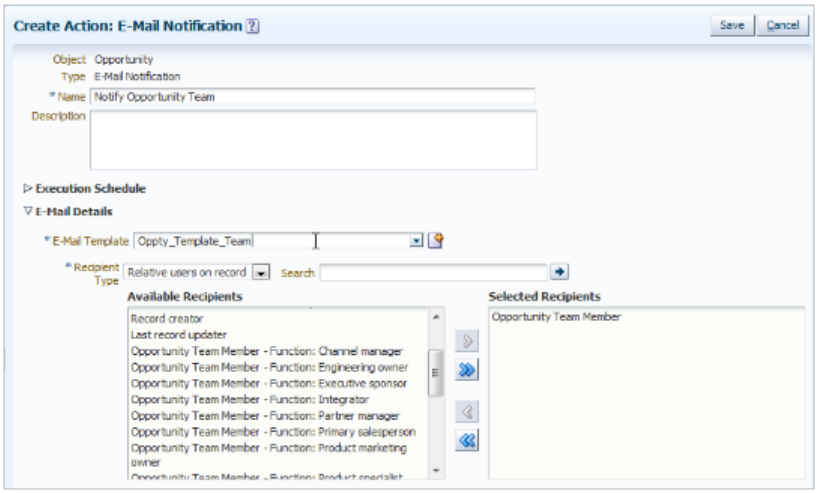
1. On the Create Object Workflow page, select **Field Updates** event action.

   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.

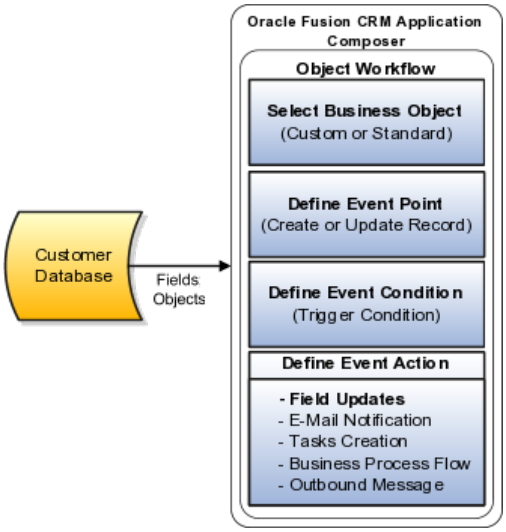
   When you create a template, you specify action-related text and field tokens that are populated at run time. See: E-Mail Templates: Explained topic.
4. Locate and specify **Specific users** under **Recipient Types**. See: recipient types in E-Mail Templates: Explained topic.
5. Click **Save**.

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 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

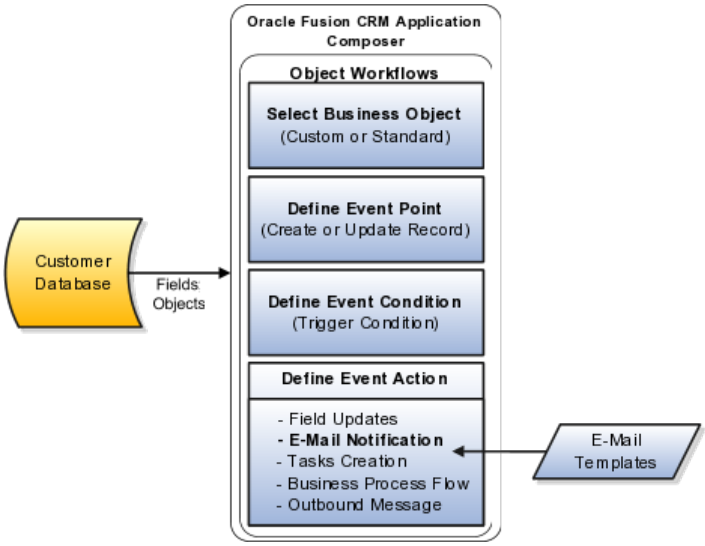
While defining an event action, you can optionally set an execution schedule that governs when an event action should happen. Else, the event actions are run immediately. When multiple event actions are configured, the Field Updates happen first before any other event action is triggered. For event actions 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 directs, 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

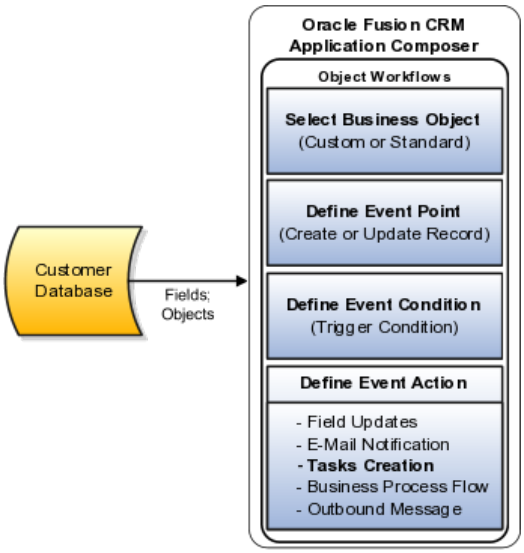
Oracle Fusion tasks integrate with BPEL human tasks and are available in the BPEL worklist application.

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

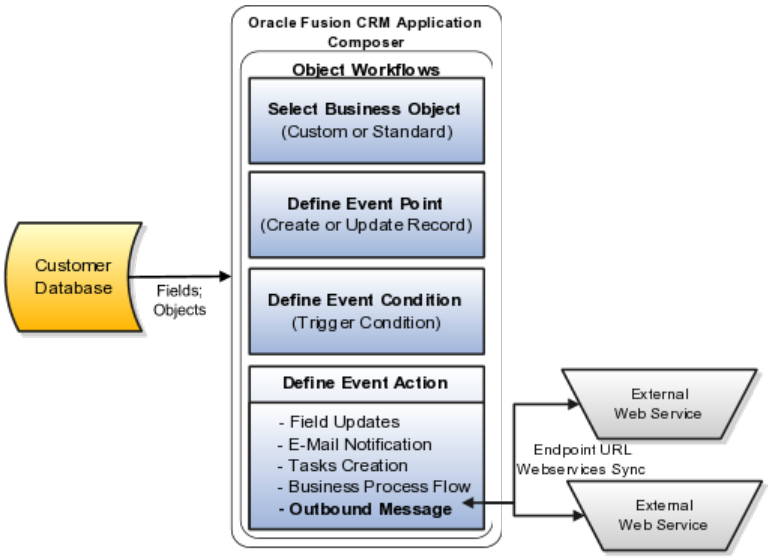
While defining an event action, you can optionally set an execution schedule that governs when an event action should happen. Else, the event actions are run immediately. When multiple event actions are configured, the Field Updates happen first before any other event action is triggered. For event actions 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 Fusion CRM, and must conform to the service WSDL of Oracle Fusion CRM.

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`.

For more information on basic administration involving deployment, configuration, and management of Web services, refer part II of the Oracle Fusion Middleware Security and Administrator's Guide for Web Services.

Outbound Messages Protection

You must perform the following setup activities when you are using the **Protect Message** option:

1. Oracle Fusion Application 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 Fusion 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 Fusion 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 with Oracle Fusion Customer Relationship Management (CRM) applications. The third-party Web service must conform to the service WSDL defined by Oracle Fusion CRM.

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 Fusion 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 $OBJECT_NAME$.

For more information about Oracle Enterprise Repository, see Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository.

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.

```
//Sample OutboundMessageService.wsdl
<wsdl:definitions
 name="OutboundMessageService"
 targetNamespace="http://xmlns.oracle.com/apps/crmCommon/content/
outboundMessage/"
 xmlns:errors="http://xmlns.oracle.com/adf/svc/errors/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
```

```
 xmlns:tns="http://xmlns.oracle.com/apps/crmCommon/content/
outboundMessage/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:types="http://xmlns.oracle.com/apps/crmCommon/content/
outboundMessage/types/"
 >
 <wsdl:import namespace="http://xmlns.oracle.com/adf/svc/errors/"
 location="ServiceException.wsdl"/>
 <wsdl:types>
 <schema xmlns="http://www.w3.org/2001/XMLSchema">
 <import namespace="http://xmlns.oracle.com/apps/crmCommon/content/
outboundMessage/types/"
 schemaLocation="OutboundMessageService.xsd"/>
 </schema>
 </wsdl:types>
 <wsdl:message name="OutboundMessageService_processOutboundMessage">
 <wsdl:part name="parameters" element="types:processOutboundMessage"/>
 </wsdl:message>
 <wsdl:message
 name="OutboundMessageService_processOutboundMessageResponse">
 <wsdl:part name="parameters"
 element="types:processOutboundMessageResponse"/>
 </wsdl:message>
 <wsdl:portType name="OutboundMessageService">
 <wsdl:documentation/>
 <wsdl:operation name="processOutboundMessage">
 <wsdl:input message="tns:OutboundMessageService_processOutboundMessage"/>
 <wsdl:output
 message="tns:OutboundMessageService_processOutboundMessageResponse"/>
 <wsdl:fault name="ServiceException" message="errors:ServiceException"/>
 </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="OutboundMessageServiceSoapHttp"
 type="tns:OutboundMessageService">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/
soap/http"/>
 <wsdl:operation name="processOutboundMessage">
 <soap:operation soapAction="http://xmlns.oracle.com/apps/crmCommon/
content/outboundMessage/processOutboundMessage"/>
 <wsdl:input>
 <soap:body use="literal"/>
 </wsdl:input>
 <wsdl:output>
 <soap:body use="literal"/>
 </wsdl:output>
 <wsdl:fault name="ServiceException">
 <soap:fault name="ServiceException" use="literal" encodingStyle=""/>
 </wsdl:fault>
 </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="OutboundMessageService">
 <wsdl:port name="OutboundMessageServiceSoapHttpPort"
 binding="tns:OutboundMessageServiceSoapHttp">
 <soap:address location="http://adc2111013:7101/OMInterface/
OutboundMessageService"/>
 </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

XSD File Example

This section provides an example of the OutboundMessageService.xsd, for your reference. The parameters are marked in the .xsd file as parameters, $OBJECT_TARGET_NAMESPACE$ and $OBJECT_NAME$.

