Oracle SCM Cloud

Configuring and Extending Product Lifecycle Management

22B

Oracle SCM Cloud
Configuring and Extending Product Lifecycle Management

22B

F53706-01

Authors: Divya Begur, Usha Pereira

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Use help icons Help icon to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 Extend PLM with Application Composer

About This Guide

You can configure and extend the application interfaces and business objects in Oracle Product Lifecycle Management (PLM) Cloud to support your business needs. The recommended tool for such configuration is Application Composer.

This guide describes the types of configuration that you can perform and lists what's supported and what's not. You can also find object-specific examples and frequently asked questions for your reference.

For detailed information on the capabilities of the Application Composer tool and how to use it, see the Oracle Applications Cloud Configuring Applications Using Application Composer guide.

Related Topics

- Oracle Application Cloud Configuring Applications Using Application Composer

Overview of Application Composer

Application Composer is a browser-based tool that you as an administrator can use to configure applications. You can use this tool to make data model changes that could previously be done only by application developers.

Administrators can create and configure layouts to meet business requirements. For example, you can create a new object and related fields, and then create new interface pages to expose that object to users. Application Composer is a design-at-runtime tool, which means that you can navigate to Application Composer directly from a Cloud application, make your changes, and see most changes take immediate effect, without having to sign back into the application.

> **Note:** Application Composer replaces configuration tools such as Data Composer and Page Composer. Previously created objects, attributes and other configured entities are all carried over when you upgrade your Oracle PLM Cloud applications. However, administrator-configured entities aren't initially visible to the user. Previously configured attributes become visible again when they're added to a page layout.

> **Note:** Application Composer is supported for use only in English. Additionally, Application Composer isn't supported for use with iPad devices.

What You Can Configure

Use Application Composer to:

- Edit the display label and help text of standard fields;

- Create conditional layouts;
- Assign fields to layouts;
- Create fields of different types (such as text, long text, number, date, choice list, and check box) and add them to standard and administrator-defined objects;
- Define application actions using validation rules, triggers, and functions;
- Set field-level and object-level validation rules;
- Create new objects and child objects;
- Create new subtabs and connect to external applications;
- Hide or show objects and tabs;
- Use web services to create and update objects.
- Delete custom attributes that are created in a sandbox.

> **Note:** Ensure that the custom attributes you're deleting aren't in use or have never been published.

Attributes, or fields, must be assigned to a layout for the application user to see and work with them. A conditional statement assigned to a layout determines when it's displayed and who can see it.

Configuration Scenarios

Let's look at some scenarios where you can tailor the application interface to achieve a specific requirement.

Scenario 1

Your business process requires that a requirements specification must be created for every idea that has an Approved status. Also, the requirements specification must have the same name and description as the original idea.

Solution A: Within Application Composer you can add a button to ideas that invokes a web service which will create a requirements specification and copy the name and description of the idea to the new requirements specification.

Solution B: Within Application Composer you can create a trigger that invokes a web service whenever an idea is set to Approved status, which will create a requirements specification and copy the name and description of the Idea to the new requirements specification.

Scenario 2

On a quality issue, your users must select a service type and then a service center. The selection of the service center is dependent upon the selection of the service type.

Solution A: Within Application Composer, create two fixed choice list fields with appropriate values for service type and service center. Designate the service type field as the parent of the service center field and map the values as desired so that when a user selects a service type, only the applicable service centers are available for selection.

Solution B: Within Application Composer, create two fixed choice list fields with appropriate values for service type and service center. Use Groovy scripting to create a trigger that will automatically select the service center based on the selection of service type.

Scenario 3

When users create a quality action, you want to make sure the triage date is automatically set to three days after the creation date.

Solution: Within Application Composer, create a new date field called Triage Date. Set the field to not be updatable and define the default value using a Groovy script expression that captured the creation date and adds three days.

Configuration Nodes

When you open Application Composer, each object type that's available to you is displayed, along with a set of configuration nodes. Here's what you can do in each node:

- Fields: modify standard fields and create new fields.
- Pages: create and modify page layouts.
- Actions and Links: create internal actions or links to external applications with Groovy scripts.
- Server Scripts: create validation rules, triggers, and object functions with Groovy scripts.

> **Note:** Supported functionality for change orders differs from other objects.

Related Topics

- Define Fields
- Application Pages for Custom Objects
- Actions and Links
- Server Scripts

PLM Objects You Can Configure

PLM uses various objects in its work flows, and you can configure or extend all of them to some extent. When you plan your configuration requirements, be aware of what's supported for each object type.