```
//Sample: OutboundMessageService.xsd
```

```
<schema elementFormDefault="qualified" targetNamespace="http://
xmlns.oracle.com/apps/crmCommon/content/outboundMessage/types/"
 xmlns:ns0="http://xmlns.oracle.com/adf/svc/errors/"
 xmlns:ns1="$OBJECT_TARGET_NAMESPACE$"
 xmlns: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="$OBJECT_TARGET_NAMESPACE$"
 schemaLocation="$OBJECT_NAME$.xsd"/>
 <import namespace="http://xmlns.oracle.com/adf/svc/errors/"
 schemaLocation="ServiceException.xsd"/>
 <element name="processOutboundMessage">
 <complexType>
 <sequence>
 <element name="object" type="ns1:$OBJECT_NAME$"/>
 </sequence>
 </complexType>
 </element>
 <element name="processOutboundMessageResponse">
 <complexType>
 <sequence/>
 </complexType>
 </element>
</schema>
```

Security Policy

The outbound messages can use either an authentication-only client-side security policy or a transport-level security policy that protects the message during transfer. The default authentication-only policy used by object workflow outbound message is `oracle/wss10_saml_token_client_policy`. This policy includes Security Assertion Markup Language (SAML) tokens in outbound Simple Object Access Protocol (SOAP) request messages. This policy should only be used when the target web service is located within a secure network segment.

The corresponding service can use any compatible service policy, such as `oracle/wss10_saml_token_service_policy` or `oracle/wss_saml_or_username_token_service`.

When the service is outside your firewall, you must protect the message by selecting the **Protect Message** option on the Create Action: Outbound Message page. When protected, the outbound message uses `oracle/wss_username_token_over_ssl_client_policy` with one-way secure socket layer (SSL) enabled by default. The corresponding service must use a compatible service policy, such as `oracle/wss_username_token_over_ssl_service_policy`.

For more information on basic administration involving deployment, configuration, and management of Web services, refer part II of the Oracle Fusion Middleware Security and Administrator's Guide for Web Services.

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.

Note

If you are working in a sandbox, exit from the sandbox and create the e-mail template in the mainline application.

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 Oracle Fusion CRM 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 to Oracle Fusion. While defining an

e-mail template, the fields you select and embed in a template are specific to the selected object. The fields are populated with values at run time.

Options available for configuring e-mail templates include:

- Specifying **Name** and **Description** for the template.
- Attaching 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.

- Specifying whether an e-mail template is **Active**.

Note

You can use only the **Active** templates when creating object workflows for e-mail notifications.

- Inserting the **Fields** or **Functions**, as applicable. You can use **Functions** to insert **Date**, **Current date and time**, or a **Hyperlink to record**, which triggers an e-mail notification. You can also **Browse** and then insert the content of a local file in the body of the e-mail template.
- Performing 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 also 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.

Configuring Object Workflow

1. Click the **Navigator** menu.
2. Click the **more... >>** link.
3. Click the **Application Composer** link.

   You are on the Overview page of the Application Composer. This page shows the various tasks available to customize and extend your CRM 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 Groovy 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 editor. 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: 1) Only the BudgetAvailableDate is updated. 2) The BudgetAvailableDate is not null. 3) The opportunity record that you will create is updated. For your activity, the following

script has been written for you using the BudgetDateAvailable field: `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.
21. In the Actions region, click the Create button on the right of the Field Updates action.
22. You are on the Create Action: Field Updates page. Enter the desired information into the Name field. Enter "Update Close Date".
23. In the Execution Schedule region, keep the default setting, which is to update fields right away when the workflow is triggered.
24. In the Field Update Details region, click the Field to Update list.
25. Click the Close Date list item.
26. Click the Value list.
27. Click the Date Budget Available list item.
28. Make sure the '+' operator is selected, and enter the desired information in the Days field. Enter "7".
29. Click the Update More Fields link.
30. Click the Field to Update list.
31. Click the Level of Risk list item.

    In this section, 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.

32. Click the Populate with next value in list option.
33. Click the Update More Fields link.
34. Click the Field to Update list.
35. Click the Strategic Value list item. .

    Note: If there are multiple Strategic Value fields in the list, click the first one

36. Click the list on the right of the Value group of options.
37. Click the Medium list item.

38. Save the Field Update event action. In the upper-right region of the page, click the Save button.
39. 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.
40. In the Actions region, click the Create button on the right of the Task Creation event action.
41. Enter the desired information into the Name field. Enter "Follow Up Call".
42. In the Task Details region, enter the desired information into the Subject field. Enter "Follow up with customer on budget available date".

    Note: Expand the Task Details region if not already expanded.
43. 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".
44. Click the field-token list on the right of the Subject field.
45. Click the Customer list item.

    Note: If there are multiple entries for Customer, select the first one.
46. Click the << Insert button.
47. 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.
48. 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: ".
49. 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 ([$PrimaryContactEmailAddress$])`
50. Click after "Opportunity" in the Description text.
51. Click the field-token list on the right of the Description field.
52. Click the Opportunity Name list item
53. Click the <<Insert button
54. Click after "Follow up with" in the Description text.
55. Click the field-token list on the right of the Description field.
56. Click the Primary Contact list item.
57. Click the <<Insert button
58. Click after "Email ID:" in the Description text.
59. Click the Primary Contact E-Mail list item.
60. Click the <<Insert button
61. You will now set the task's Due Date to be 3 days after the new Budget Available Date. Click the Due Date list.

62. Click the Date Budget Available list item.
63. Make sure the '+' operator is selected, and use the increment button (up arrow) to set the value in the Days field to 3.
64. Click the Start Date list.
65. Click the Date Budget Available list item. Accept the default 0 (zero) in the Days field.
66. Click the Owner list.
67. Click the OwnerResourcePartyid list item.
68. Click the Assignees list
69. Click the OpportunityResource Team list item.
70. Click the Category list.
71. Click the Call, outbound list item.
72. Click the Priority list.
73. Click the 1 - Very high list item.
74. Save the Task Creation event action. In the upper-right region of the page, click the Save button.
75. 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.
76. You are on the Object Workflows page. 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.
77. In the Common Setup pane on the left, click the E-Mail Templates link.
78. 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.
79. In the Search Results region, click the Actions menu.
80. Click the Create menu item.
81. You are on the Create E-Mail Template page. Click the Object list.
82. Click the Opportunity list item.
83. Enter the desired information into the Name field. Enter "Budget Available Date Update".
84. Enter the desired information into the E-Mail Subject field. Enter "Opportunity customer budget available date has changed".
85. 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.
86. Click the field-token list on the right of the E-Mail Subject field.
87. Click the Opportunity Name list item.
88. Click the <<Insert button

89. 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 .".
90. In the E-Mail Body region, use the Fields list-item of the Select list to insert the following field tokens: - `Customer ([$PartyUniqueName $]) - Opportunity Name ([$Name$]), - Date Budget Available ([$BudgetAvailableDate$]) - Close Date ([$EffectiveDate$])`

    Note: Rich text formatting is available only if your e-mail account supports HTML format e-mail.
91. Click after "- " in the e-mail body.
92. In the E-Mail Body region, click the field-token list on the left of the Insert button.
93. Click the Customer list item.
94. Click the Insert button.
95. Click after "regarding opportunity" in the e-mail body.
96. Click the field-token list.
97. Click the Opportunity Name list item.
98. Click the Insert button.
99. Click after "has moved to" in the e-mail body.
100. Click the field-token list.
101. Click the Date Budget Available list item.
102. Click the Insert button.
103. Click after "close date is" in the e-mail body.
104. Click the field-token list.
105. Click the Close Date list item.
106. Click the Insert button.
107. Save the E-Mail Template. In the upper-right region of the page, click the Save button.
108. You are back to the E-Mail Templates page. In the Common Setup pane on the left, click the Object Workflows link.
109. You are on the Object Workflows page. Search for the object workflow that you were creating before you moved to configuring e-mail templates. Click the Object list.
110. Click the Opportunity list item.
111. Click the Search button.
112. 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.
113. Click the Edit button.

114. 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
115. In the Actions region, click the Create button on the right of the E-Mail Notification event action.
116. 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".
117. Select the e-mail template that you created. In the E-Mail Details region, click the E-Mail Template list.
118. From the list of e-mail templates, identify and select the template that you created. Click the Budget Available Date Update template.
119. Click the Recipient Type list.
120. Click the Specific e-mail addresses list item.
121. Enter the desired information into the E-Mail Addresses field.
122. Save the E-Mail Notification event action. In the upper-right region of the page, click the Save button.
123. 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.
124. 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 create in Activity 2.

Triggering the Workflow and Invoking Event Actions

In this section, you will trigger (launch) the object workflow that you created in the previous activity by entering the budget availability date for an opportunity.

- 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. 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). In the Name field, enter "50 Solar Green Servers".
5. In the Sales Account field, enter "Solar Inc (KIRKLAND, US)".
6. Click the Sales Account search button adjacent to the Sales Account field.
7. 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.

8. In the Search Results region, click the Solar Inc (Kirkland, US) cell.

   Ensure that the All Sales Accounts tab is selected.
9. Click the OK button.
10. In the Revenue Items region, click the Add Row button.
11. Enter the desired information into the Name field. Enter "DG 150 Green Servers".
12. Enter the desired information into the Quantity field. Enter "50".
13. Enter the desired information into the Estimated Price field. Enter "2000".
14. In the upper-right region of the page, click the Save and Edit button.
15. 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
16. 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.
17. Under the Show Less area in the Summary region, click the Level of Risk list.
18. Click the High list item.
19. In the upper-right region of the page, click the Save button.
20. You are still on the Edit Opportunity page. In the Activity Center region, click the Tasks tab.
21. Currently there are no Tasks with your sign in initials.
22. In the Additional Details region, click the Opportunity Team tab.
23. In the Team Members table, locate your sign-in username. You will now add another team member: Mateo Lopez.
24. In the Opportunity Team tab, click the Add Team Members button.
25. You are on the Select and Add: Team Members popup.
26. Find a team member and click the Done button.
27. Search and add a contact to the opportunity. This will be the primary contact for the opportunity.
28. In the Additional Details region, click the Contacts tab.
29. In the Contacts tab, click the Add Row button.
30. Enter the desired information into the Name field. As you type in the contact name, the contact name and e-mail ID appear.
31. Click the desired contact.
32. In the upper-right region of the page, click the Save and Close button.

33. 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.
34. From the table in the Opportunities region, identify and select the opportunity record that you created. Click the 50 Solar Green Servers link.
35. You are on the Edit Opportunity page. Before you proceed, expand the Show More area in the Summary region if not already expanded.
36. Under the Show Less area in the Summary region, click the Date Budget Available (calendar) button.
37. Click 9 as the date, or click any future date.
38. In the upper-right region of the page, click the Save and Close button.

    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.
39. 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.
40. In the Summary region, Close Date is now set to 7 days after the Date Budget Available value.
41. Under the Show Less area in the Summary region, Strategic Value is now set to Medium.
42. 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.
43. You have verified the invocation of the Field Updates event action.
44. 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
45. In the Activity Center region, click the Due Date list under the Tasks tab.
46. Click the All Tasks list item.
47. In the Tasks tab, click the arrow button adjacent to the Category list.
48. 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.

49. Click the Done button.
50. In the upper-right region of the page, click the Save and Close button.
51. You have verified the invocation of the Task Creation event action. Next, verify the invocation of the E-Mail Notification event action.
52. Locate the Subject of your e-mail. Click the Subject link.
53. 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.
54. In the upper-right region of the page, click the Sign Out link.
55. 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 applications involved in the approval flow process:

- Oracle Fusion CRM Application Composer

  Oracle Fusion CRM Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application developers, to customize and extend an Oracle Fusion CRM application.

  For more information on how you customize and extend your applications using CRM application composer, see Extending CRM Applications: 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 Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

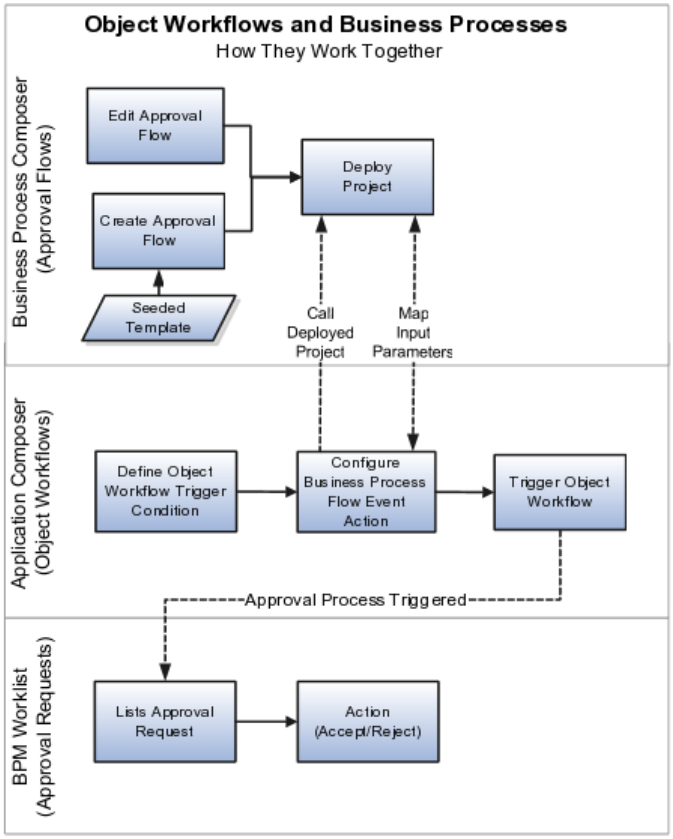
- Oracle Business Process Management (BPM) Worklist application
  Oracle BPM Worklist application displays tasks or approvals that are assigned to a user or a group. The worklist tasks appear on the Home page of the Oracle Fusion Applications, where the assignee approves or rejects an approval request.

Note

You do not perform any configuration in the worklist application as part of configuring object workflows and business processes.

For information on the worklist application, see "Using Approval Management" in Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

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, refer Oracle Fusion Applications CRM Extensibility Guide on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.
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 application.

---

a. **Approvalstatus_c** field: This is 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

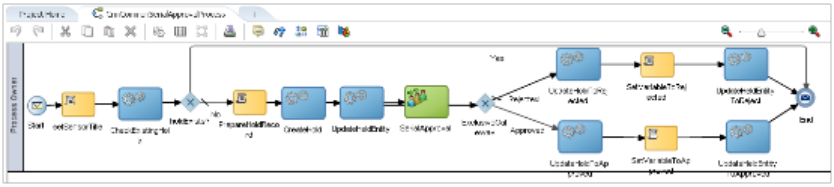
You must have at least one business process flow deployed for use before defining an object workflow to trigger an approval process.

To access Oracle Business Process Composer:

1. Navigate to the Application Composer using the main menu of the Oracle Fusion Applications.
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**.

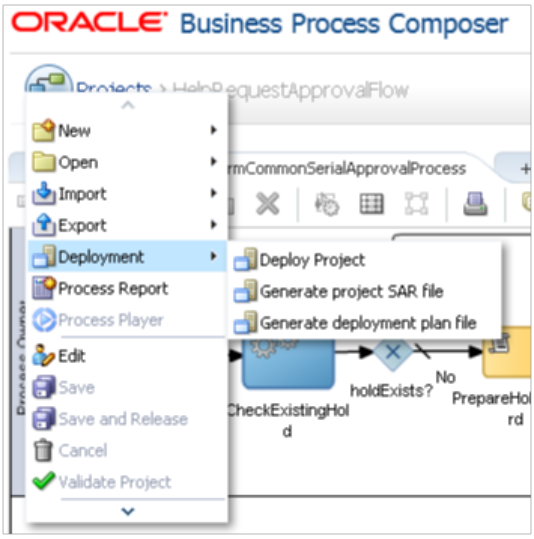
      If popup blocker is enabled on your browser, the business process composer may not open after you click **OK**, and your browser may get locked. As a workaround, use the browser back button to go back to the Business Processes option in the application composer, and click **Edit** for the business process that you were creating.

      This opens the Oracle Business Process Composer in a separate Window.



   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:

Field	Value
Revision ID	Provide an identifiable ID.
Deployer Username	Your sign in name for Oracle Fusion.
Password	Your password for Oracle Fusion.
Mark composite revision as default	Select this option.

For more information about the options in this popup and deploying business processes, refer "Deploying a BPM Project" in Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

5. Click **Deploy**.

   A confirmation message appears when your project has been deployed successfully.

6. Close the Oracle Business Process Composer.
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.

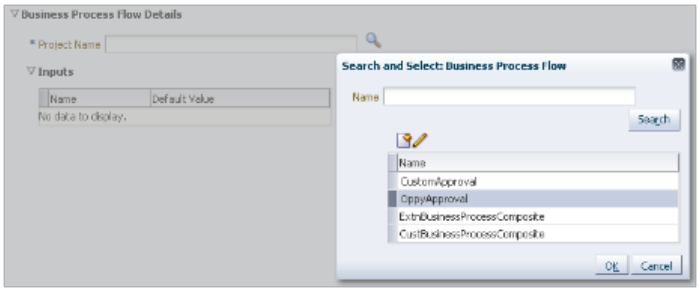
1. From the overview page of the application composer, select **Object Workflows**.
2. On the Create Object Workflows Page, select the object to which you added the custom fields.
3. Enter a name for your object workflow.
4. Define a trigger condition using When a record is updated event point.

   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.



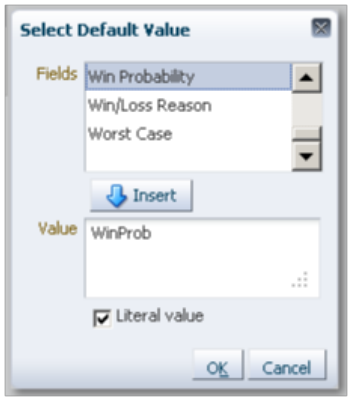
   d. Specify the **Input** parameters that you want to pass to the deployed approval flow.

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`.

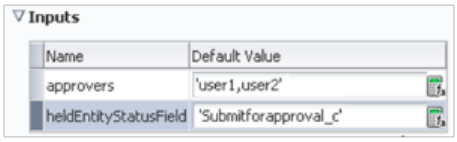
This figure illustrates how you select the fields to map from object workflows.



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 applications involved in the approval flow process:

- Oracle Fusion CRM Application Composer

  Oracle Fusion CRM Application Composer is a browser-based configuration tool that enables business analysts and administrators, not just application developers, to customize and extend an Oracle Fusion CRM application.

  For more information on how you customize and extend your applications using CRM application composer, see Extending CRM Applications: 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 Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

- 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 the Oracle Fusion Applications, 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.

For information on the worklist application, see "Using Approval Management" in Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

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.

   For more information about Oracle MDS, refer "Understanding a Metadata Repository" in Oracle Fusion Middleware Administrator's Guide on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

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**.

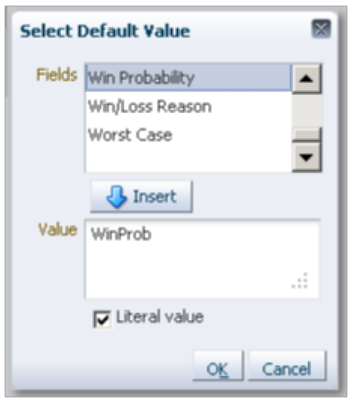
Carefully specify the parameters that you want to map from object workflows to the approval flow in business process composer. The input parameters supply the fields or values to the business process flow when the object workflow is triggered.

You can map only the following parameters for approval flows:

Parameter	Description
approvers	Specifies the owner or the one who should receive the approval request.
heldEntityStatusField	Specifies the field that displays the status of the approval request.

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.

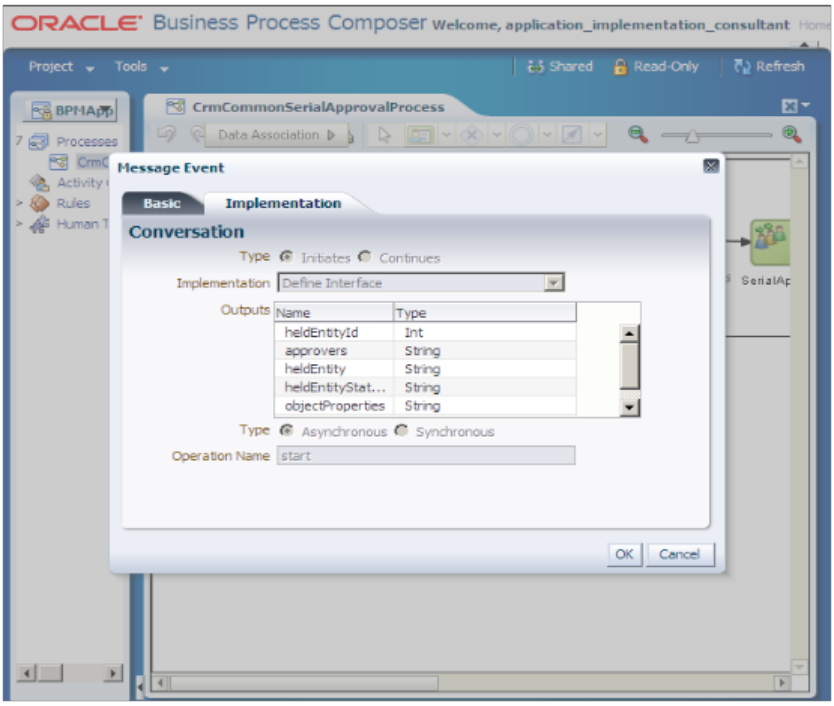


To pass runtime values, do not define the parameters as **Literal**. In this case, you do not select **Literal** in the **Select Default Value** dialog, or you type-in the parameters without quotes. For example, to pass a runtime field for approver, enter `user1`.

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.



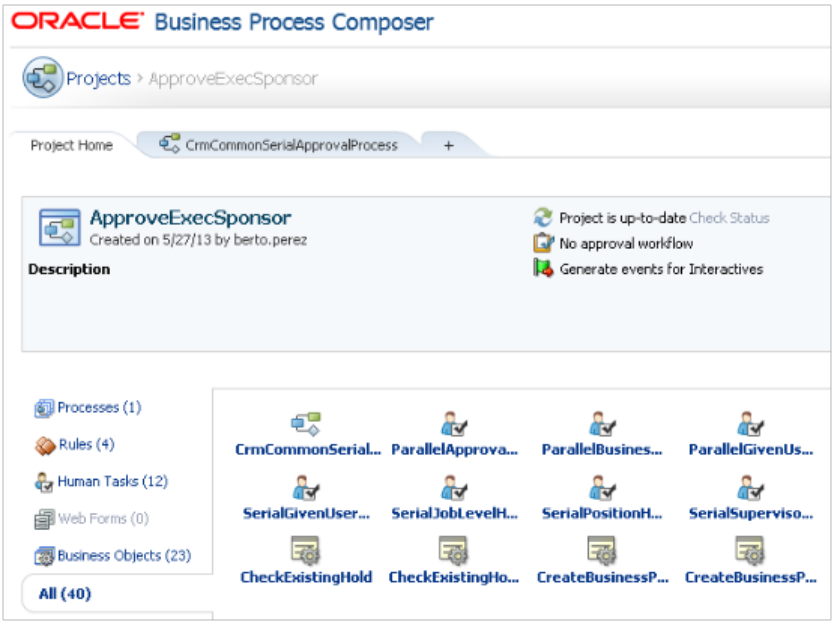
Notice the **heldEntityStatus** and **approvers** parameters. These are the parameters you are mapping to from the object workflows.

5. **Using the seeded template**.

   The seeded template **ExtnBusinessProcess** has the basic **CrmCommonSerialApprovalProcess** process flow. This template contains an approval process covering a typical CRM 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.

For information on working with business rules, services, and tasks and how to model process flows, refer Oracle Fusion Middleware Modeling and Implementation Guide for Oracle Business Process Management at Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

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 Fusion Applications as an administrator.
2. Create a sandbox and activate it.

   Sandboxes in Oracle Fusion Applications 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 application; 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 application unless published.

---

19. Publish your sandbox to make these changes available for use in the mainline application.

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 application customizations overwrite the mainline application'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:

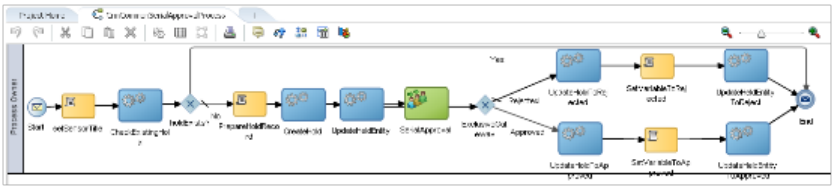
1. Under the Administration menu, select **Manage Sandboxes...**.
2. Select your sandbox and click **Publish**.
3. Click the Navigator menu at the top.
4. Click **Lead Qualification** under **Sales**.
5. Click **Create Lead**.
6. Verify that **Large Deal Approval** field has been added to the Create 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.

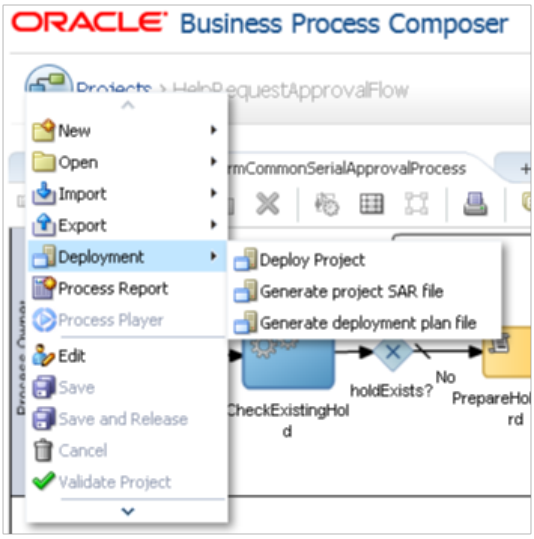
To create and deploy a process using Oracle Business Process Composer:

1. Navigate to the Application Composer using the main menu of Oracle Fusion Applications.
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:

Field	Value
Revision ID	Provide an identifiable ID.
Deployer Username	Your sign in name for Oracle Fusion Applications.
Password	Your password for Oracle Fusion Applications.
Mark composite revision as default	Select this option.

For more information about the options in this popup and deploying business processes, refer "Deploying a BPM Project" in Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

5. Click **Deploy**.

   A confirmation message appears when your project has been deployed successfully.

6. Close the Oracle Business Process Composer.

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 **Groovy Builder** icon.

   In the groovy builder, enter the following event condition:

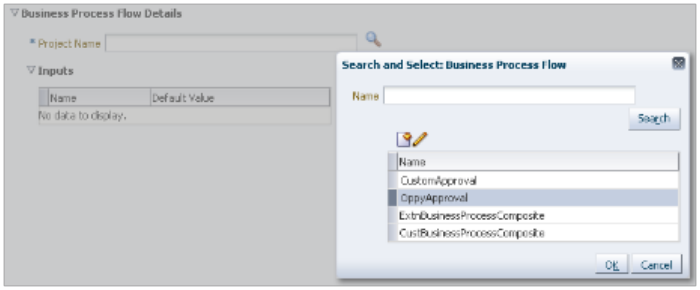
   ```
   if(DealAmount>10000 && CurrencyCode=='USD') return true;
   ```

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.

Name	Default Value
approvers	'SALES_MANAGER'
heldEntityStatusField	'LargeDealApproval_c'

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 Fusion Applications 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.
9. Sign out of Oracle Fusion Applications.

10. Sign in to Oracle Fusion Applications 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.

12. Sign out of Oracle Fusion Applications.
13. Sign in to Oracle Fusion Applications with a SALES_REPRESENTATIVE role.
14. Navigate to **Lead Qualification**.
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.

8

Application Composer: Creating Custom Subject Areas

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 for an Oracle Fusion CRM application, or create a report subject area using a wizard. The report subject areas that you create using the wizard are called custom subject areas and are 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 Fusion CRM represents an entity; it can be, for example, a sales account or an opportunity.
  - Standard object: An object that is delivered when the Oracle Fusion CRM is first used is called standard object, for example, opportunity.
  - Custom object: A new object that is added during the deployment of Oracle Fusion CRM is called a custom object.
- **Fields or attributes**: A field, also called an attribute, is where entity information is stored in Oracle Fusion CRM. 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 Fusion CRM is delivered with several entities, called standard objects, which store entity-specific information such as Marketing Campaign, Opportunities, and more.

Oracle Fusion CRM 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 Tickets.

In the 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 Fusion CRM is considered standard; therefore, an object that is delivered when the Oracle Fusion CRM is first used is called a standard object, for example, Opportunity.

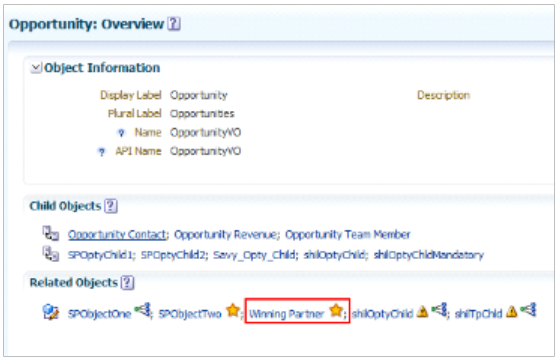
Any configuration performed by the customer and later added to Oracle Fusion CRM is considered custom. If a new object or subject area is added during the deployment of Oracle Fusion CRM, it is then custom or called a custom object, custom field or custom attribute, or custom subject area.

When planning the design of your custom subject area, use the Application Composer to identity standard and custom objects or fields within Oracle Fusion CRM.

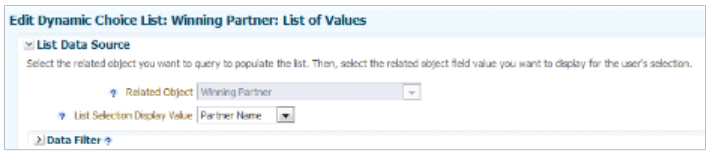
Primary Object

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 Fusion CRM application.

Note

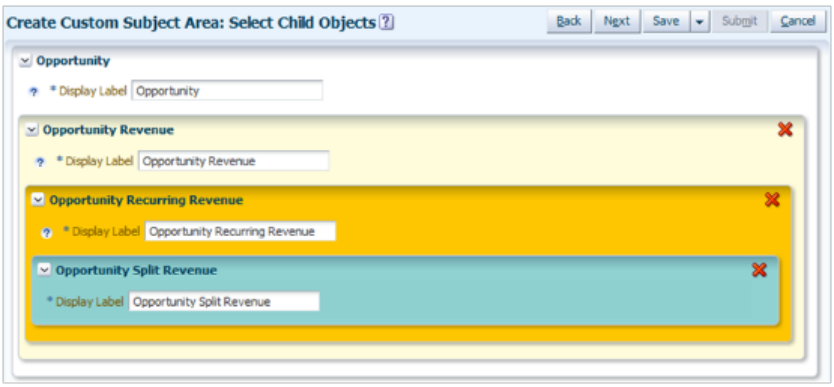
The common object Notes and Tasks are not reportable. They cannot be included in a custom subject area.

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.



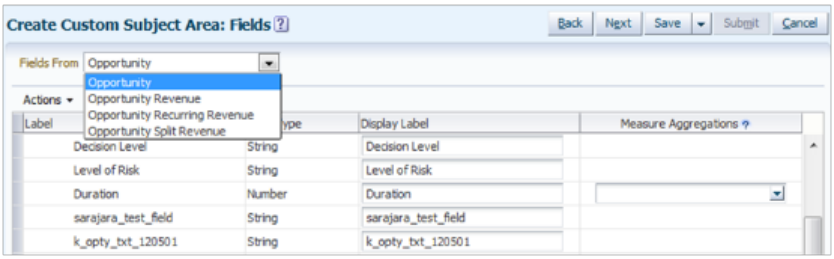
Note

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

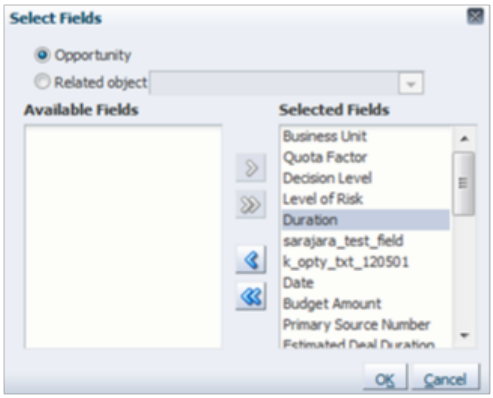
Related Objects

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 to the custom subject area.



The same process is followed for any other child object added to the custom subject area.

Fields or Attributes in Custom Subject Areas: Explained

A field, also called an attribute, is where entity information is stored in Oracle Fusion CRM. An opportunity name is stored in a field under the opportunity entity or object. Oracle Fusion CRM 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
- Fixed choice lists
- Dynamic choice lists

Custom Attributes on Standard Objects

For key objects in Oracle Fusion CRM 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.

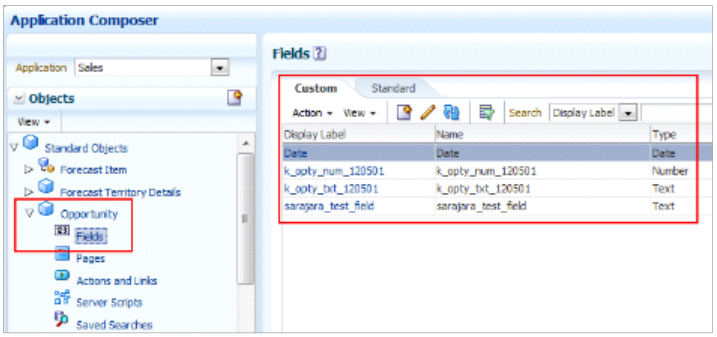
When you create a custom field, you can create reports for the following data types:

- Boolean
- Number
- Currency
- Date
- String
- Percentage
- Phone
- Date time

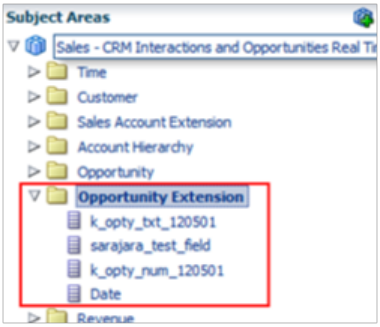
To create reports using extension dimension attributes:

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
    - Total
    - Count
    - Count Distinct
    - Maximum
    - Minimum
    - First
    - Last
    - Median
    - Standard Deviation
    - Standard Deviation Population
  - For date fields, a measure can be:
    - All
    - Maximum
    - Minimum

For more information about selecting the data for a report using the Oracle Business Intelligence Composer, see "Using BI Composer to Work with Analyses" in Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition)on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.

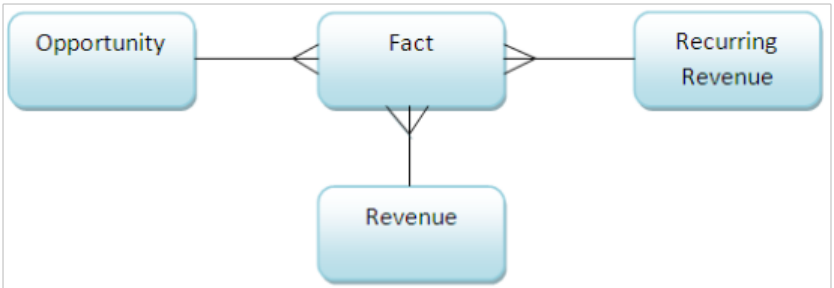
You can select measures based on your reporting needs. For example, you can use measures to view product sales per store, state, or country. Or, to view the number of support tickets opened or closed per day, week, or month, and so on.

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.



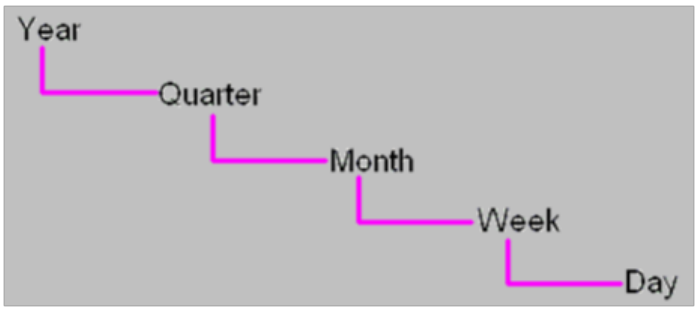
Note

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.



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 Fusion Applications:

1. Navigate to **Application Composer** from the **Navigator** menu.
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-grand 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.

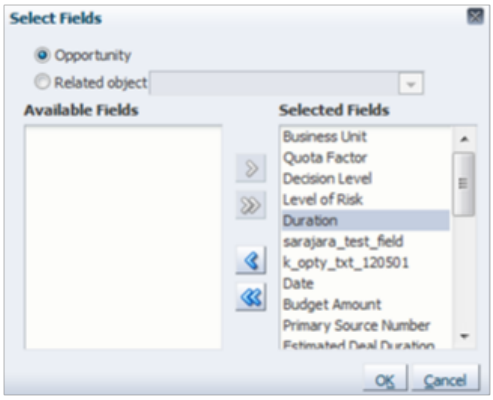
10. Click the + icon.

    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:

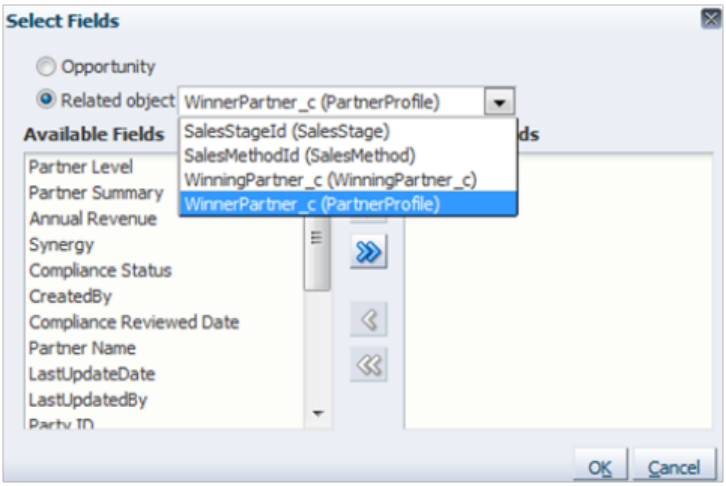
1. Select the opportunity object in the **Fields From** drop down list.
2. Click **Select Fields** to display the Select Fields screen



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.

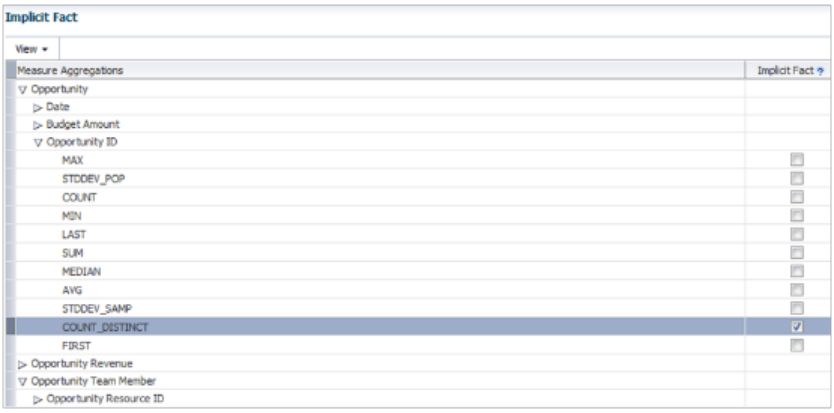
5. Repeat steps 1 to 4 for removing undesired fields or adding related objects from each object that is added to the custom subject area.
6. Select the desired measures to be generated for number, date, or currency fields type from all available objects added to the custom subject area.

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.

---

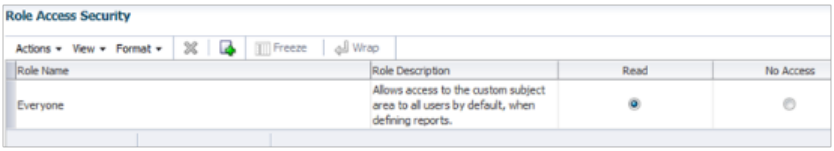
7. Select the **Implicit Fact** column as desired.



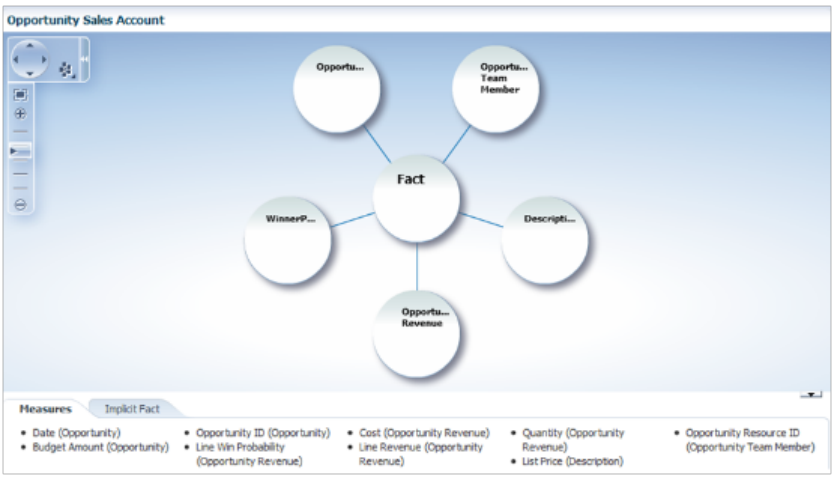
8. Click **Next**.
9. If required, select the **Date** columns for date leveling functionality in the custom subject area.
10. Select the desired security level for the **Everyone** Role Name, which is added by default or add additional Role Names by clicking in the **+** icon and define the security 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 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 Fusion CRM 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.

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.

For more information about selecting the data for a report using the Oracle Business Intelligence Composer, see "Using BI Composer to Work with Analyses" in Oracle Fusion Middleware User's Guide for Oracle Business Intelligence Enterprise Edition (Oracle Fusion Applications Edition) on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.

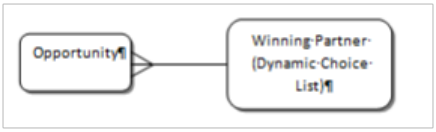
Creating a Custom Object and Associating it With a Custom Dynamic Choice List Field: Worked Example

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.

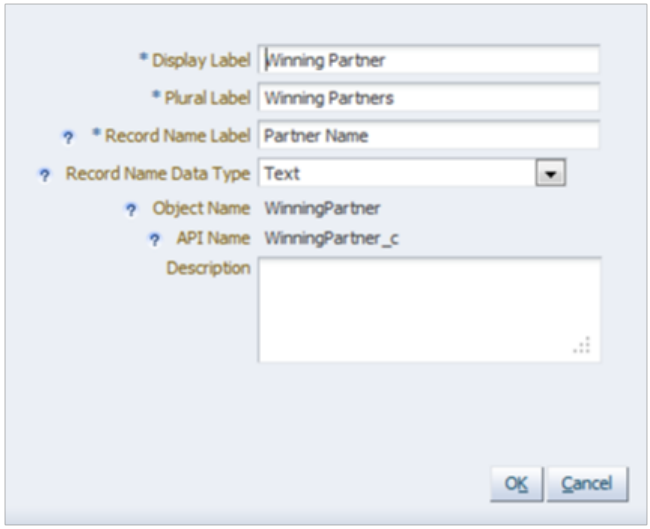


Winning Partner Custom Object Configuration

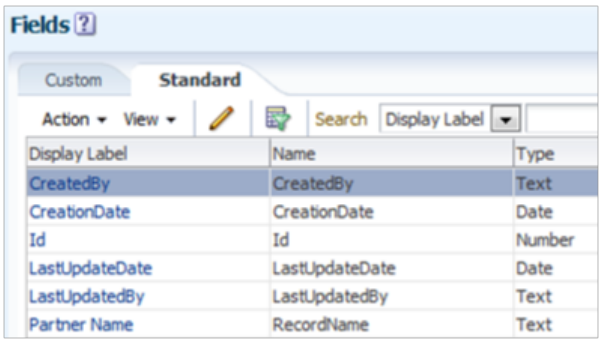
In this section, you are creating a custom object called "Winning Partner". Before making any customization in Oracle Fusion CRM application, you should have a sandbox session active.

For more information on sandboxes and how to use them, refer Guidelines for customizing Oracle Fusion CRM using Application Composer and Sandboxes (Doc ID 1484889.1) on My Oracle Support (MOS) at https://support.oracle.com/.

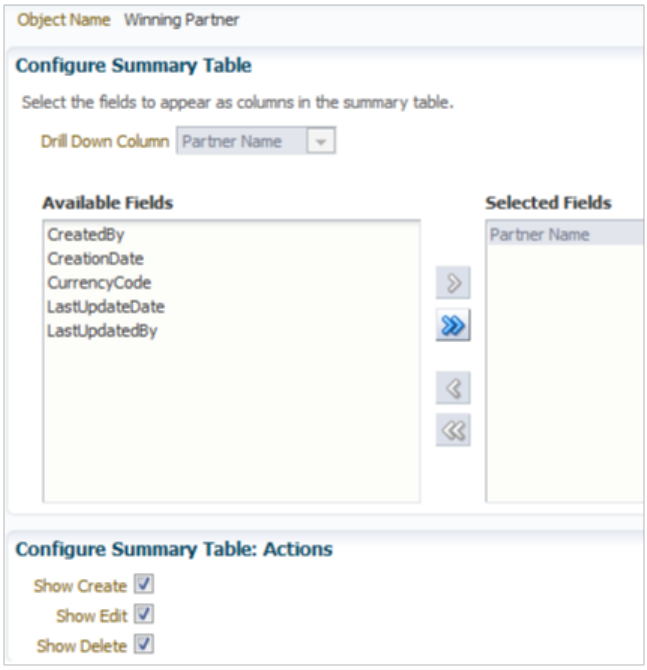
1. When your Sandbox session is active, click the **Navigator** menu.
2. Click on **Application Composer** menu item.
   You are on the main page of the Application Composer. You are adding an attribute to a standard object that belongs to the Sales application.
3. In the left pane, click the **Application** 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:



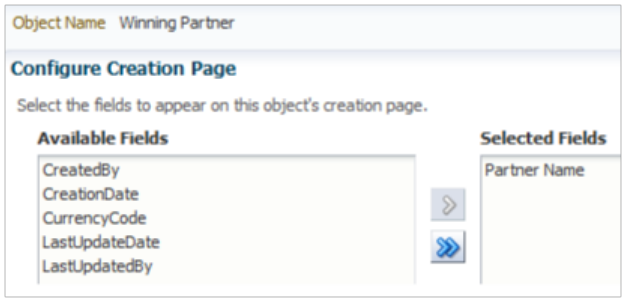
   The following standard fields are created automatically for new custom objects:



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:

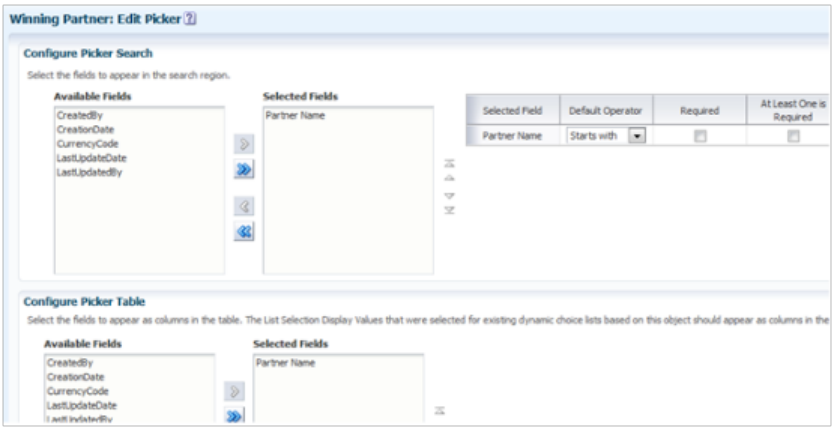


c. Define the Creation Page as shown:



d. Define the Details Page, just the Default Summary

9. Configure a picker for the Winning Partner custom object as follows:

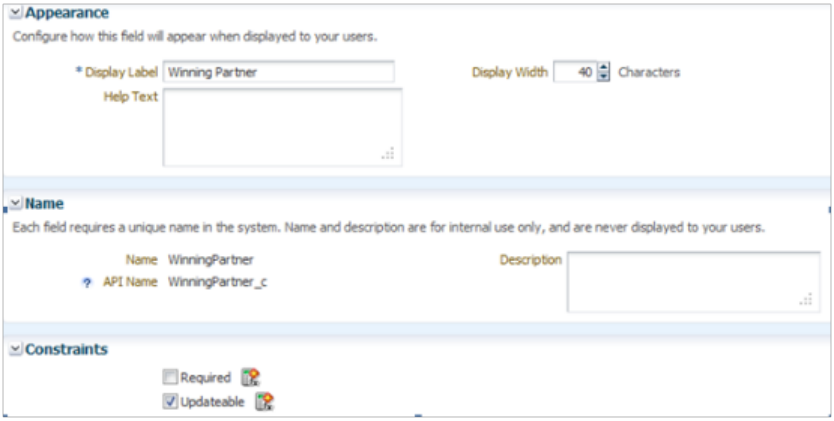


Winning Partner Dynamic Choice List

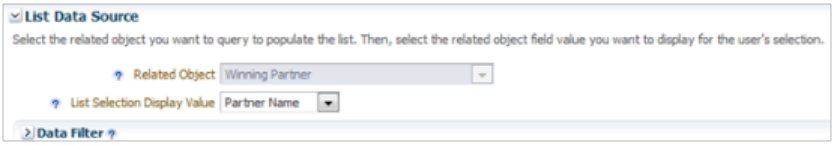
In this section, you are creating a custom dynamic choice list under the Opportunity object based on the created Winning Partner custom object.

To do this, navigate to the 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:

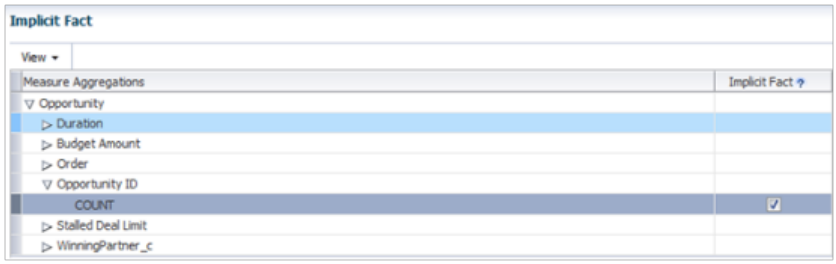
1. In the **Application** list in the left pane, select **Sales** application. This is where you created your Winning Partner customizations.
2. Navigate to the Custom Subject Area page
3. Click on **Create** icon
4. Enter "Opportunity Winning Partners" in the **Label** field.
5. Select **Opportunity** 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.

Label	Data Type	Display Label	Measure Aggregations
sarajara_test_field	String	sarajara_test_field	
k_opty_txt_120501	String	k_opty_txt_120501	
Date	Date	Date	
Budget Amount	Number	Budget Amount	Total
Primary Source Number	String	Primary Source Number	
Estimated Deal Duration	String	Estimated Deal Duration	
Key Internal Sponsor	String	Key Internal Sponsor	
Line of Business	String	Line of Business	
Order	Number	Order	Count
Phase	String	Phase	
Comments	String	Comments	
Budgeted	String	Budgeted	
Opportunity ID	Number	Opportunity ID	Count
Strategic Value	String	Strategic Value	
Opportunity Number	String	Opportunity Number	
Date Budget Available	Date	Date Budget Available	
Stalled Deal Limit	Number	Stalled Deal Limit	Total
Sales Objective	String	Sales Objective	
Opportunity Name	String	Opportunity Name	
k_opty_num_120501	Number	k_opty_num_120501	
Win/Loss Reason	String	Win/Loss Reason	
▽ WinningPartner_c		WinningPartner_c	
Id	Number	Id	Count
Partner Name	String	Partner Name	

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**.
14. Select the Implicit Fact column.



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.

20. Click **Submit**.

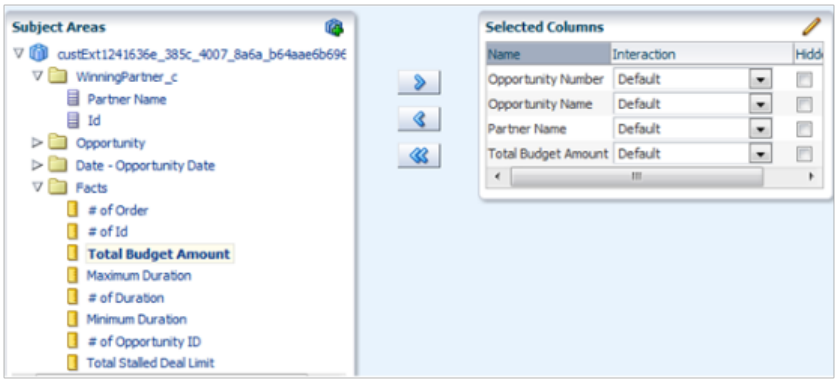
    Wait until you are navigated back to the custom subject area table.
21. Click **Refresh** icon after few minutes until you see the status of the created custom subject area as **OK**.

Creating and Viewing a Report for the Custom Subject Area

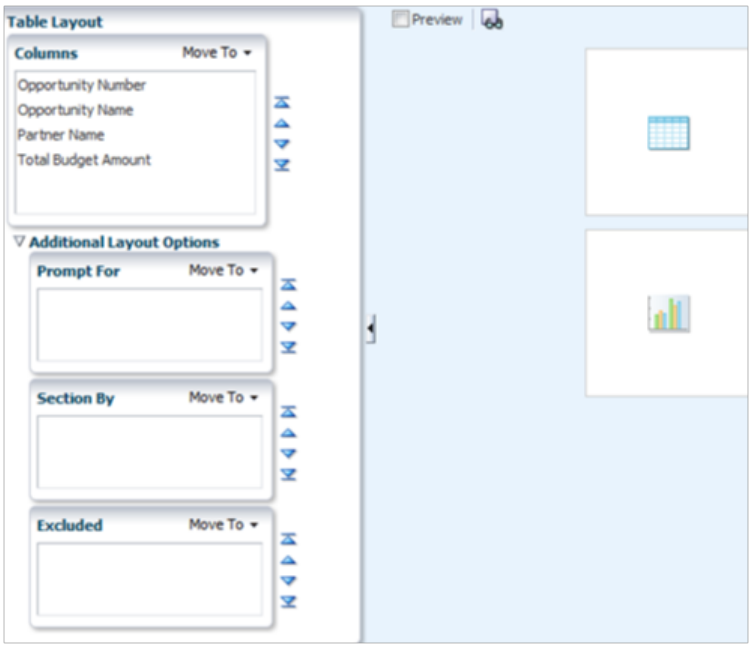
In this section, you first create a report using the published custom subject and then view it.

1. Click the **Navigator** menu item.
2. Under the Tools category, click **Reports and Analytics**.
3. Click **Create**, and then click **Analysis**.

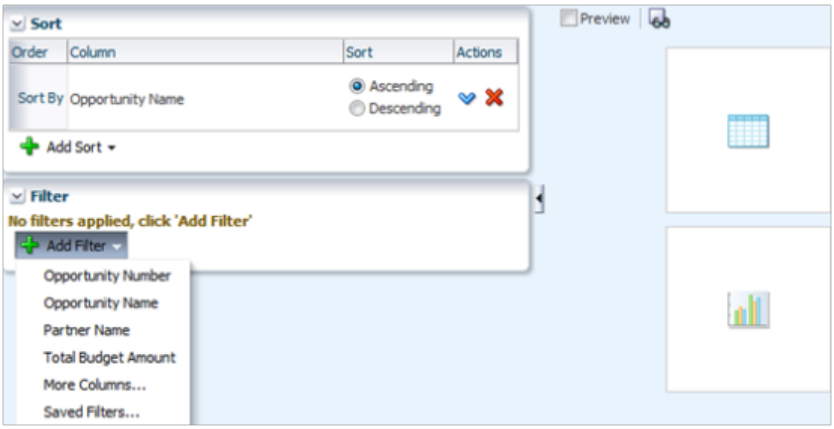
4. Under **Select Subject Area**, locate and select the published **Custom: Opportunity Winning Partners** custom subject area.
5. Review the custom subject area: Expand each folder and review its attributes and measures.
6. Follow the steps in the report creation wizard to create a report:
   a. Add 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.

8. Select **Opportunity Winning Partners**.

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 customization in Oracle Fusion CRM application, 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 Fusion CRM using Application Composer and Sandboxes (Doc ID 1484889.1) on My Oracle Support (MOS) at https://support.oracle.com/.