Object Types

You can configure the landing page, the Create dialog box, and the Details page of the following objects:

- Idea
- Requirements Specification and Requirements
- Proposal
- Quality Action
- Quality Issue

- Change Order
- Change Request
- Problem Report
- Corrective Action
- Concept

  Note: You can also configure the landing page and details page of concepts, but not its Create dialog box.

Note: A Create dialog box must include all fields that are tagged as Required. This includes Required fields that you have renamed. If not, an error message appears as you set up the Create dialog box for a business object.

What You Can Configure For Each Object Type

This table summarizes the specific kinds of configuration you can perform on each PLM object type.

Note: Configuration for Change Orders is not covered in this table.

KINDS OF CONFIGURATION	APPLIES TO THESE OBJECT TYPES	EXCEPTIONS
Additional Attribute Types, which include: • Dynamic Choice List • Fixed Choice List • Check box • Text • Number • Currency • Percentage • Datetime • Date • Long text • Formula • Record Type	**Innovation Management**: • Idea • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue • Quality Actions	**Innovation Management**: • Concept structures don't support Long text and Formulas.
Page Layouts	**Innovation Management**: • Idea • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue	Not applicable.

KINDS OF CONFIGURATION	APPLIES TO THESE OBJECT TYPES	EXCEPTIONS
	• Quality Action	
Subtabs, that includes • Context Link • Relationship	**Innovation Management**: • Idea • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue • Quality Action	Not applicable.
Show or Hide Tabs	**Innovation Management**: • Idea • Requirements Specification • Proposal • Concept **Quality Management**: • Quality Issue • Quality Action	**Innovation Management**: • Requirement • Component
Configure Buttons or Actions	**Innovation Management**: • Idea • Requirements Specification • Proposal **Quality Management**: • Quality Issue • Quality Action	**Innovation Management**: • Requirement • Concept
Field Groups	**Innovation Management**: • Idea • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue • Quality Action	Not applicable
Configure URL Tabs	**Innovation Management**:	**Innovation Management**:

KINDS OF CONFIGURATION	APPLIES TO THESE OBJECT TYPES	EXCEPTIONS
	• Idea • Requirements Specification and Requirement • Proposal • Concept and Component	• Requirement **Quality Management**: • Quality Issue • Quality Action
Configure first-level Objects	**Innovation Management**: • Idea • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue • Quality Action	**Innovation Management**: • Requirement
Configure Child Objects	**Innovation Management**: • Ideas • Requirements Specification and Requirement • Proposal • Concept and Component **Quality Management**: • Quality Issue • Quality Action	**Innovation Management**: • Requirement

Note: In Oracle PLM Cloud applications, you're always in a live environment. When you plan to create user interface entities in Application Composer, you must first open a practice state sandbox. Once that's done, it's safe to open Application Composer and work without changing the production interface pages until you're completely prepared.

What You Can Configure For Change Orders

Let's look at the configurations that are applicable for change orders.

- Page Layout
- Subtabs that include context links and relationships
- Show or Hide Tabs
- Buttons or Actions
- URL Tabs
- First-level Objects

- Child Objects

Note: In the Pages node for Change Orders, you can make page layouts for the Details page but you can't modify the fields on the page layout. You can't configure the landing page or the Create dialog box.

Related Topics

- Overview of Using Application Composer
- Configuration Life Cycle

Set Up Access

You must be assigned certain job roles along with associated privileges to access and work in Application Composer.

Ensure that you have the following roles:

- Any role that contains the Manage Extensible Object privilege, for example, Product Data Steward.
- Custom Objects Administration role.
- Product Manager role.
- Quality Analyst role.

Related Topics

- Get Started

Use a Sandbox

You use sandboxes to make application changes and test them without impacting other users in the environment. Make changes to the application whenever you're in a sandbox rather than making direct changes in the mainline environment.

To access the sandboxes, navigate to **Configuration** > **Sandboxes** work area.

Completed application changes created within a test-only sandbox have to be published to be available to other users in the application in the mainline metadata. While creating the sandbox, in the Publishable field, select **Yes** or **No**. If you set the Publishable option to **No**, you can use your sandbox for testing only, but can't publish it.