Adding Custom Fields Using CRM Application Composer

To add custom fields:

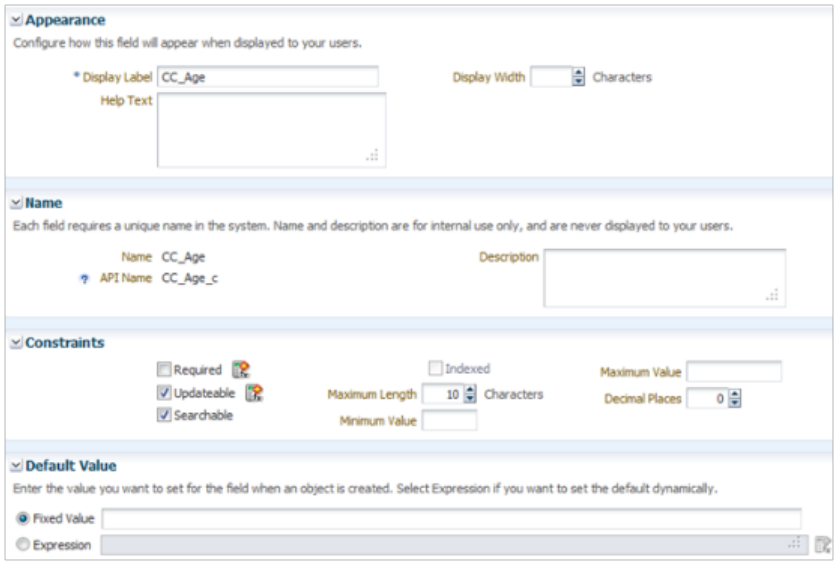
1. When your sandbox session is active, click the **Navigator** menu.
2. Click **Application Composer**.

   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.

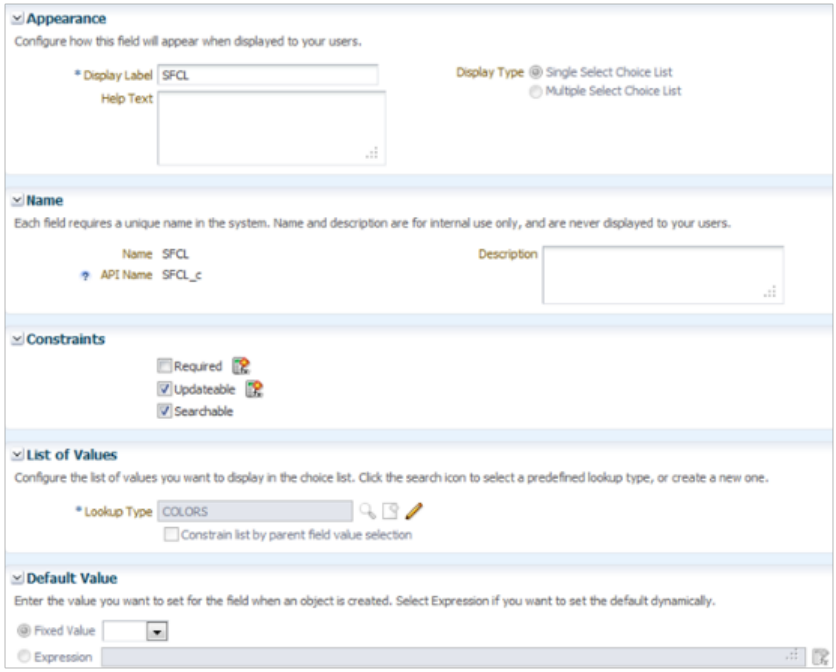
   The Custom tab is displayed on the Fields screen
7. Click the create a custom field icon to start adding the following fields:

   - Add a **Text** field.

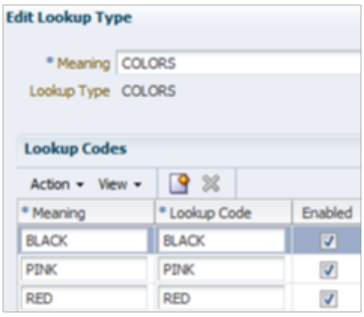
   - Add a **Number** field.



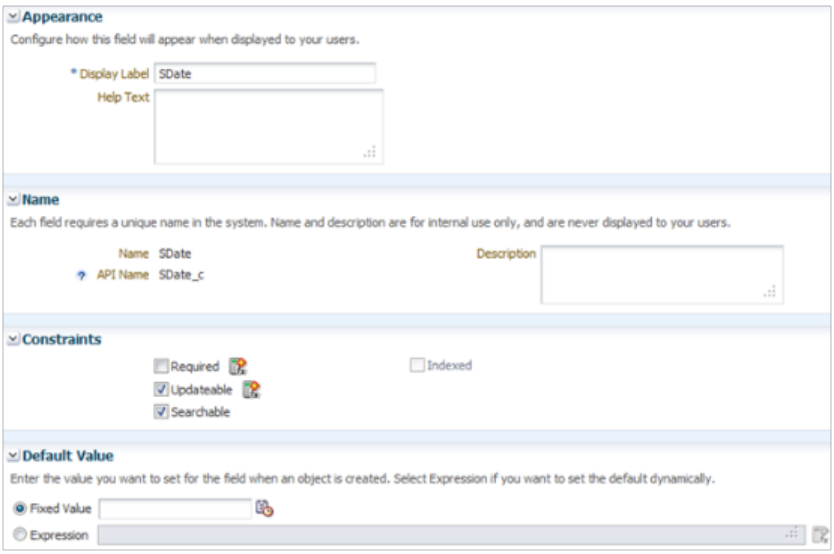
- Add a **Fixed Choice List** field



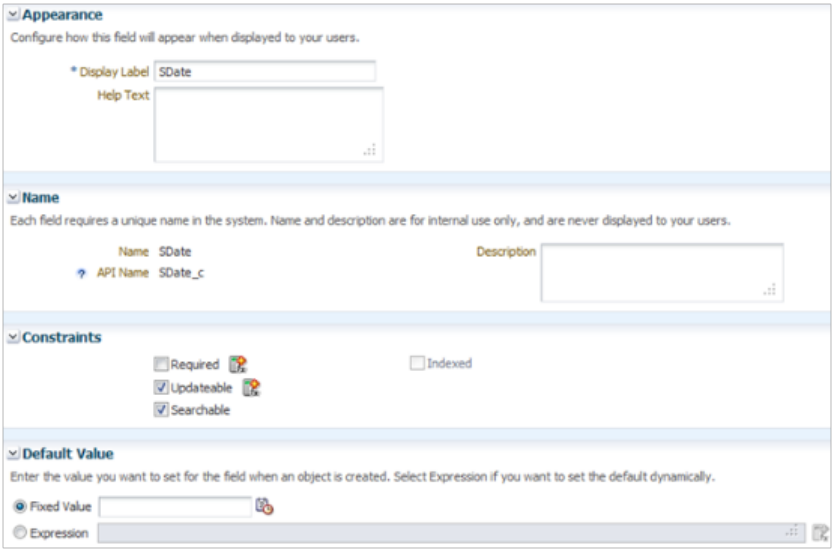
- Create the Lookup Values for the **Fixed Choice List**.



- Create a **Date** field.



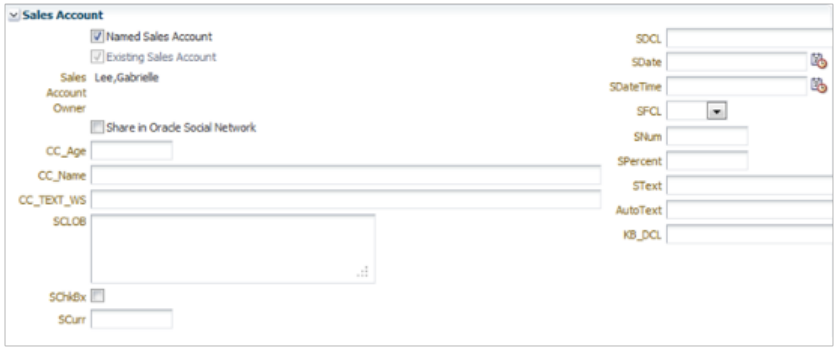
- Create a **Percentage** field.



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.

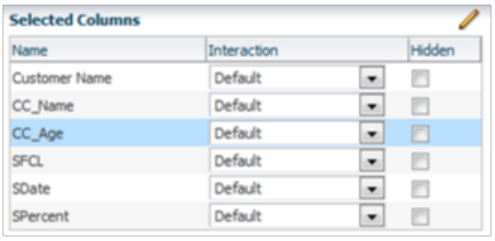
Note

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 a**s 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 the 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 the 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 **RevenueEffectivedate** 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 Sales - CRM Pipeline 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. The Oracle Fusion application then uses 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.

9

Page Composer: Customizing Oracle Fusion CRM Applications

Customizing Oracle Fusion CRM 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 Fusion CRM user interface pages.

In this chapter, you will learn about:

- How to customize standard desktop pages, as well as dashboards
- How to customize simplified pages
- How to use Direct Selection mode to make user interface changes

---

Important

Page Composer supports two editing modes: Design View and Source View. In Oracle Fusion CRM, 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.

Customizing Oracle Fusion CRM Pages Using Page Composer: Explained

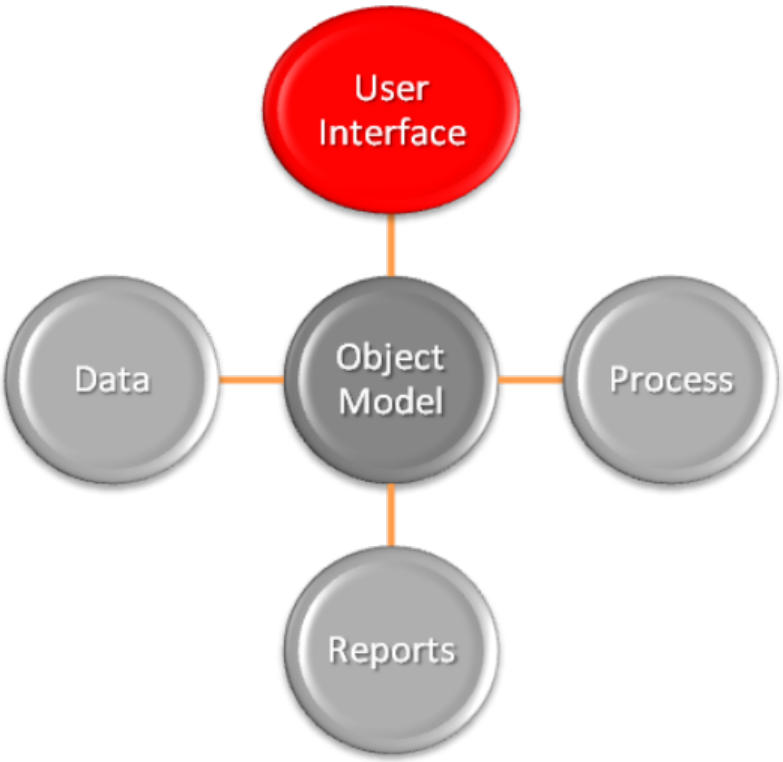
Page Composer is a page editor that you can use to easily edit the user interface composition at run time. In Oracle Fusion CRM, Page Composer is intended for simple user interface editing functions, such as showing and hiding regions, fields, and tables, changing the order of regions, or changing a 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 Fusion CRM 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

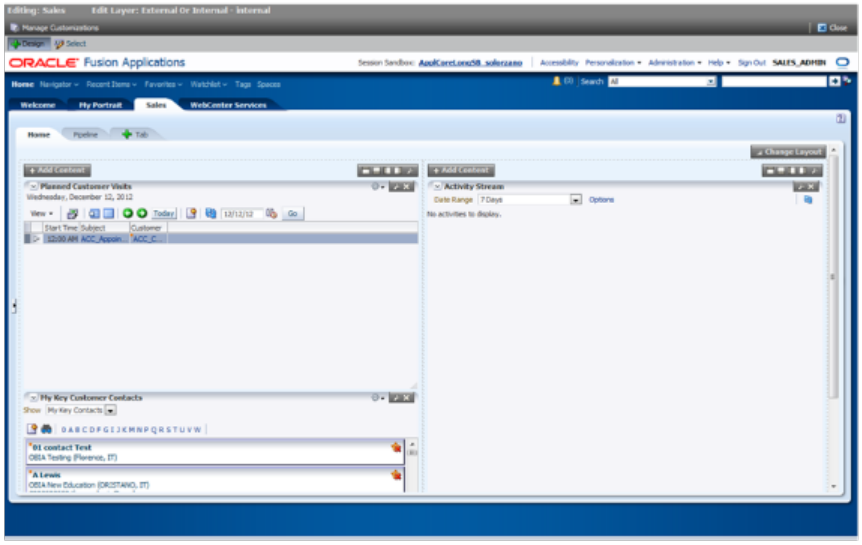
Page Composer supports two editing modes: Design View and Source View. In Design View mode, you can edit by direct manipulation of the target components and their properties. In the Source View mode, you can edit from the programmatic code of the target component. In Oracle Fusion CRM, Page Composer 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 Fusion CRM, 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.

Application	Duty Role	Job Role
CDH	Master Data Management Application Administrator	Master Data Management Application Administrator
Sales Catalog	Sales Catalog Administrator Duty	Sales Catalog Administrator
Sales	Sales Administrator Duty	Sales Administrator
PRM	Channel Administrator Duty	Channel Administrator

PRM	Channel Partner Portal Administrator Duty	Channel Partner Portal Administrator

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 Fusion CRM, 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.

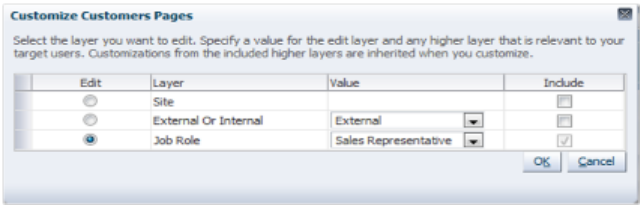
Additionally, you can do Page Composer customizations in any of the following MDS customization layers:

- 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

The Fusion 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 Fusion CRM 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
- Partner Registration: Review and Accept Terms
- Partner Registration 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.

Customization Task	Design View - Standard Mode	Design View - Direct Selection Mode
Change local area layout. For example, change a two-column layout to three-column layout (eight layouts are available).	Yes	No
Add, rename, remove dashboard tabs (except for Home tab).	Yes	No
Expand, collapse dashboard Regional pane (or move the splitter location).	Yes	No
Add and remove panel boxes to or from the dashboard local area.	Yes	No
Add predefined content to dashboard panel boxes (Business Intelligence reports, CRM portlets, and common components such as Calendar).	Yes	No
Edit dashboard panel box properties: show, hide box, reorder child regions, display options, and style.	Yes	No

Add and remove ADF Components to or from dashboards (for example, regions, hyperlinks, images, text boxes, movable boxes, and Web pages) and edit their properties.	Yes	No
Customize saved searches (create and edit).	Yes	No
Hide or show field.	No	Yes
Change field label.	No	Yes
Make field required or not.	No	Yes
Make field read-only or updateable.	No	Yes
Reorder fields in a form.	No	Yes
Reorder table columns.	Yes	Yes
Hide or show table columns.	Yes (end users can optionally display columns at run time)	Yes (end users cannot display columns at run time)
Set table column width with the mouse.	Yes	No
Set table column width and minimum width in percent or pixels.	No	Yes
Enable, disable column sorting.	No	Yes

Customizing the Oracle Fusion CRM Welcome Dashboard

Also known as the Oracle Fusion CRM 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.

Property	Description
Description	Text used by screen readers, for information in addition to what is provided in the Short Desc property.
Label	Display text for the component, for example the field prompt or the single prompt for a group of check boxes

Read only	Whether users can edit the component, for example if a check box can be selected or not
Rendered	Whether the component is visible or hidden to users on the page
Required	Whether users must enter something for the component before saving the page
Short Desc	Text that appears when users hover or focus on the component, for example hover over a field label or click in the text box
Show Required	Whether an asterisk is displayed to indicate that the component is required

When you access component properties on a workstation page using Page Composer, more properties are available.

Save and Label

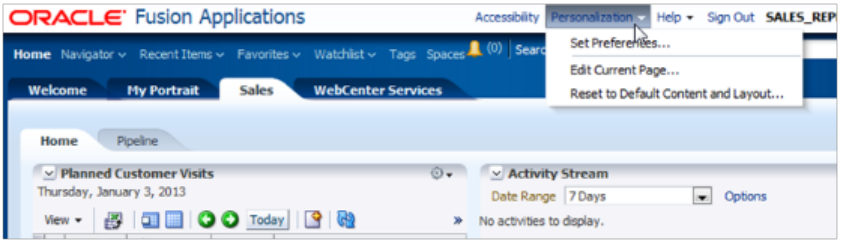
Instead of just saving, optionally save and label your changes so that you can later revert to your saved customizations. Labels are stored with a prefix of **composer_**. For example, if you enter **myLabel**, then the label is **composer_myLabel**.

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 UI 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

ADF Compo Name	Type	Selectab	Rendere Attribu	Read-Only Attribu	Require Attribu	Label Attribu	Minimu Width	Maximu Width	Sortabl	Header Text	Reorder Children
af:active	Field	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA

af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:input	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:outpu	Field	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	NA
af:outpu	Field	Yes	Yes	NA	NA	NA	NA	NA	NA	NA	Yes
af:panel	Field	Yes	Yes	NA	NA	Yes	NA	NA	NA	NA	NA
af:panel	Contain	Yes	No	NA	NA	NA	NA	NA	NA	NA	Yes
af:richT	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:select	Field	Yes	Yes	Yes	Yes	Yes	NA	NA	NA	NA	NA
af:Colu	Contain	Yes	Yes	NA	Yes	NA	Yes	Yes	Yes	Yes	NA
af:panel	Contain		Yes	NA	NA	NA					NA
af:show	Contain		Yes	NA	NA	NA					NA
af:table	Contain	Yes	No	NA	NA	NA	NA	NA	NA	NA	Yes
fnd:app table	Contain		No	NA	NA	NA					NA
af:panel	Contain		No	NA	NA	NA					Yes

10

Oracle Fusion Common CRM Extensibility

Oracle Fusion Common CRM Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Fusion Common CRM.

In this chapter, you will learn about:

- Each Common CRM object that is part of the Trading Community Foundation, and how to extend those objects using Oracle Fusion CRM Application Composer.

Oracle Fusion Common CRM Objects: Explained

You can customize a variety of pages and regions throughout Oracle Fusion CRM using the Oracle Fusion CRM Application Composer. The Application Composer lets you create custom fields and objects for a specific CRM 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 Oracle Fusion Marketing or Oracle Fusion Sales. Oracle Fusion Common CRM objects, however, are unique in that the customizations you make to these objects can be displayed in a variety of CRM applications, and in a variety of pages and regions, at run time. This topic provides an overview of each Common CRM object that is part of the Trading Community Foundation. (This topic does not address the common component objects which are also listed within the Common CRM application container: the interaction, note, and task objects.)

The Trading Community Foundation Common CRM standard objects that you can extend are:

- Trading Community Address
- Trading Community Customer Contact Profile
- Trading Community Group Profile
- Trading Community Organization Profile
- Trading Community Person Profile
- Trading Community Relationship
- Trading Community Resource Profile

Note

Some of these objects are also documented in the related topic, Customizing Oracle Fusion Customer Center Pages: Explained.

Trading Community Address

The Trading Community Address object is an intersection object that stores a reference to a physical location/address, as well as a reference to a trading community organization, person, or group. Any extensions to this object are available for display in customer, contact, partner, and resource address user interfaces within Oracle Fusion CRM applications.

To display your extensions, such as custom fields, first create your custom fields using the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Address object.

This table indicates, for the Trading Community Address object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Address Creation Region	• Edit Customer - Address Create region in Customer Center • Edit Contact - Address Create region in Customer Center • Edit Partner Profile - Address Create region in Partner Center • Edit Partner - Partner Members - Address Create region within Partner Center
Edit Address Summary Table	• Edit Customer - Addresses region in Customer Center • Edit Contact - Addresses region in Customer Center • Edit Partner Profile - Address region in Partner Center • Edit Partner - Partner members - Address region within Partner Center

Edit Person Quick Creation Region	• Consumer Create region within Customer Center • Contact Create region within Customer Center • Person Create region within Customer Hub
Edit Organization Quick Creation Region	• Customer Create region within Customer Center • Organization Create region within Customer Hub

Note

Oracle Fusion CRM 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.

Trading Community Customer Contact Profile

The Trading Community Customer Contact Profile object stores all intrinsic information for all customer contact persons, in the context of the contact's relationship with a customer. For extensions specific to a customer contact relationship, use this object. By contrast, a standalone contact in Oracle Fusion CRM 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 Trading Community Person Profile object.

Any extensions to this object are available for display in various pages within Oracle Fusion CRM applications. Generic person extensions are also visible from the Oracle Fusion Customer Hub application in the Person work area.

To display your extensions, such as custom fields, first create your custom fields using the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Customer Contact Profile object.

This table indicates, for the Trading Community Customer Contact Profile object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Person Quick Creation Region	• Consumer Create region within Customer Center • Contact Create region within Customer Center

Edit Contact Summary Table	• Edit Customer - Contacts table in Customer Center • Edit Partner - Members table in Partner Center
Edit Contact Read-Only Region	• Edit Contact - Basic Information region in Customer Center
Edit Contact Details Region	• Edit Customer: Contact Details within Customer Center • Edit Organization: Relationships within Customer Hub • Edit Partner - Partner Members - Person Details region within Partner Center

Trading Community Group Profile

The Trading Community Group Profile object stores all intrinsic information for all groups within the trading community of the deploying company. In Oracle Fusion CRM, groups could be customers of the deploying company. The Trading Community Group Profile 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 Group Profile 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 groups within your trading community.

In Oracle Fusion CRM, 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 Fusion Customer Hub application 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 the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Group Profile object.

This table indicates, for the Trading Community Group Profile object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Summary Table	• Oracle Fusion Customer Hub application in the Groups work area
Edit Group Update Form	• Oracle Fusion Customer Hub application in the Groups work area

Edit Group Create Form	• Oracle Fusion Customer Hub application in the Groups work area

Trading Community Organization Profile

The Trading Community Organization Profile object stores all intrinsic information for all organizations within the trading community of the deploying company. Examples of organizations from a CRM perspective 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 Trading Community Organization Profile 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 Trading Community Organization Profile object. The Organization Profile 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 organizations within your trading community.

Any extensions to this object are available for display in various pages within Oracle Fusion CRM applications. Generic organization extensions are also visible from the Oracle Fusion Customer Hub application in the Organization work area.

To display your extensions, such as custom fields, first create your custom fields using the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Organization Profile object.

This table indicates, for the Trading Community Organization Profile object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Organization Details Region	• Customer Details region within Customer Center • Organization Details region within Customer Hub • Competitor Summary region within Sales
Edit Organization Quick Creation Region	• Customer Create region within Customer Center • Organization Create region within Customer Hub

Edit Organization Read-Only Region	• Basic Information Region within Customer Center • Basic Organization Information Region within Customer Hub

Trading Community Person Profile

The Trading Community Person Profile object stores all intrinsic information for all persons within the trading community of the deploying company. Examples of persons from a CRM perspective are customers (also called consumers), contacts, internal employee resources, and partner members. To extend the customer contact relationship, use the Trading Community Customer Contact Profile object. To extend internal resources, use the Trading Community Resource Profile object. To extend person customers (also called consumers), use the Sales Account object (under the Customer Center application container). (The Trading Community Person Profile 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 Person Profile 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 persons within your trading community.

Any extensions to this object are available for display in various pages within Oracle Fusion CRM applications. Generic person extensions are also visible from the Oracle Fusion Customer Hub application in the Person work area.

To display your extensions, such as custom fields, first create your custom fields using the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Person Profile object.

This table indicates, for the Trading Community Person Profile object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Person Read-Only Region	• Basic Information region within Customer Center (for Consumer or Person customer) • Contact Basic Information within Customer Center • Basic Person Information region within Customer Hub
Edit Person Details Region	• Customer Details region within Customer Center (for Consumer or Person customer) • Manage Contacts: Contact Details within Customer Center

Edit Person Quick Creation Region	• Consumer Create region within Customer Center • Contact Create region within Customer Center • Person Create region within Customer Hub

Trading Community Relationship

The Trading Community 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 Trading Community Customer Contact Profile object for customer contact relationship-specific extensions. Use this Trading Community 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 Fusion CRM applications. Generic person extensions are also visible from the Oracle Fusion Customer Hub application in the Person work area.

To display your extensions, such as custom fields, first create your custom fields using the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Relationship object.

This table indicates, for the Trading Community Relationship object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Relationship Summary Table	• Edit Organization: Relationships within Customer Hub • Edit Person: Relationships within Customer Hub • Edit Group: Relationships within Customer Hub • Create Group: Group Members within Customer Hub
Edit Relationship Creation Region	• Edit Organization: Relationships within Customer Hub • Edit Person: Relationships within Customer Hub • Edit Group: Relationships within Customer Hub • Create Group Members within Customer Hub

Trading Community Resource Profile

The Trading Community Resource Profile object defines users of the Oracle Fusion CRM application. 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 the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the pages and regions listed below. You access the configuration pages in the Application Composer from the Pages node under the Trading Community Resource object.

This table indicates, for the Trading Community Resource object, which Application Composer configuration page lets you populate which page or region with your application customizations.

Configuration Page in Application Composer	Related Page or Region
Edit Details Region	• 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.
Edit Search Region for the Workarea Summary Table	• The search region is available in the Resource Directory work area, under Search Resources.

Extending Simplified Pages for Contacts: Explained

Using Oracle Fusion CRM 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 Trading Community Person Profile 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

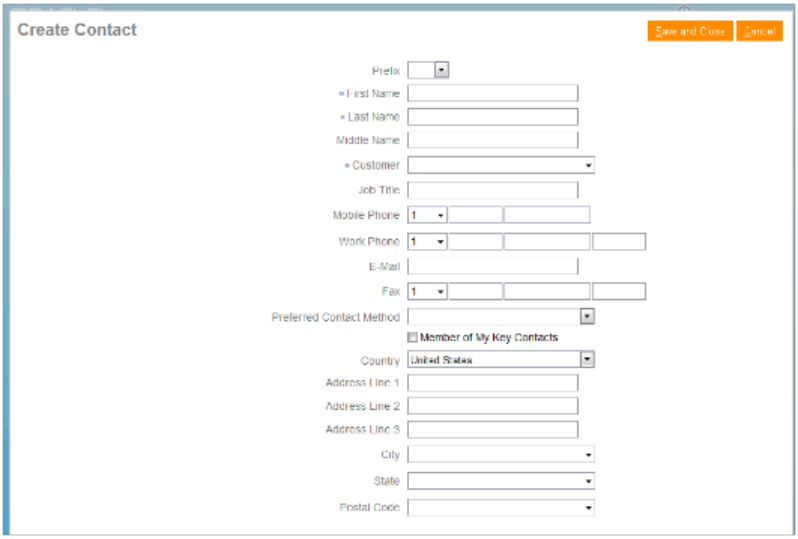
- Contact Overview page

Create Contact Page

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

- Hide or show existing fields that belong to the Trading Community Person Profile object.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).

  Custom fields that you add to the Trading Community Person Profile 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)

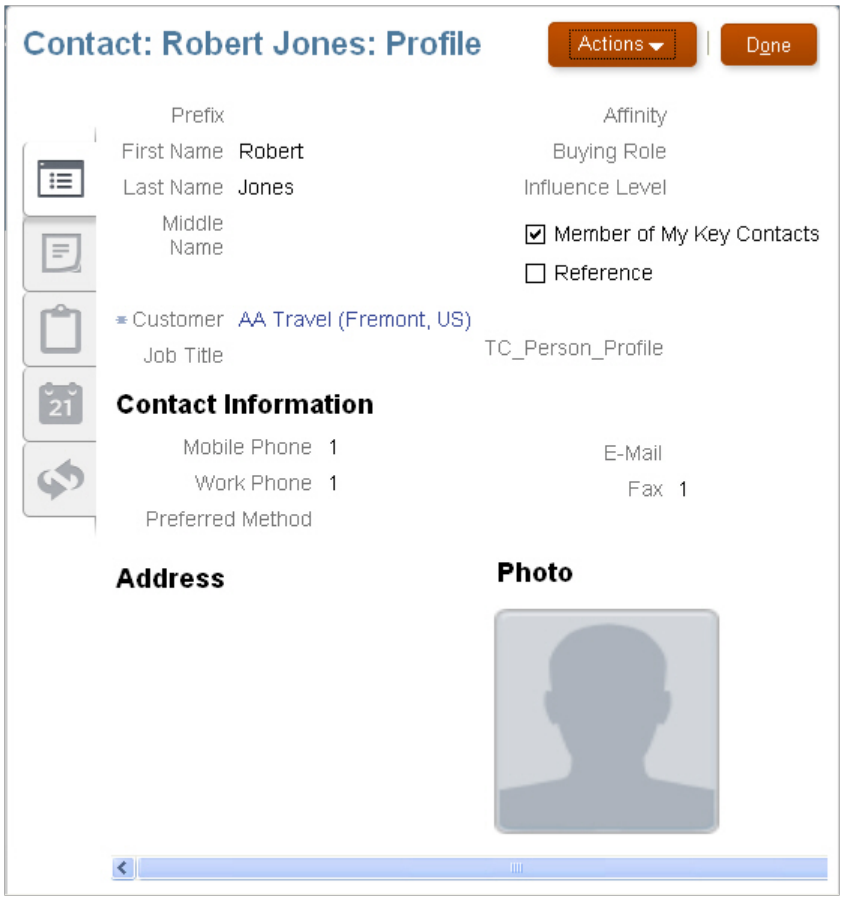
You can extend the following items on the Contact detail page (Profile page):

- Hide or show existing fields that belong to the Trading Community Person Profile object.
- Reorder fields.
- Change display labels.
- Add custom fields (all types).

  Custom fields that you add to the Trading Community Person Profile 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.

11

Oracle Fusion Customer Center Extensibility

Oracle Fusion Customer Center Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Fusion Customer Center.

In this chapter, you will learn about:

- Each Customer Center object, and how to extend those objects using Oracle Fusion CRM Application Composer
- Which Oracle Fusion Common CRM objects are associated with Customer Center pages
- How to customize Customer Center pages using Page Composer

Customizing Oracle Fusion Customer Center Pages: Explained

You can customize a variety of pages and regions in Oracle Fusion Customer Center using the Oracle Fusion CRM 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 category.

Customizing Oracle Fusion Customer Center Pages Using Application Composer

In general, every top-level CRM 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.

Customer Center Page	Customer Center Region	Application	Underlying Business Object
Consumer Profile	Addresses region	Common	Trading Community Address
Consumer Profile	Consumer Basic Information region	Common	Trading Community Person Profile
Consumer Profile	Consumer Details region	Common	Trading Community Person Profile
Consumer Profile	Sales Account region	Customer Center	Sales Account
Contact Profile	Address region	Common	Trading Community Address
Contact Profile	Basic Information region	Common	Trading Community Org Contact
Contact Profile	Contact Details region	Common	Trading Community Org Contact
Contact Profile	Contacts region (also known as the Contacts List)	Common	Trading Community Org Contact
Create Consumer page and also the Quick Create Consumer page	New fields are added at the bottom of the page	Customer Center	Sales Account
Create Consumer page and also the Quick Create Consumer page	No specific region	Common	Trading Community Person Profile
Create Contact page and also the Quick Create Contact page	No specific region	Common	Trading Community Org Contact
Create Customer page and also the Quick Create Customer page	Contact Information region	Common	Trading Community Org Contact
Create Customer page and also the Quick Create Customer page	Customer Information region	Common	Trading Community Org Profile
Create Customer page and also the Quick Create Customer page	New fields are added at the bottom of the page.	Customer Center	Sales Account
Customer Profile	Addresses region	Common	Trading Community Address
Customer Profile	Basic Information region	Common	Trading Community Org Profile
Customer Profile	Customer Details region	Common	Trading Community Org Profile
Customer Profile	Sales Account region	Customer Center	Sales Account

Edit Customer page	Team Members region	Customer Center	Sales Account Resource (child object to the Sales Account)
Overview > Summary tab	Sales Accounts region (also known as the Sales Account List)	Customer Center	Sales Account
Real-Time Search: Customers page	Search region, by way of Advanced Search > Add Fields	Customer Center	Sales Account

Note

To make only minor user interface changes to Customer Center pages without creating objects or fields, use Page Composer instead of Application Composer.

Oracle Fusion Common CRM Objects

The Oracle Fusion Common CRM objects that are associated with Customer Center pages are:

- Trading Community Org Profile
- Trading Community Person Profile
- Trading Community Address
- Trading Community Org 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.

Business Object	Configuration Page in Application Composer	Related Customer Center Page	Related Customer Center Region
Trading Community Org Profile	Edit Customer Quick Creation Form	Create Customer page and also the Quick Create Customer page	Customer Information region
Trading Community Org Profile	Edit Read Only Form	Customer Profile	Basic Information region
Trading Community Org Profile	Edit Details Form	Customer Profile	Customer Details region
Trading Community Person Profile	Edit Contact/Consumer Quick Creation Form	Create Consumer page and also the Quick Create Consumer page	No specific region
Trading Community Person Profile	Edit Read Only Form	Consumer Profile	Consumer Basic Information region
Trading Community Person Profile	Edit Details Form	Consumer Profile	Consumer Details region
Trading Community Address	Edit Detail Form	Customer Profile	Addresses region
Trading Community Address	Edit Detail Form	Contact Profile	Addresses region
Trading Community Address	Edit Detail Form	Consumer Profile	Addresses region

Trading Community Org Contact	Edit Creation Form	Create Customer page and also the Quick Create Customer page	Contact Information region
Trading Community Org Contact	Edit Creation Form	Create Contact page and also the Quick Create Contact page	No specific region
Trading Community Org Contact	Edit Read Only Form	Contact Profile	Basic Information region
Trading Community Org Contact	Edit Contact Details Form	Contact Profile	Contact Details region
Trading Community Org Contact	Edit Summary Table	Contact Profile	Contacts region (also known as the Contacts List)

Oracle Fusion Customer Center Objects

The Oracle Fusion 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.

Business Object	Configuration Page in Application Composer	Related Customer Center Page	Related Customer Center Region
Sales Account	Edit Creation Form	Create Customer page and also the Quick Create Customer page	New fields are added at the bottom of the page.
Sales Account	Edit Creation Form	Create Consumer page and also the Quick Create Consumer page	New fields are added at the bottom of the page.
Sales Account	Edit Details Form	Customer Profile	Sales Account region
Sales Account	Edit Details Form	Consumer Profile	Sales Account region
Sales Account	Edit Summary Table	Overview, then Summary tab	Sales Accounts region (also known as the Sales Account List)
Sales Account	Not applicable. Custom fields are automatically available from the list of additional fields.	Real-Time Search: Customers page	Search region, by way of Advanced Search, then Add Fields
Sales Account	Edit Regional Panes	Customer Work Area	Panes in the regional area
Sales Account Resource (child object to the Sales Account)	Edit Summary Table	Edit Customer: Sales Account Team page	Team Members region

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.

Customizing Oracle Fusion Customer Center Using Page Composer: Explained

To customize the Customers work area and Oracle Fusion 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 Fusion Customer Center Using Page Composer: Worked Example

This example demonstrates a set of use cases for customizing Oracle Fusion 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**.

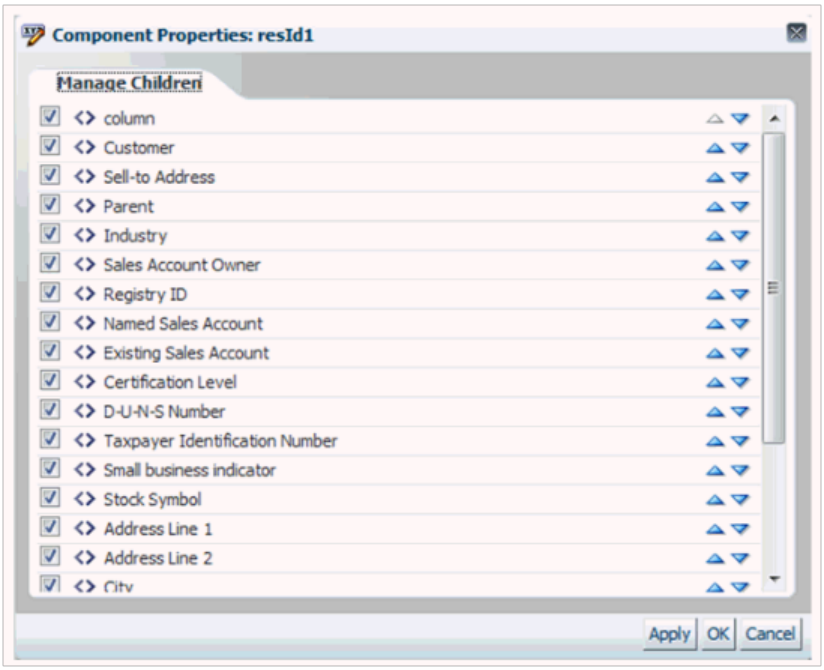
   The page opens in the Page Composer Design mode.
6. Click **Select**to enter the Direct Selection mode.
7. Mouse-over the Registry ID column header in the Sales Account Search Results table and click the column header.

   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.

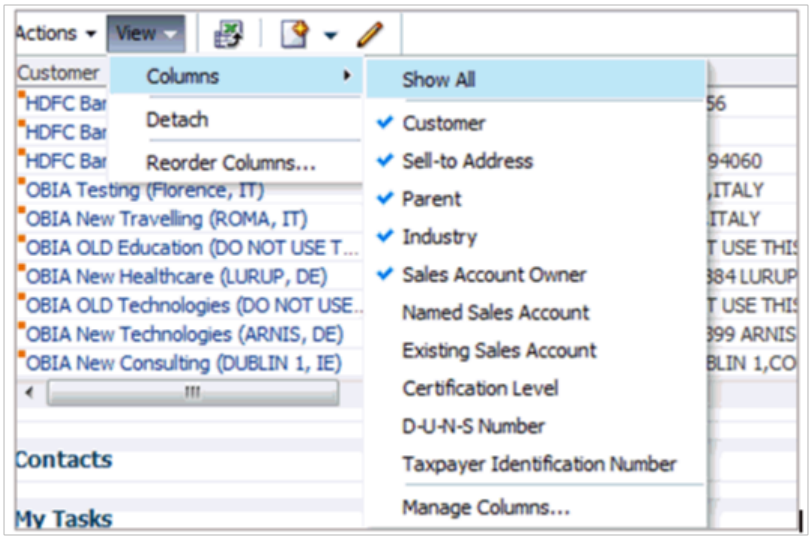
   The item is removed from the table View menu and from the table.

   The following figure shows the Component Properties page.



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.



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.

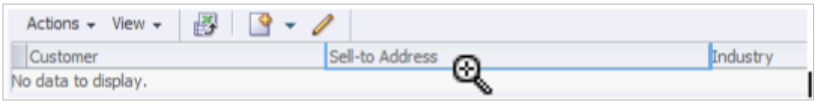
3. Click the Administration global menu.
4. In the Customize Customers Pages dialog box, select the **Site** layer.
5. Click **OK**.

   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.

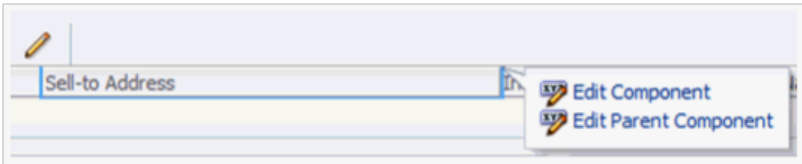
   The column header is automatically outlined with a blue border.

   The following figure shows the Sell-to Address link.



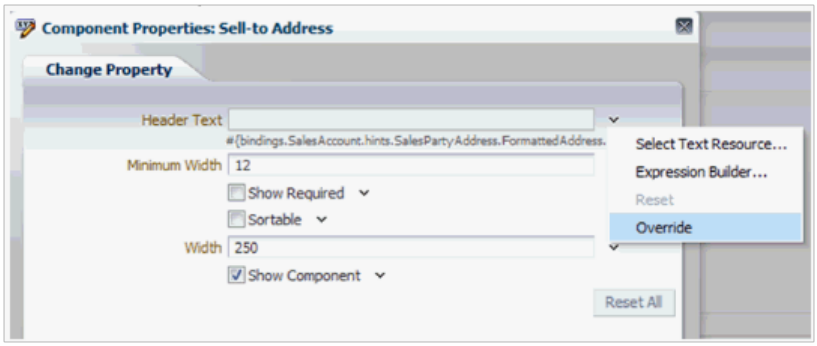
8. Click **Edit Component**.

   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.



    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.

   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.

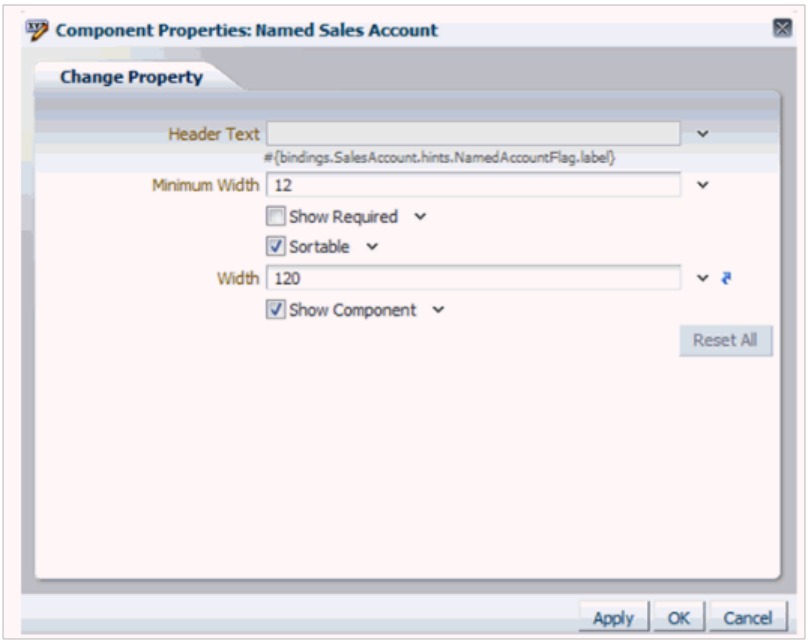
   ---

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 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.

   The column header is outlined automatically with a blue border.
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.

3. Search for any customer record in the Sales Account Search region.

4. In the search results table, drill into the customer record.

   The Edit Customer <Customer Name> Profile page appears by default.

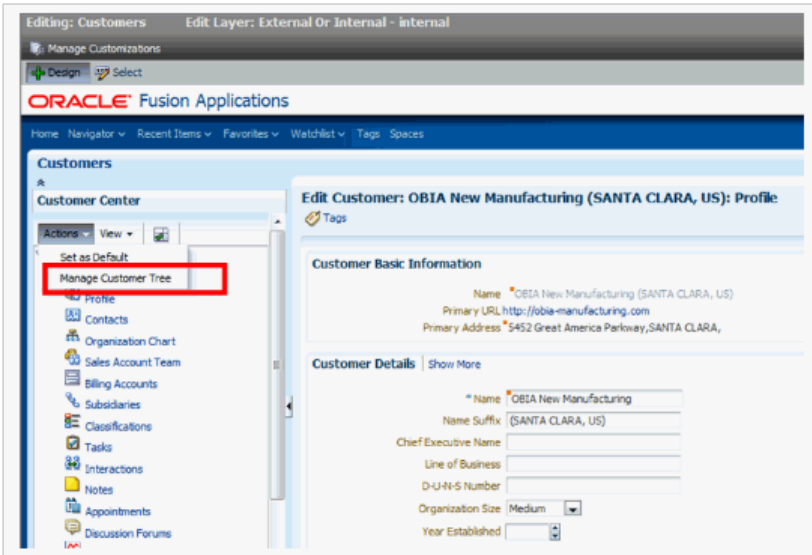
   In the following figure the Profile node is selected and highlighted in the Customer Center Tree.

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.



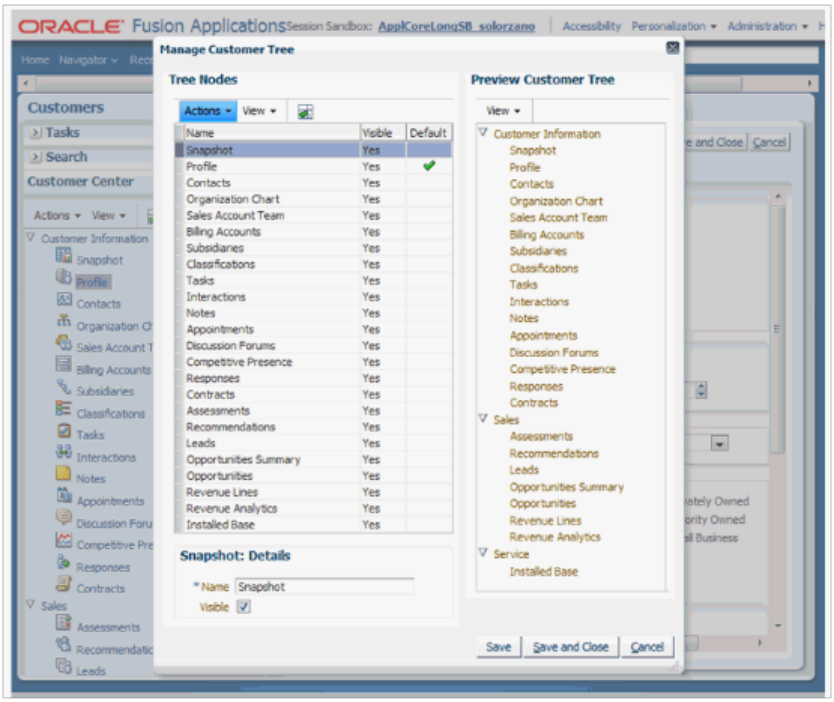
The Manage Customer Tree dialog box appears.

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.

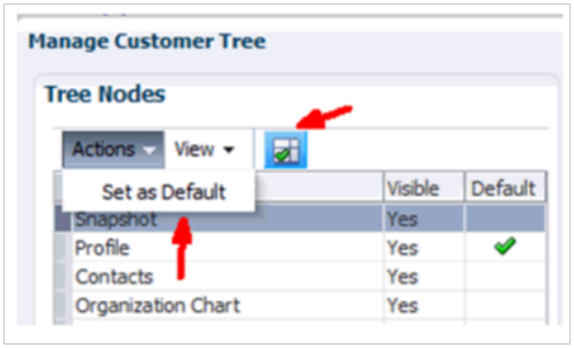
The following figure shows the Manage Customer Tree Dialog box.



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.

e. 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 Oracle Fusion CRM 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 Trading Community Person Profile and Trading Community Organization Profile 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.

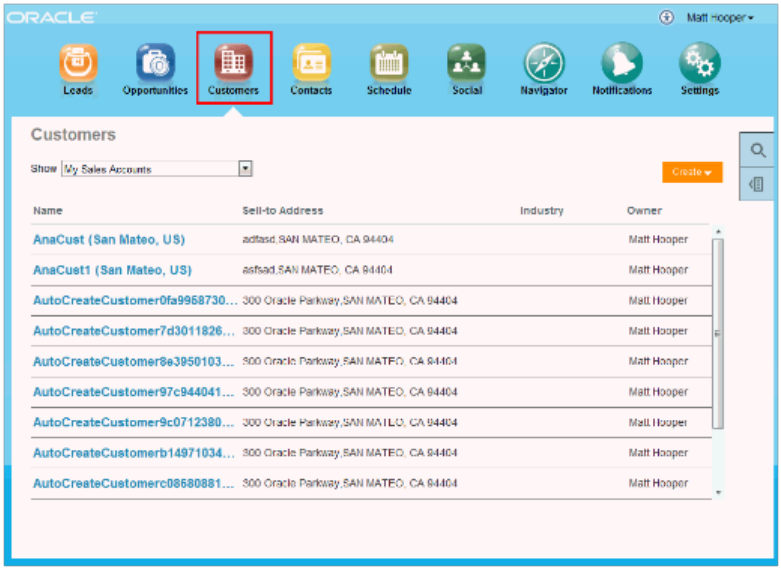
Note that these pages are not extensible:

- 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.



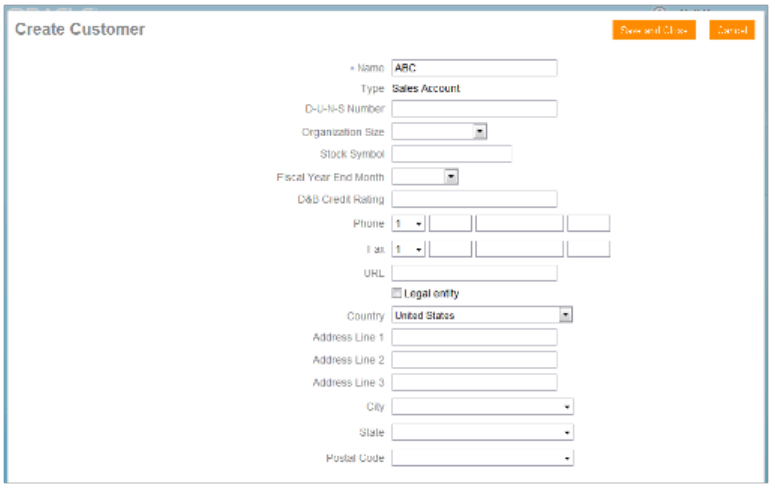
Note that you cannot make changes to these components on the page:

- 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 Trading Community Organization Profile and Sales Account objects.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
  - Custom fields that you add to the Trading Community Organization Profile 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 Trading Community Organization Profile object in Application Composer to extend this region.

- Sales Account region

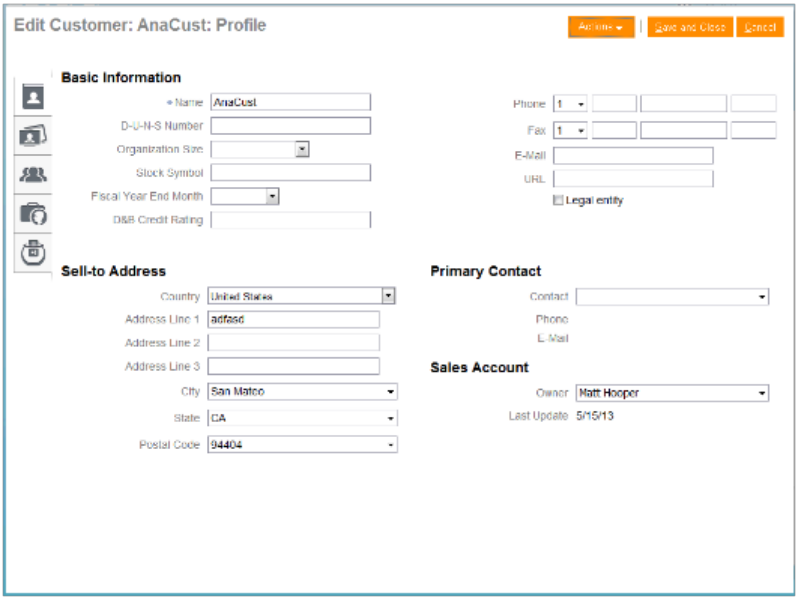
  Use the Sales Account object in Application Composer to extend this region.

- 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:

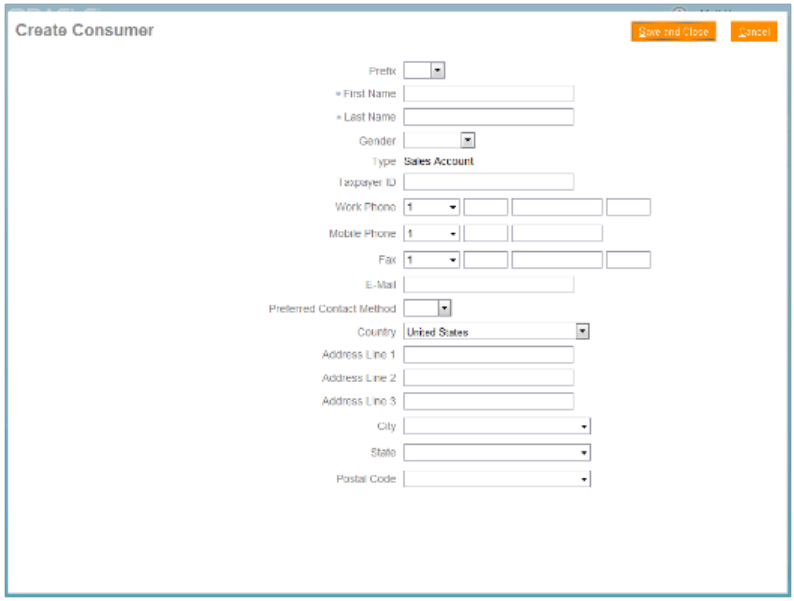
- Subtab regions
  - Contacts
  - Opportunities
  - Leads
- Actions button
- Custom buttons and actions

- 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 Trading Community Person Profile and Sales Account objects.
- Reorder fields.
- Change field labels.
- Add custom fields (all types).
  - Custom fields that you add to the Trading Community Person Profile 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 Trading Community Person Profile 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.

12

Oracle Fusion Sales and Oracle Fusion Opportunity Management Extensibility

Oracle Fusion Sales and Oracle Fusion Opportunity Management Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Fusion Sales and Oracle Fusion Opportunity Management.

In this chapter, you will learn about:

- Each Oracle Fusion Sales and Oracle Fusion Opportunity Management object, and how to extend those objects using Oracle Fusion CRM Application Composer
- How to customize Oracle Fusion Sales and Oracle Fusion Opportunity Management pages using Page Composer

Customizing Sales Pages Using Application Composer: Explained

You can customize a variety of regions in Oracle Fusion Sales, including Sales Competitor and Sales Reference Customer regions, using the Oracle Fusion CRM Application Composer. The Application Composer lets you create custom fields and objects, which you then add for display in the runtime Sales Competitor and Sales Reference Customer applications. To access the Application Composer, select **Application Composer** from the Navigator menu, under the Tools category.

Understanding Which Oracle Fusion 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 the Application Composer to customize those pages.

The Oracle Fusion 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 the 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 the Application Composer and create your custom fields.

Sales Page	Sales Region	Application	Underlying Business Object
Competitors Overview page	Competitors Summary (List View)	Sales	Sales Competitor
Create Competitor page	Default details region	Sales	Sales Competitor
Edit Competitor page	Default details region	Sales	Sales Competitor
Sales References Overview page	References Summary (list view)	Sales	Sales Reference Customer
Edit Sales Reference page	Default details region	Sales	Sales Reference Customer

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 the 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 the Application Composer. Next, use the 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 the 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 select the Pages node for the object.

The following table indicates which 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.

Business Object	Configuration Page in Application Composer	Related Sales Page	Related Sales Region
Sales Competitor	Competitors Summary (list view) in Overview page: Select Sales Competitor object, select Pages link, and then select the Edit Summary Table link.	Competitors Overview page	Competitors Summary (list view) region

Sales Competitor	Create Competitor page: Select Sales Competitor object, select Pages link, and then select the Edit Creation Page link.	Create Competitor page	Create Competitor page
Sales Competitor	Edit Competitor page, default details: Select Sales Competitor object, select Pages link, and then select the Edit Summary Form link.	Edit Competitor page	Edit Competitor page
Sales Reference Customer	References summary (list view) in Overview page: Select Sales Reference Customer object, select Pages link, and then select the Edit Summary Table link.	References Overview page	References summary (list view)
Sales Reference Customer	Edit Reference page: Select Sales Reference Customer object, select Pages link, and then select the Edit Summary Form link.	Edit Reference page	Edit Reference page

Customizing Opportunity Pages Using Application Composer: Explained

You can customize a variety of regions in Oracle Fusion Opportunity Management using Oracle Fusion CRM Application Composer. The Application Composer lets you modify pages and create custom fields and objects, which you then make available in the run time Opportunity Management application. To access the Application Composer, select **Application Composer** from the Navigator menu, under the Tools category. The opportunity areas that can be customized are listed under **Sales** in the **Application** list of values.

Note that in the Sales area in the Application Composer screens, you can also make changes to the Oracle Fusion 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 Oracle Fusion Opportunity Management Pages Are Extensible

To customize Opportunity Management 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 Opportunity Management 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.

Sales Page	Sales Region	Application	Underlying Business Object
Opportunity Overview page	Opportunity summary (list view) table	Opportunity Management	Opportunity
Create Opportunity page	Create Opportunity page	Opportunity Management	Opportunity
Edit Opportunity page	Summary region (above the Detail region available by selecting the Show More option)	Opportunity Management	Opportunity
Edit Opportunity page	Detail region (expand the Show More option)	Opportunity Management	Opportunity
Edit Opportunity page	Revenue table	Opportunity Management	Opportunity
Edit Opportunity page	Revenue table detail stamp (the region uncovered when you expand the revenue item row)	Opportunity Management	Opportunity
Edit Opportunity page	Contacts summary table	Opportunity Management	Opportunity Contact (child object of opportunity)
Edit Opportunity page	Contacts summary table actions	Opportunity Management	Opportunity Contact (child object of opportunity)
Edit Opportunity page	Opportunity Team summary table	Opportunity Management	Opportunity Resource (child object of opportunity)
Edit Opportunity page	Opportunity Team summary table detail stamp (the region uncovered when you expand the team member row)	Opportunity Management	Opportunity Resource (child object of opportunity)

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 Management 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 Management 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 Management 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 Management 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 Management pages and regions, as well as the Application Composer configuration pages where you can make user interface changes on those pages and regions.

Business Object	Configuration Page in Application Composer	Related Opportunity Management Page	Related Opportunity Management Region
Opportunity	Opportunity summary (list view) in Overview page: Select Opportunity object, select Pages link, and then select the Edit Summary Table link	Overview page	Opportunity summary (list view)
Opportunity	Create Opportunity page: Select Opportunity object, select Pages link, and then select the Edit Creation Page link.	Create Opportunity page	Create Opportunity page
Opportunity	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.	Edit Opportunity page	Summary region, Default area
Opportunity	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.	Edit Opportunity page	Summary region, Detailed area
Opportunity	Edit Opportunity page, Revenue Items table: Select Opportunity object, select Pages link, and then select the Edit Revenue Table link.	Edit Opportunity page	Revenue Items table

Opportunity	Edit Opportunity page, Revenue table, detail stamp: Select Opportunity object, select Pages link, and then select the Edit Revenue Table link.	Edit Opportunity page	Revenue Items table, detail stamp
Opportunity	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.	Edit Opportunity page	Contacts tab, contacts summary (list view) table
Opportunity	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.	Edit Opportunity page	Contacts tab, contacts summary (list view) table actions
Opportunity	Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Team in the table and edit it using Actions menu.	Edit Opportunity page	Opportunity team summary (list view) table
Opportunity	Select Opportunity object, select Pages link, then under Opportunity Detail Tab, select Opportunity Team in the table and edit it using Actions menu.	Edit Opportunity page	Opportunity team summary (list view) table, detail stamp
Opportunity Contact	Opportunity Contact detail view fields: Select Opportunity Contact > Fields: Select Opportunity Contact object and then select Fields link.	Edit Opportunity page	Contact detail view (fields only)
Opportunity Revenue	Opportunity Revenue Items summary (list view) table columns: Select Opportunity Revenue object, select Pages link, and then select Edit Summary Table link.	Edit Opportunity page	Opportunity Revenue Items summary (list view) table

Opportunity Team Member	Opportunity Team member detail view fields: Select Opportunity Team Member object and then select Fields link.	Edit Opportunity page	Opportunity team member detail view (fields only)

Customizing the Dashboard and Pages in Oracle Fusion Sales Using Page Composer: Explained

The Oracle Fusion 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 Oracle Fusion Sales Pages

To customize Oracle Fusion 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 Oracle 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. Select the **Administration** menu option in the global region.
5. Select the **Customize Opportunities Pages** menu option.

   The Customize Opportunities Pages dialog appears.
6. In the Customize Opportunities Pages dialog box, check the **Edit** option for the Job Role layer.
7. In the Value column next to Job Role, select Sales Representative from the drop list.

8. 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".

9. 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.

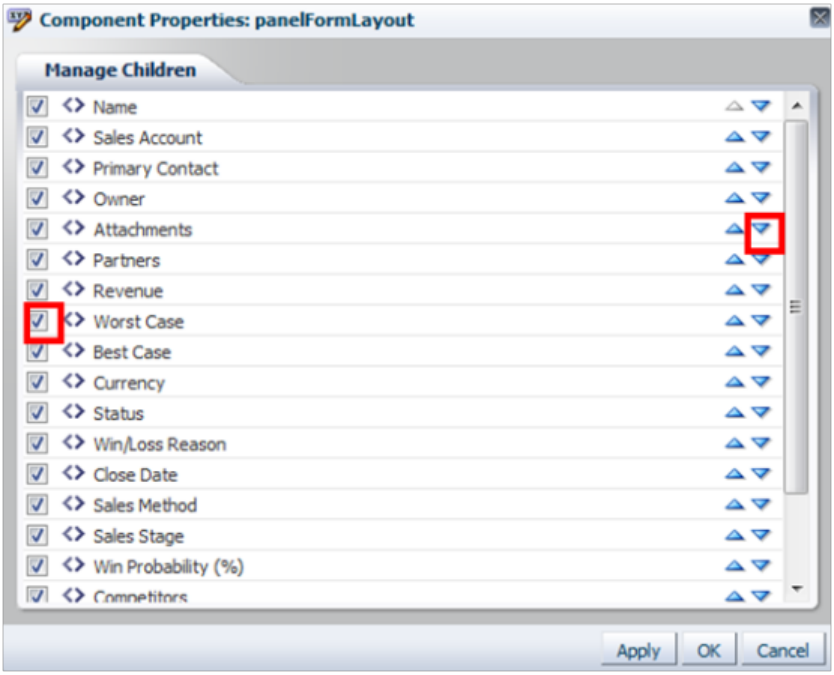
   A dialog box opens with two options: Edit Component and Edit Parent Component.

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.

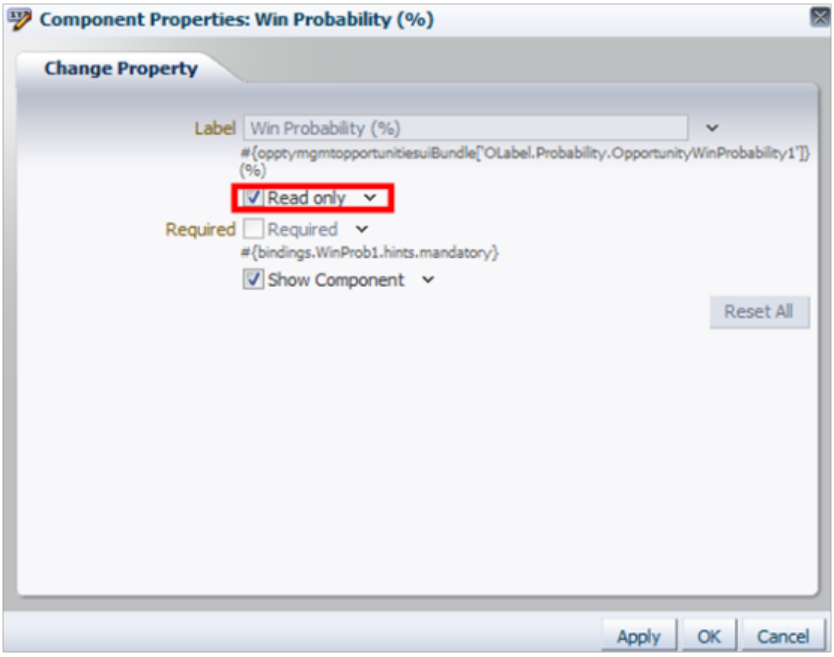
   A dialog box opens with two options: Edit Component and Edit Parent Component.

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

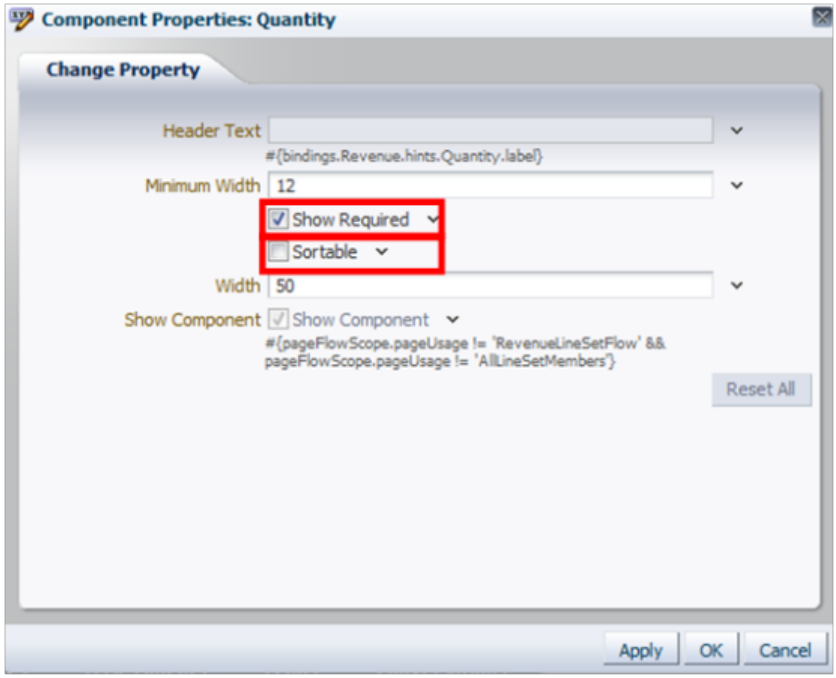
To make the Quantity column in the Revenue Items table required and not able to be sorted, use the following steps:

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. Select the Administration global menu.
3. Select Manage Sandboxes.
4. Select the row of the sandbox where you built your customization.
5. 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 Oracle Fusion Applications.
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 Oracle Page Composer, you can add a new Quota subtab or page to the Sales dashboard, plus add two Oracle Fusion Quota Management reports to the page. The purpose of the task is to create, for all salespeople using the instance, a unique tab/page in the Sales dashboard with Quota Management reports.

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 Mode

You must 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 **Sales Dashboard**. Or, select the **Sales** tab on the landing page.
3. Select the **Administration** menu option in the global region.
4. Select the **Customize Opportunities Pages** menu option.

   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: Sales" and "Edit Layer: Job Role".

Adding a New Subtab to the 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 Quota name.

Adding Reports to the 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.

Changing the Layout of Dashboard

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.

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. Select the **Administration** global menu.
3. Select **Manage Sandboxes**.
4. Select the row of the sandbox where you built your customization.
5. Select the **Publish** button to commit your changes.

Verifying Your 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 Page Composer editor.
2. Sign out of Oracle Applications.
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 by Job Role Using Page Composer: Worked Example

This example demonstrates how, using Oracle Page Composer, you can create a custom opportunity saved search for a specific job role.

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.
- Set the profile option, Require Additional Criteria for Opportunity Search Enabled (MOO_SRCH_ADDL_CRITERIA_REQD), according to your business requirements. Set it to Yes to require additional criteria in the search (for example, Opportunity Name or Opportunity Number) other than the required fields.

Invoking Page Composer Edit Mode

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. Select the **Administration** menu option in the global region.
4. Select the **Customize Opportunities Pages** menu option.

   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: Sales" and "Edit Layer: Job Role".

Creating and Running the Search

Next, create and run the custom search.

Before beginning this portion of the example, consider the following concepts:

- Sales opportunities (and other CRM objects like leads) have the concept of record sets.
- Record sets represent your data set, or the data you have access to.
- A record represents a single opportunity and a record set represents multiple opportunities.
- Opportunities have inherent data security on them that depends on your membership in the sales team and territory hierarchy.

In this example, you are creating a saved search based on:

- Record sets where the salesperson is on the sales team;
- Opportunity close dates that fall within the current and next quarters of the calendar; and
- Opportunities in open status.

1. With Page Composer edit mode invoked, open the Search panel in the Opportunities overview page/workarea.
2. Set up the search criteria for the new saved search, using the values in the following table:

Criteria	Value
Record Set	Records where I am on the team
Close Period	Equals: Current and Next Quarter
Status	Equals: Open

3. Run the search: Select the Search button.
4. Save the search: Select the Save button.
5. In the Create Saved Search dialog box, enter a meaningful name in the Name field. For example, enter All Opportunities in My Team.

6. 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 Opportunities work area.
7. Select 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. Select the Administration global menu.
3. Select Manage Sandboxes.
4. Select the row of the sandbox where you built your customization.
5. Select the Publish button to commit your changes.

Verifying Your 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 Page Composer editor.
2. Sign out of Oracle Applications.
3. Sign in as a user with the Sales Representative role.
4. Navigate to the Opportunities work area.
5. Confirm that the saved search you created is the default saved search and that it loads automatically.

Extending Simplified Pages for Opportunities: Explained

Using Oracle Fusion CRM 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.

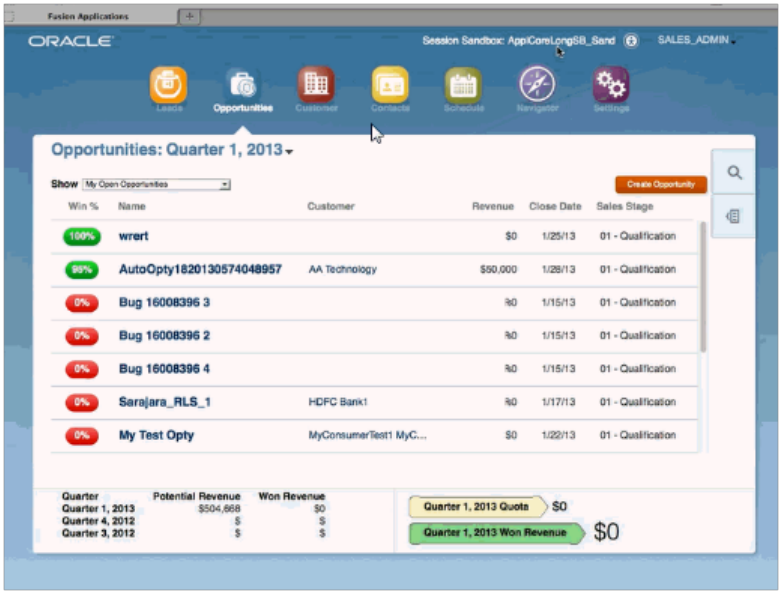
- Opportunity overview table
- Create Opportunity page
- Edit Opportunity page

Opportunity Overview Table

You can extend the following items in the Opportunity overview table:

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

This figure shows the Opportunities overview 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

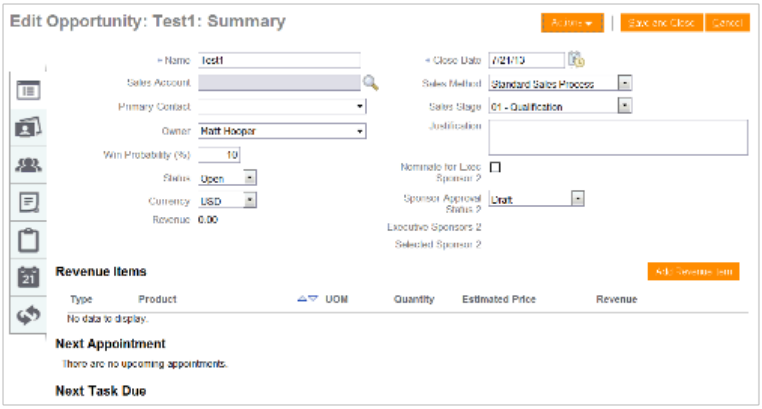
  In Application Composer, add custom fields to the Opportunity Contact object. Then, navigate to the Pages node for the Opportunity object, and edit the Contacts subtab in the Opportunity Detail Tab region.
- Opportunity Team subtab

  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.

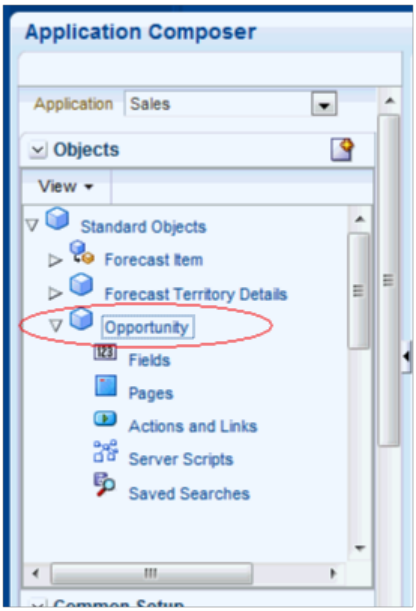
Extending Simplified Pages for Opportunities Using Application Composer: Worked Example

This example demonstrates how you can customize simplified pages for Opportunities using Oracle Fusion CRM Application Composer:

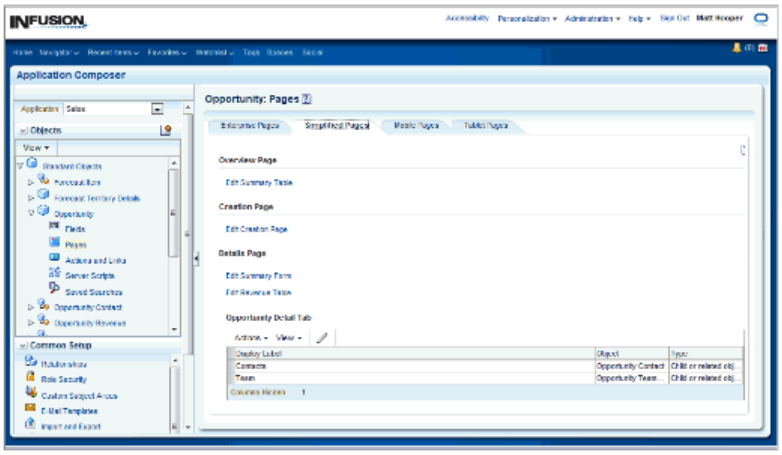
Extending Simplified Pages for Opportunities Using the Application Composer

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.

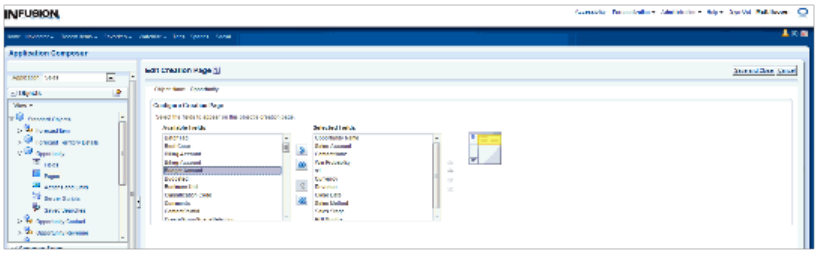
   This figure shows the Objects tree with the Opportunity object selected.



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 simplified page for Opportunities



6. For example, if you want to add the field 'Budget Amount' to the Create Opportunity page, click the Edit Creation Page link in the Simplified Pages tab.
7. In the Available Fields list on the Configure Creation page, select Budget Amount.
   This figure shows the Configure Creation page.



8. 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.

9. Click Save and Close.

Extending Simplified Pages for Opportunities using Page Composer: Worked Example

In Oracle Fusion CRM, 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.

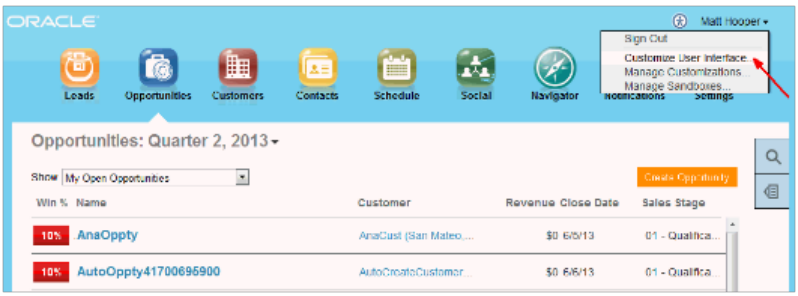
This example demonstrates how you can hide the **Include in Forecast** field for a sales representative in the Opportunities page and leave it available for the rest of the roles.

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 Fusion 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.



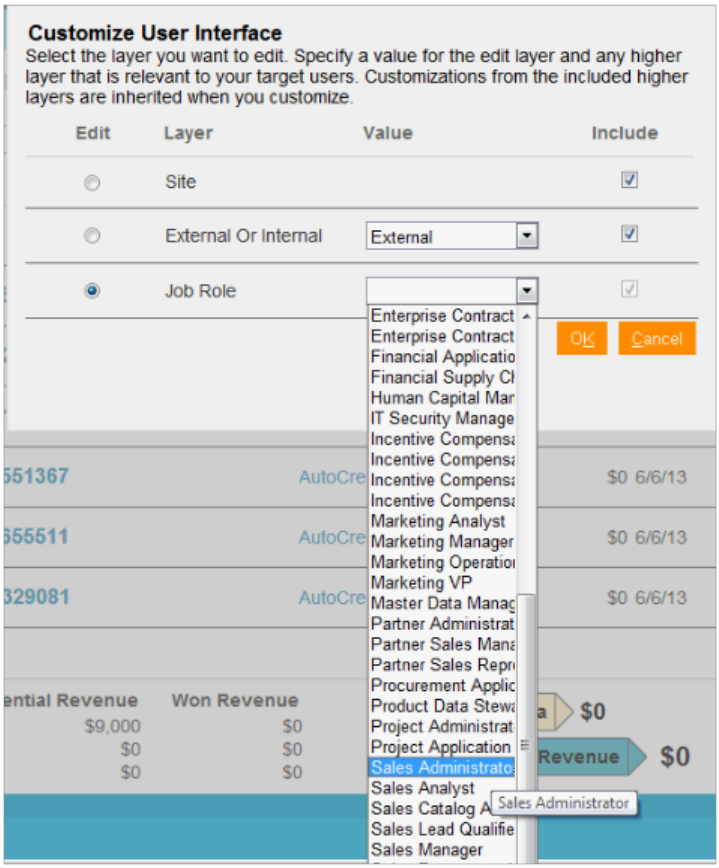
3. In the Customize Fusion Application Pages, select the layer that you want to edit.

   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.

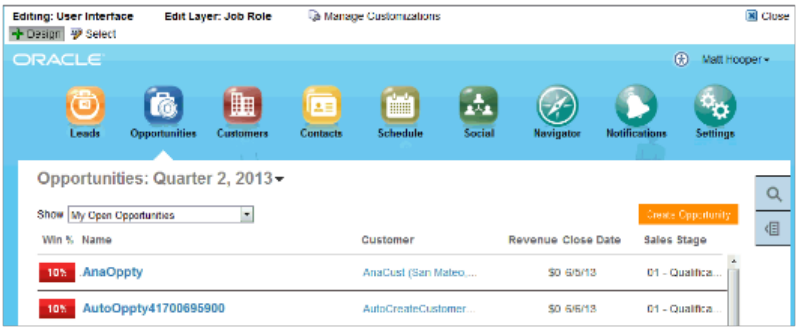


5. After you invoke the Page Composer in a particular layer (in this case, the Job Role layer) two buttons are displayed.

   The two buttons are:

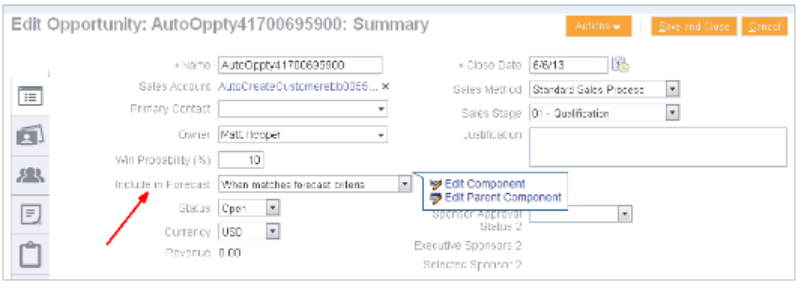
   - Design: Represents design mode used to extend transactional pages.
   - Select: Allows you to select components in a page to edit.

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



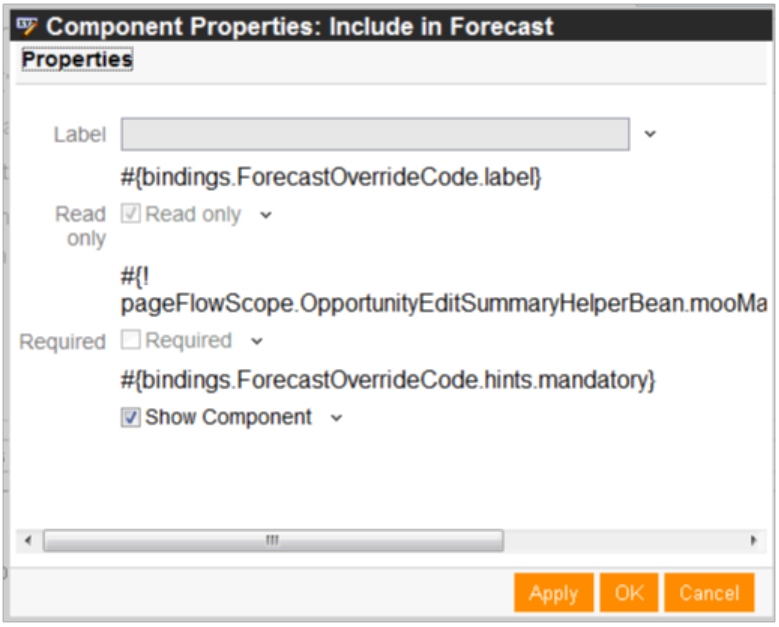
6. Drill into an opportunity and click Select.
7. In the Edit Opportunity page that appears, select the **Include in Forecast** field that you want to hide.

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.

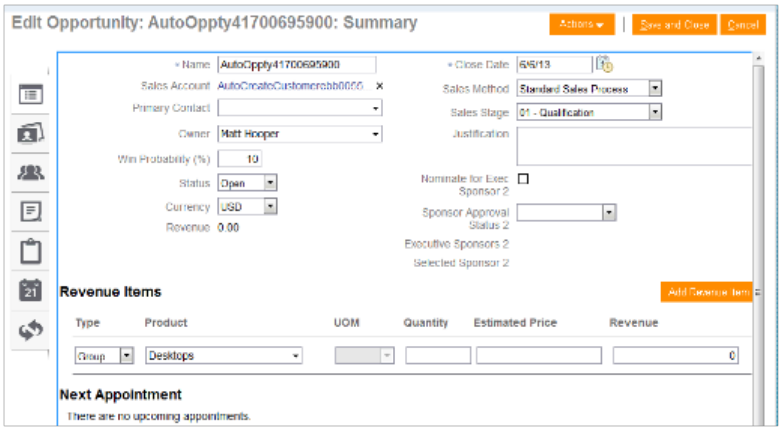
    This figure shows the Component Properties window.



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.



13

Oracle Fusion Sales Forecasting Extensibility

Oracle Fusion Sales Forecasting Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Fusion Sales Forecasting.

In this chapter, you will learn about:

- Each Oracle Fusion Sales Forecasting object, and how to extend those objects using Oracle Fusion CRM Application Composer

Customizing Oracle Fusion Sales Forecasting Pages: Explained

You can customize a variety of regions in Oracle Fusion Sales Forecasting using the Oracle Fusion CRM Application Composer. The Application Composer lets you create custom fields and objects, which you then add for display in the runtime Sales Forecasting application. To access the Application Composer, select **Application Composer** from the Navigator menu, under the Tools category.

Understanding Which Oracle Fusion 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 the Application Composer to customize those pages.

The Oracle Fusion 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 the 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 the Application Composer and create your custom fields.

Sales Forecasting Page	Sales Forecasting Region	Application	Underlying Business Object
Edit Forecast page	Territory Details region (expand the Show More option)	Sales	Forecast Territory Details
Edit Forecast page	Forecast Items tab	Sales	Forecast Item (child of Forecast Territory Details)

Next, expose those custom fields on the Forecast Items tab by accessing the appropriate Application Composer configuration page, listed in the following section.

Note

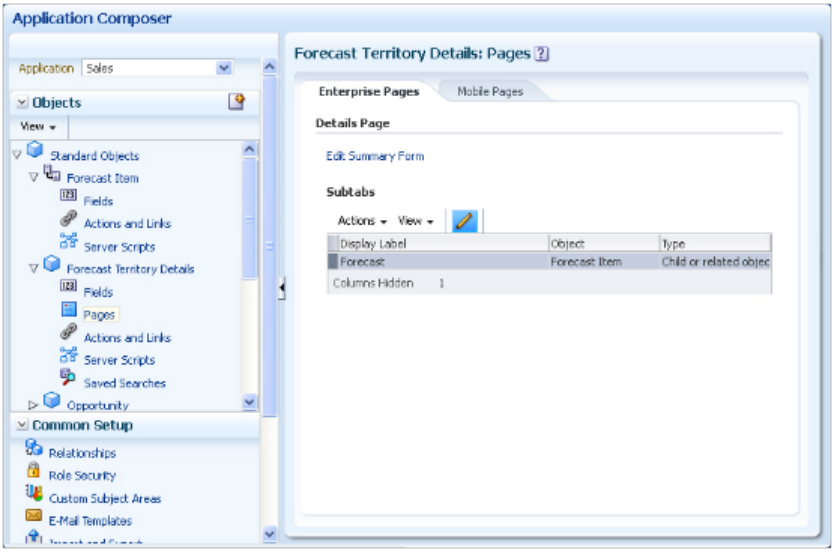
To make only minor user interface changes to Sales Forecasting pages without creating new objects or fields, use Page Composer instead of the Application Composer.

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 the Application Composer. Next, use the Application Composer's configuration pages to add those custom fields to the desired Sales Forecasting regions. You access the configuration pages in the 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.



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 the Application Composer configuration pages where you can make user interface changes on those pages and regions.

Business Object	Configuration Page in Application Composer	Related Sales Forecasting Page	Related Sales Forecasting Region
Forecast Item (child of Forecast Territory Details)	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.	Edit Forecast page	Forecast Items tab
Forecast Territory Details	Select the Pages node for this object, then click Edit Summary Form.	Edit Forecast page	Territory Details region **Note** To view your custom fields in the Territory Details region, you must expand the Show More option which appears above the Forecast Items tab.

14

Oracle Fusion Marketing Extensibility

Oracle Fusion Marketing Extensibility: Overview

Read this chapter to learn about how to extend objects and pages that belong to Oracle Fusion Marketing.

In this chapter, you will learn about:

- How to extend the sales campaign object using Oracle Fusion CRM Application Composer
- How to create a sales lead validation rule using Application Composer

Customizing Sales Campaigns in Oracle Fusion Marketing: Explained

Use the Oracle Fusion CRM 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 the Application Composer, select Application Composer from the Navigator menu, under the Tools 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 the 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.

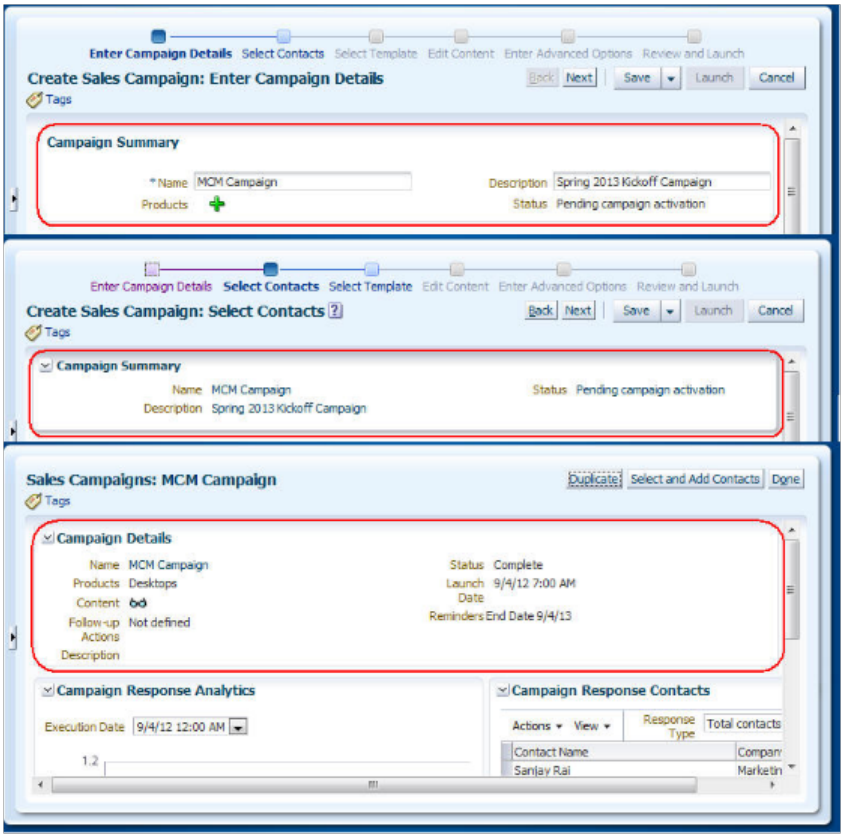
Understanding Which Oracle Fusion Sales Campaign Pages Are Extensible

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:

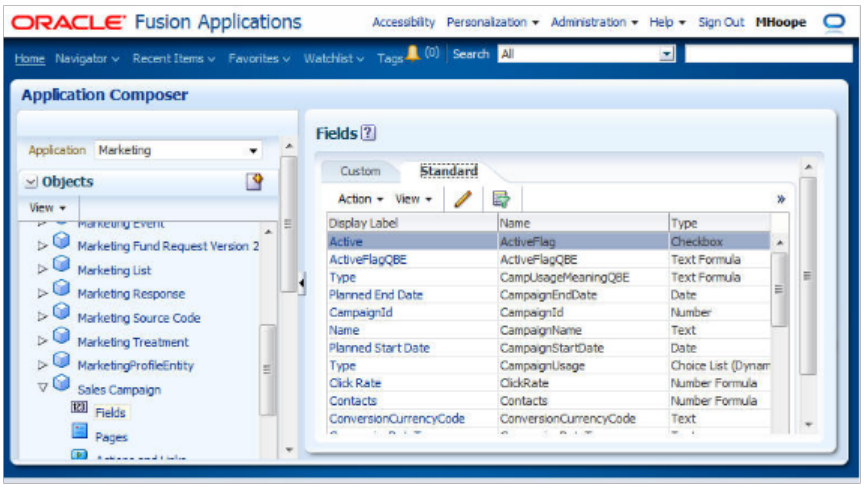
Region and Page	Are the fields editable?
Campaign Summary region on the Create Sales Campaign: Enter Campaign Details page	Yes
Campaign Summary regions on other Sales Campaign pages	Only if the campaign is in Draft status. Otherwise, not editable.
Campaign Details region (displays on Sales Campaign pages for completed campaigns)	No

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:

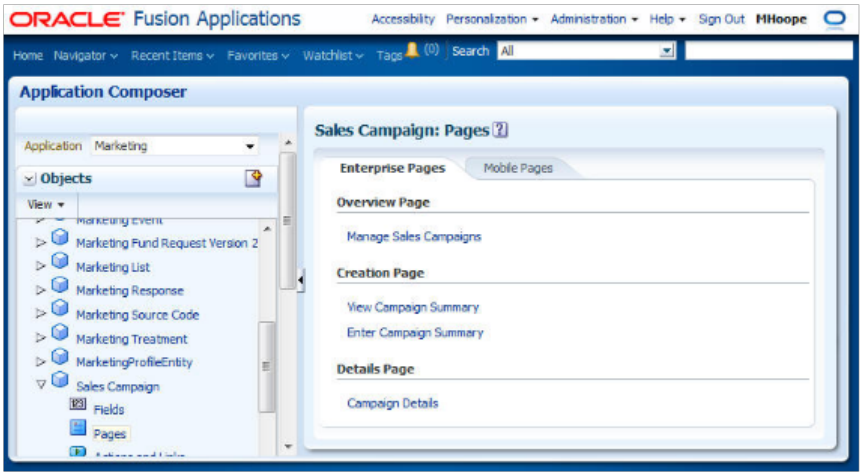
- Display Label
- Required check box
- Short Description
- Updateable
- Depends On

Adding Your Changes to the Run Time Application

After you've created custom fields for the sales campaign object, use the Application Composer's configuration pages to add those custom fields to the extensible regions listed above. You access the configuration pages in the 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.

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 the Oracle Fusion CRM 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 the **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 the **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.
```

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 Fusion Applications as an administrator.
2. Create a sandbox and activate it.

   Sandboxes in Oracle Fusion Applications 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 application; 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 application unless published.

---

19. Publish your sandbox to make these changes available for use in the mainline application.

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 application customizations overwrite the mainline application'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. Under the Administration menu, select **Manage Sandboxes...**.
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

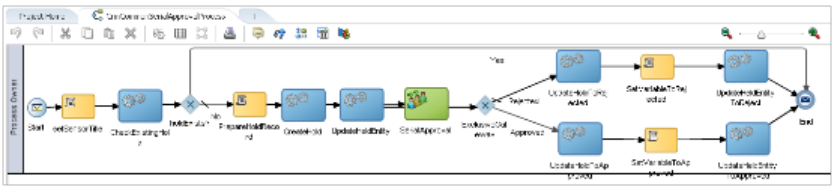
In this step, you will create a business process (also called project) in Oracle Business Process Composer using the seeded template and deploy it.

To create and deploy a process using Oracle Business Process Composer:

1. Navigate to the Application Composer using the main menu of Oracle Fusion Applications.
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**.

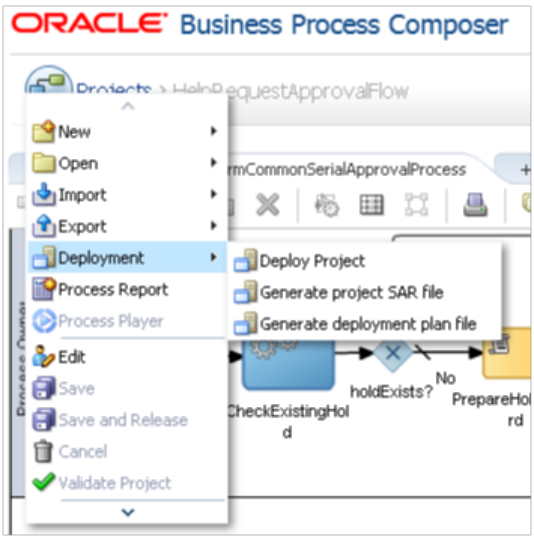
      If a popup blocker is enabled on your browser, the business process composer may not open after you click **OK**, and your browser may get locked. As a workaround, use the browser back button to go back to the Business Processes option in the application composer, and click **Edit** for the business 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:

Field	Value
Revision ID	Provide an identifiable ID.
Deployer Username	Your sign in name for Oracle Fusion Applications.
Password	Your password for Oracle Fusion Applications.
Mark composite revision as default	Select this option.

   For more information about the options in this popup and deploying business processes, refer "Deploying a BPM Project" in Oracle Fusion Middleware Business Process Composer User's Guide for Oracle Business Process Management on Oracle Technology Network, at http://www.oracle.com/technetwork/indexes/documentation/index.html.

5. Click **Deploy**.

   A confirmation message appears when your project has been deployed successfully.

6. Close the Oracle Business Process Composer.
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.

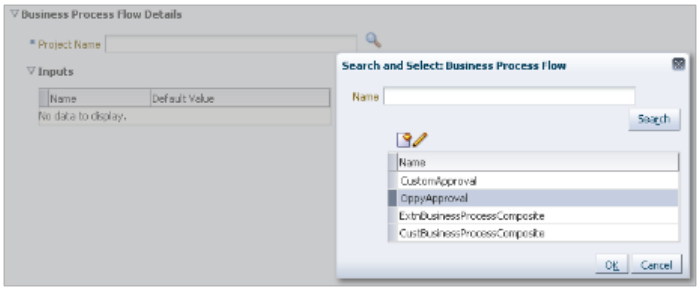
   Click on the **Groovy Builder** icon.

   In the groovy builder, enter the following event condition:

```
if(DealAmount>10000 && CurrencyCode=='USD') 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.

Name	Default Value
approvers	'SALES_MANAGER'
heldEntityStatusField	'LargeDealApproval_c'

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 Fusion Applications 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.
9. Sign out of Oracle Fusion Applications.
10. Sign in to Oracle Fusion Applications 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.
12. Sign out of Oracle Fusion Applications.
13. Sign in to Oracle Fusion Applications with a SALES_REPRESENTATIVE role.
14. Navigate to **Lead Qualification**.
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.

Extending Simplified Pages for Leads: Explained

Using Oracle Fusion CRM 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 overview table
- Create Lead page
- Edit Lead page (Summary page)

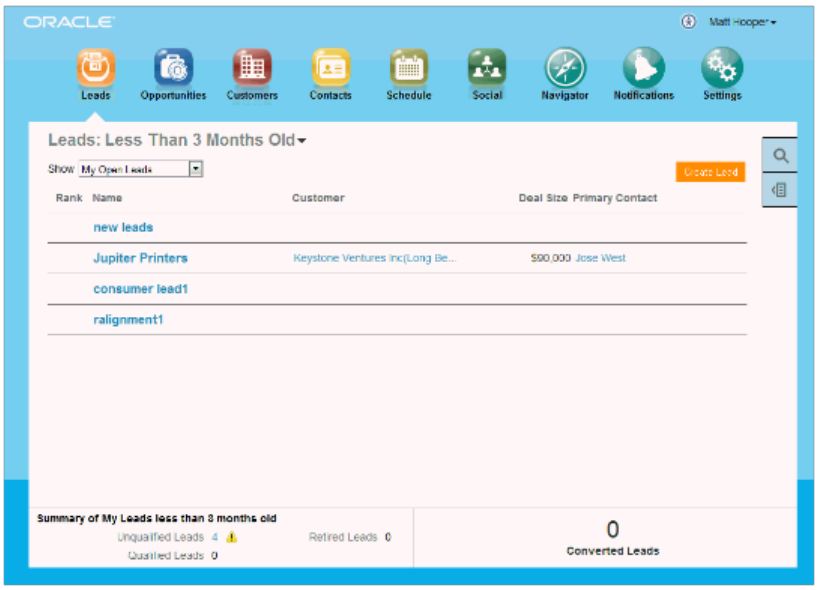
Note

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 Overview Table

You can extend the following items in the Leads overview table:

- 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:

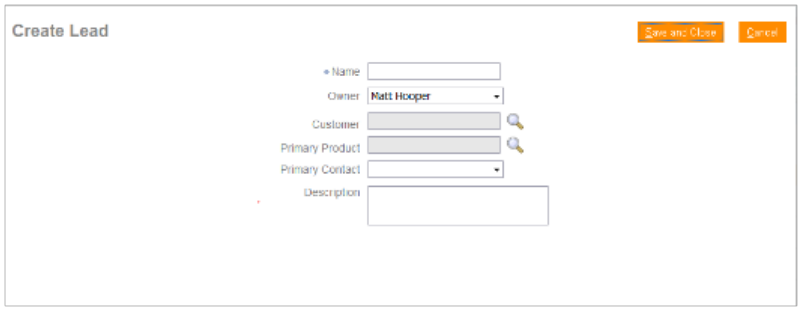
- Show filter
- Leads: Date filter
- Search
- Actions menu (cannot add or hide and show)
- Custom column widgets:
  - Rank
  - Custom font size for Name

- 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

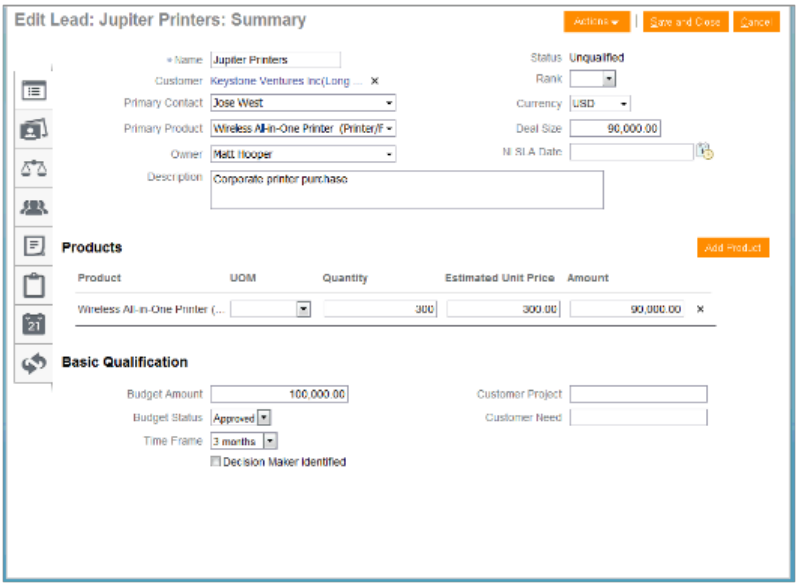
  In Application Composer, add custom fields to the Sales Lead Resorces object. Then, navigate to the Pages node for the Sales Lead Version 2 object, and edit the Sales Team subtab in the Leads tabs component region.

- 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 Oracle Fusion CRM 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:

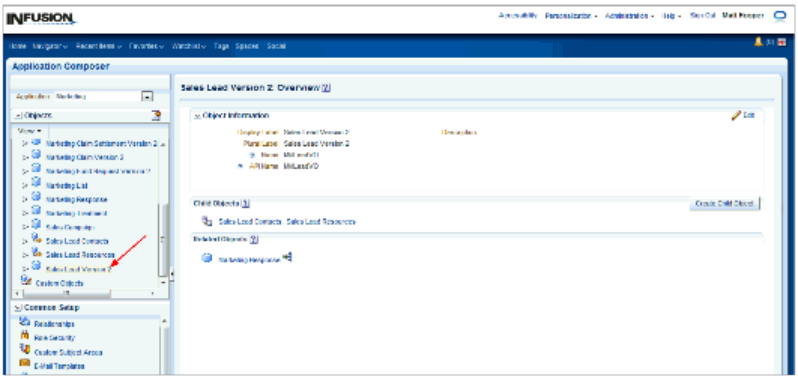
- Adding a check box to identify strategic leads
- Adding a Groovy script that checks if the size of the deal is greater than 1000 USD when a user saves a lead with this check box selected. If the deal is smaller than this amount, then users are prevented from saving the lead and receive an error message 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:

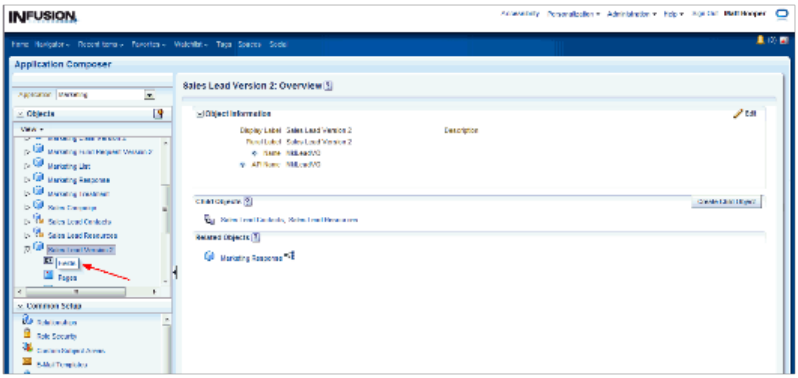
1. Navigate to Application Composer and select **Marketing** from the **Application** list. Select **Lead** from the **Objects** panel.

   This figure shows the Sales Lead Overview page in Application Composer.



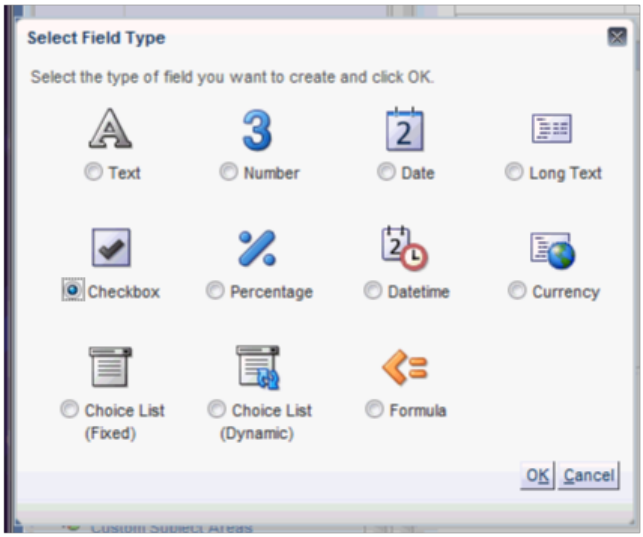
2. Under the Lead object, select the **Fields** link.

   This figure shows the Sales Lead Overview page with the **Fields** object selected for a Lead.



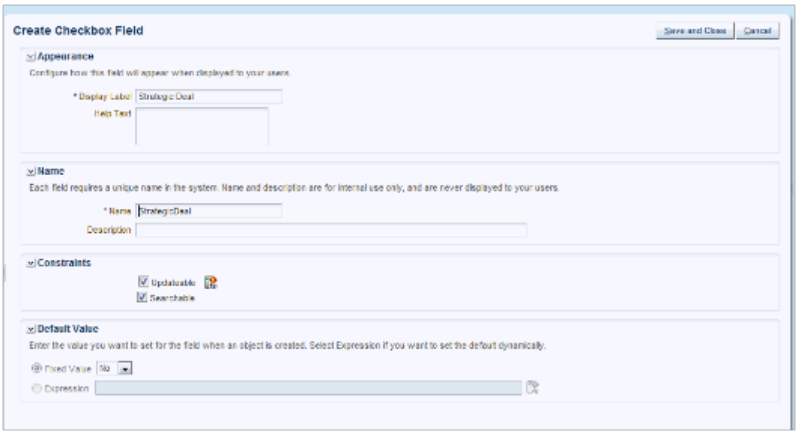
3. Click the **New** button and, in the Select Field Type window, select **Checkbox**.

   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:

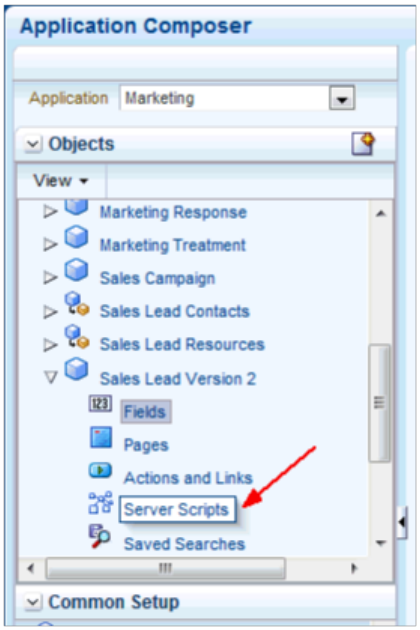
```
def retVal
if (StrategicDeal_c == 'Y')
{
 if (DealAmount >= 1000)
 {
 retVal = true
 }
 else
```

```
 {
 retVal = false
 }
}

if (StrategicDeal_c
 == null || StrategicDeal_c
 == 'N')
{
 retVal = true
}
 return(retVal)
```

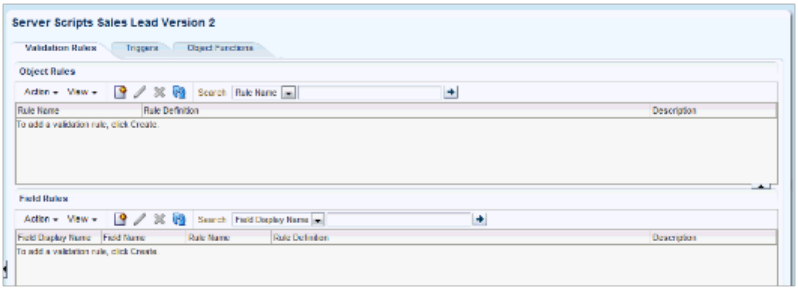
7. In the Objects panel, select **Server Scripts** under the **Sales Lead** object.

   This figure shows the Server Scripts object selected for a lead.



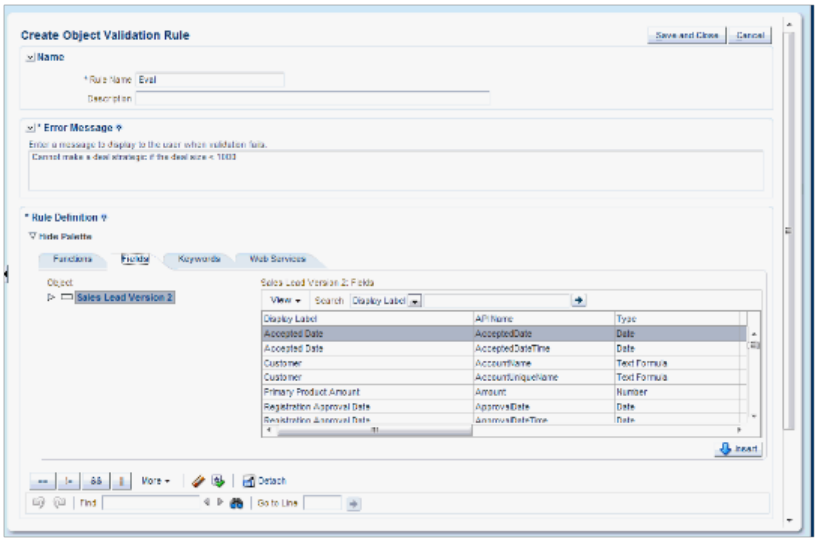
8. In the Server Scripts Lead page, select the Validation Rules tab and click the **New** button to create a new object rule.

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



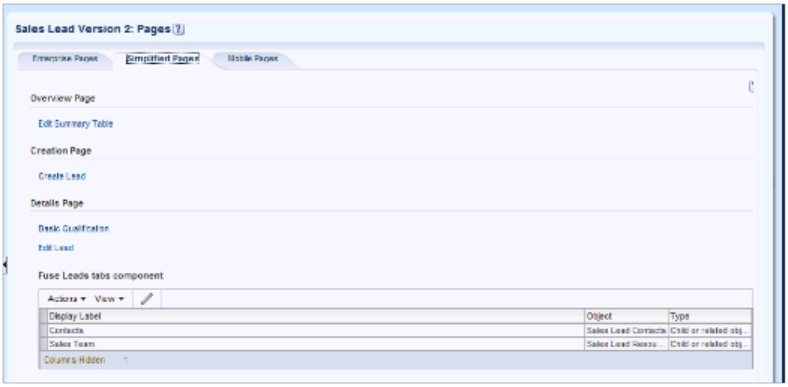
9. In the Create Object Validation Rule page, enter the rule name as "Eval" and enter a simple error message in the Error Message text region.

This figure shows the Create Object Validation Rule page.



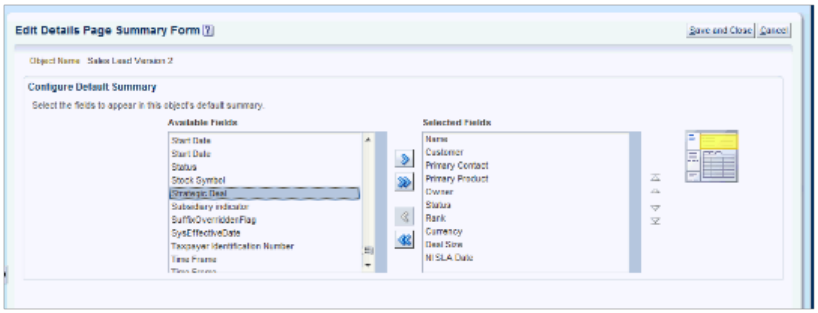
10. Cut and paste the script that you have written to validate the condition.
11. Now you need to add this field to the simplified pages. In Application Composer, select the **Pages** link under the Sales Lead object and then select the Simplified Pages tab.

    This figure shows the Simplified Pages tab.



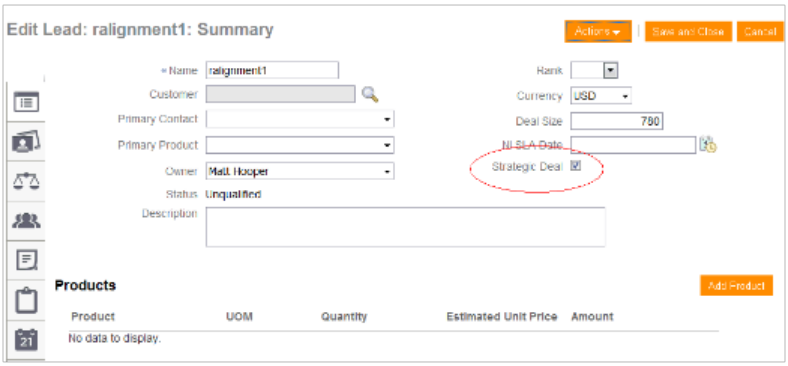
12. In the Details Page, click **Edit Lead**.
13. In the Edit Details Page Summary Form, select the **Strategic Deal** field in the Available Fields list and double click on this field to move it to the Selected Fields list.

    This figure shows the Edit Details Page Summary Form page.



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.



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.

15

Oracle Fusion Partner Relationship 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 Partner Management Dashboard and Pages Using Page Composer: Explained

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 Fusion CRM.

For more information on Source View, see "Introducing Design View and Source View" in Oracle Fusion Middleware User's Guide for Oracle WebCenter Spaces

on Oracle Technology Network at http://www.oracle.com/technetwork/indexes/documentation.

The 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.

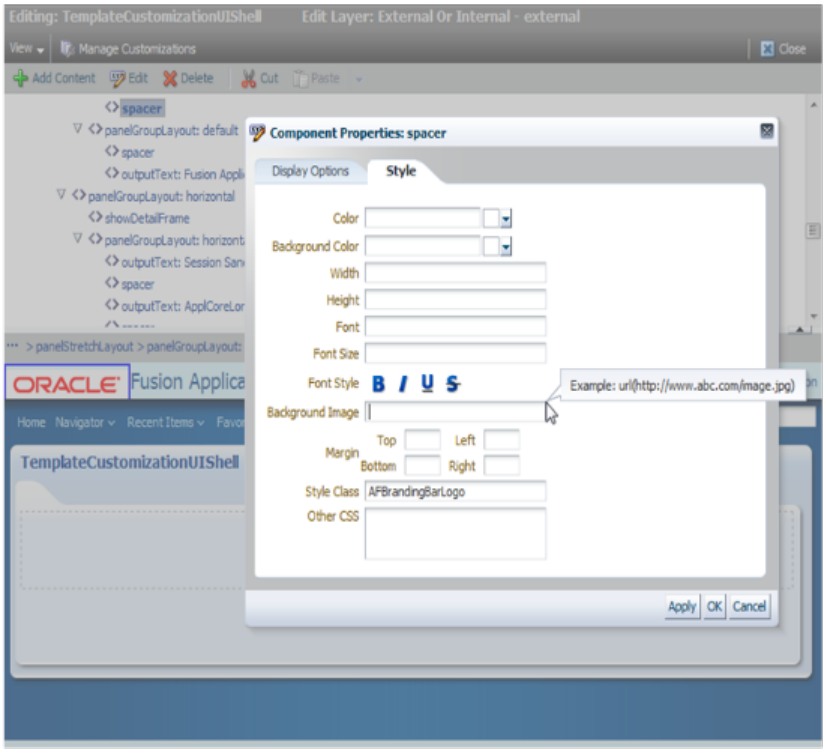
   The source tree moves to the corresponding spacer component.

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.



6. Click **Display Options** and modify the **Short Desc** value with a description.

   Tip

   This value appears when you mouse-over the logo.

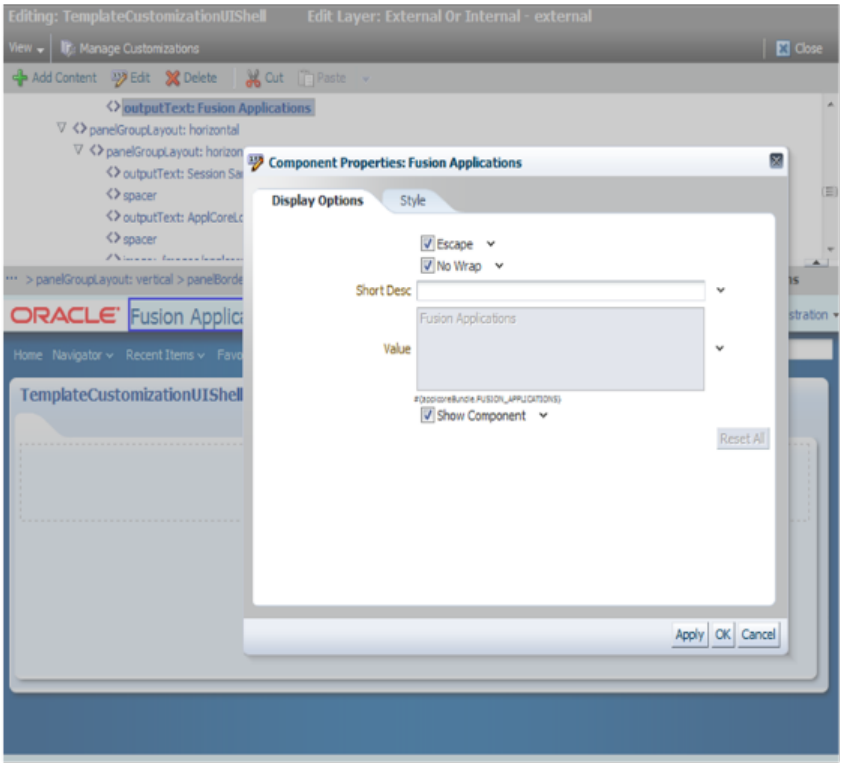
7. Click **OK**.

   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.

16

Oracle Fusion Mobile Sales Extensibility

Oracle Fusion Mobile Sales Extensibility: Overview

Read this chapter to learn about how to extend Oracle Fusion Mobile Sales.

In this chapter, you will learn about:

- How to use Oracle Fusion CRM 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

Mobile Sales Extensibility: Explained

Oracle Fusion CRM Application Composer lets implementors customize the Oracle Fusion Mobile Sales iPhone and BlackBerry applications. Using Application Composer, implementors can manage which objects and fields are visible on the Mobile Sales application without having to do specific customizations for any particular device.

Implementors can manage the following for the mobile sales application:

- Enable standard Oracle Fusion 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 Mobile Sales for mobile-enabled Sales, Customer Center, Marketing, and Common objects (standard or custom objects).

Customizing Oracle Fusion Mobile Sales: Worked Example

This example shows you how to customize Oracle Fusion Mobile Sales for a specified sales object, using Oracle Fusion CRM Application Composer. For more details about Application Composer, refer to the Oracle Fusion Applications CRM Extensibility Guide.

1. Open Oracle Fusion CRM Application Composer by using the Navigator menu, and selecting **Application Composer** under the **Tools** category.
2. Select the application you want to customize within Application Composer.
3. Within the application you chose in step 1, select the 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 Fusion 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 Fusion Mobile Sales Customizations Using a Sandbox: Worked Example

The following steps illustrate how to test Oracle Fusion Mobile Sales (Mobile Sales) customizations using a sandbox. Sandboxes are standalone

environments where you can define and test customizations, before deploying the customizations to the main Oracle Fusion 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 Oracle Fusion CRM Application Composer to customize Mobile Sales 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 Fusion CRM with a user that has a Customer Relationship Management Administrator job role.
2. Navigate to the **Administration** menu in the global area of the application and select the **Manage Sandboxes** menu item.
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 Fusion Mobile Sales Using the Oracle Fusion CRM Application Composer

1. Open the Oracle Fusion CRM Application Composer by selecting **Application Composer** under the Tools 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 Fusion CRM application.

Check Your Customizations in the Oracle Fusion Mobile Sales Application

1. Log in to Oracle Fusion CRM as an Oracle Fusion Mobile Sales 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 Fusion CRM browser window open on your laptop or PC, open Oracle Fusion Mobile Sales on your smartphone and log in using the same user you used to log in to Oracle Fusion CRM. Logging in as the same user in step 1 enables you to view the sandbox you selected in step 2 on your smartphone's Mobile Sales application. Note that only your user is accessing the sandbox on the Mobile Sales application (as long as the Oracle Fusion CRM 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 Mobile Sales users you will need to publish your sandbox. Refer to the Publishing Sandboxes chapter of the Oracle Fusion Applications Extensibility Guide for Business Analysts for more information about publishing sandboxes.

17

Exporting and Importing Customizations

Exporting and Importing Customizations: Overview

Read this chapter to learn about how to export and import customizations across Oracle Fusion application instances on the same release.

In this chapter, you will learn about:

- How to use Extensions Manager in Oracle Fusion CRM Application Composer to export and import customizations
- How to use customization sets to move customizations from one environment to another, for example from test to production

---

Note

You can also import and export custom object data (as opposed to customization metadata). In Oracle Fusion, 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 Extensions Manager, click the Metadata Manager link in the Common Setup pane, which appears in the regional area of Application Composer's home page.

To create or upload customization sets, use the Customization Migration page by selecting Customization Migration from the Navigator, under the Tools category.

Exporting and Importing Supported CRM Customizations: Explained

Once you have completed creating customizations in Oracle Fusion CRM Application Composer, you can export and import these customizations across Oracle Fusion application instances on the same release. Use the Extensions Manager option in Application Composer 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.

Exporting Customizations

Application Composer allows you to export customizations 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. Click the Metadata Manager link in the Common Setup navigation tree in the Application Composer.

   The Metadata Manager page opens.
2. Under Import or Export Metadata, click Export.

   The Export Customizations page opens.
3. Click Actions, New.

   A message opens, asking if you want to export the customizations.

4. Click Export.
5. When you are asked to open or save the compressed file, click Save.

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.

Before you import customizations into the target application instance, ensure that you publish or delete any existing sandboxes generated by Extensions Manager. Use the Manage Sandboxes option within the Administration link in the global region to publish or delete sandboxes.

Tip

The name of the sandbox created by Extensions Manager includes the term "metadataMgmt" and the date that it was imported. For example, ApplCoreLongSB_MetadataMgrnt-2013-May-21-05-20-58-710. You can use this as a tip to search for sandboxes generated by Extensions Manager.

During import, the Extensions Manager automatically generates a sandbox and then moves the customizations to this sandbox. After import, you can login to the sandbox and set it to active to review the imported customizations. All the customizations are visible and accessible in this sandbox and you can verify the changes and choose to publish the sandbox.

To import customizations, use the following steps:

1. In the target application instance, click the Metadata Manager link in the Common Setup navigation tree in the Application Composer.

   The Metadata Manager page opens.
2. Under Import or Export Metadata, click Import.

   The Import Customizations page opens.
3. Click Actions, New.
4. In the Import Customizations box, click Browse, select the compressed file of customizations, and then click Import.
5. To view the changes in the sandbox that was generated automatically during import, click Administration, Manage Sandboxes, and Set as Active the Extensions Manager generated sandbox.

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 CRM Application Composer, ensure that you manually re-key these security changes in the target instance prior to using Extensions Manager 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 Extensions Manager.

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

Use the Functional Setup Manager (FSM) tasks to migrate FND Lookups each time an import occurs.

Supported CRM Application Composer Customizations and Best Practices: Explained

You can export and import supported customizations across Oracle Fusion applications that are on the same release and same patch level. Use the Extensions Manager feature in Oracle Fusion CRM Application Composer to export and import the supported customizations. You can access the Extensions Manager from the Metadata Manager link in the CRM Application Composer.

This topic explains the following:

- Supported customizations
- Unsupported customizations
- Best practices while using Extensions Manager

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.

Unsupported Customizations

Customizations that are not supported for the Application Composer include:

- E-mail templates: Recreate manually in the target Oracle Fusion application.
- Custom subject areas: Republish all custom subject areas in the target Oracle Fusion application. See the "Publishing Custom Subject Areas: Explained" section in the Oracle Fusion Applications CRM Extensibility Guide.
- 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:

- FND lookups
- Migrate using FSM tasks: Manage Standard Lookups, Manage Custom Lookups, and Manage Set-Enabled Lookups.
- Sales Prediction Engine (SPE) business rules: Recreate manually in the target Oracle Fusion application.
- Security job roles and duty roles: Recreate manually in the target Oracle Fusion application.
- All reports, analyses, and dashboards: Recreate manually in the target Oracle Fusion application.

Note

Application Composer supports Page Composer customizations.

You must create unsupported customizations manually in the target application.

Best Practices for Using Extensions Manager

Some of the best practices for using Extensions Manager are:

- Do not manually create supported customizations in target.
- Manually create customizations which are partially supported or not supported by Extensions Manager (See list above).
- Use FSM tasks to migrate FND Lookups.
- Commit or Delete Extensions Manager generated sandboxes prior to import.
- 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 Extensions Manager.
- During an export or import, it is recommended that you do not make customization changes in the source or target instance.

Moving Customizations Using Customization Sets: Highlights

Use customization sets to move customizations from one environment to another, for example from test to production. You create a customization set in the source environment, to capture customizations from there, and upload the set in the target environment. Use the Customization Migration page (**Navigator - Tools - Customization Migration**) to create or upload customization sets.

Customization sets are fully described in the Oracle Fusion Applications Extensibility Guide for Business Analysts.

Using Customization Sets

- Only certain types of customizations are included in customization sets. You cannot change this selection.

  See: Using Customization Set Migration to Move Customizations

FAQs for Exporting and Importing Customizations

What's the difference between Extensions Manager and Customization Set Migration?

Extensions Manager is a self-service tool which migrates only CRM customizations that are created in the Oracle Fusion CRM Application Composer. Extensions Manager is specific to CRM and migrates only CRM Application Composer customizations, from one Oracle Fusion instance to another on the same release, such as from a source development environment to a target production environment.

Customization Set Migration (CSM) is a centralized framework for migrating all customizations. This includes customizations across all Oracle Fusion

Applications product families, such as Oracle Fusion Financials, Oracle Fusion Customer Relationship Management CRM), and Oracle Fusion Human Capital Management (HCM). For more details on CSM, see the section "Using Customization Set Migration to Move Customizations" in the Oracle Fusion Applications Extensibility Guide for Business Analysts. The CSM framework migrates all customizations including CRM and HCM extensions and all security changes. The CRM customizations include all Application Composer changes, Page Composer changes, and CRM security changes.

To access Extensions Manager, navigate to the Metadata Manager link in the Application Composer.

To access CSM, select Customization Migration from the Tools section in the Navigator menu in the source environment. Customization Set Migration (CSM) is accessible as a link called Customization Migration under the Tools menu item.

In order to access Extensions Manager, ensure that you have access to the following roles:

- CRM Extensibility Administration Duty
- CRM Metadata Management duty

To access CSM, ensure that you have been granted access to an administrative job role that includes the FND_CUSTOMIZATION_SET_MANAGEMENT_DUTY duty.

Extensions Manager and Customization Set Migration are self service tools which migrate (export and import) customizations from a source instance to a target application instance in the same release.

Glossary

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.

desktop page

A page in a user interface that is optimized for extended periods of use with monitors.

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.

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.

WSDL

Abbreviation for Web Services Description Language. It is an XML format that provides a model for describing Web services.