Once in a sandbox, you can open Application Composer and work without changing the production interface pages. Also, a best practice is to create a new sandbox for each process change you intend to implement. As an example, say you want to create a set of new basic fields that fit your business process but have no dependency on the other fields, do that in one sandbox and publish it. If you want to create a validation for field or set of fields, do all of that in one sandbox and publish it. Therefore, you can group your modifications together according to the functionality that you want to provide and release/publish them incrementally one sandbox at a time.

Use the Unified Sandboxes UI, which is the default feature you get.

Note:

- Don't create or publish sandbox for Application Composer or Page Composer customizations while other users or processes are deploying extensible flexfields and descriptive flexfields configuration changes for Product Development objects.
- Don't deploy flexfields while you have an open sandbox in which you are making Application Composer or Page Composer customizations that you want to publish. Don't deploy the flexfields until the publish operation is complete. You should make sure that all the Application Composer or Page Composer customizations in sandbox are tested and published before deploying any extensible flexfields and descriptive flexfields configuration changes for Product Development Objects.

Related Topics

- Overview of Sandboxes
- Create and Activate Sandboxes

Get Started

The following procedure is an extremely abbreviated sequence to open Application Composer and have a look around.

1. Navigate to the Application Composer work area.
2. When you open Application Composer, the Application list offers **CRM Cloud** and **ERP and SCM Cloud** environments. Select ERP and SCM Cloud.
3. In the Objects row object tags are displayed. You can only see business objects that you have permission to access.
4. In this brief procedure, you can choose a business object, and we're selecting **Idea**. An idea is a standard object, populated with fields, or attributes. Navigate to **Objects** > **Standard Objects** > **Idea** > **Fields** > **Create** > **Select Field Type** to create a field.
5. After creating the fields, go to **Pages**, assign the Text fields to a duplicated **Landing** page layout, and click **Save**.
6. Open **Ideas** > **Manage Ideas** and click on an existing Idea to see the new attribute.

Related Topics

- PLM Objects You Can Configure
- Define Objects

Review Published Configuration Changes

After you finish configuration tasks in Application Composer, you can review your changes to make sure everything is in order. In Metadata Manager, click **Generate Configuration Report** to view a summary of configurations made to layout details against a published sandbox.

You can view a summary of modifications across the following in the Application Composer Configuration Report:

- Standard and Custom Objects
- Global and Object Functions
- Standard Fields or Custom Fields
- Custom Relationships
- Validations
- Triggers
- Object Workflows
- Dynamic Layout

Related Topics

- Tools for Moving and Troubleshooting Configurations

2 Examples

Create a Quality Object Using Groovy Script

In Application Composer, use groovy scripting to create quality objects. This means you can create, associate, and update quality issues and actions and their related data from within other Application Composer enabled objects.

Let's say you have a business process that requires all quality issues to have a corresponding CAPA. To help enforce this process, you write a groovy script action that creates a CAPA when the user clicks a button in the issue.

1. Navigate to **Application Composer** work area.

   > **Note:** Ensure that you're in a sandbox.

2. Select the **ERP and SCM Cloud** option from the **Application** list.
3. Select the **Quality** check box in the Object tags.
4. Expand **Standard Objects > Quality Issue > Server Scripts** to create an object function script.
5. To create an object function script:

   a. On the Server Scripts Quality Issue page, click **Object Functions**.
   b. On the Create Object Function page, enter a name in the **Function Name** field. For example: createCAPA.
   c. In the Edit Script pane, enter groovy script details:

```
def actionName = actionName("CAPA for" + Name)
def qaView = newView('Action')
def newQA = qaView.createRow()
newQA.setAttribute('ActionType', 5120)
newQA.setAttribute('Name', actionName)
newQA.setAttribute('Description', "CAPA for " + Name + " to determine the root cause.")
newQA.setAttribute('Priority', "ORA_HIGH")
qaView.insertRow(newQA)
```

   For standard types, you get the ActionTypeId from the following table:

   Quality Issues

Name	Code	ID
Design Failure - Item	ORA_ENQ_QI_DF_ITEM	1304
Maintenance - Component	ORA_ENQ_MNT_COMPONENT	1274
Maintenance - Miscellaneous	ORA_ENQ_MNT_MISCELLANEOUS	1290
Maintenance - Resource	ORA_ENQ_MNT_RESOURCE	1278

Name	Code	ID
Maintenance - Supplier Operation	ORA_ENQ_MNT_SUPPLIER_OPERATION	1294
Maintenance - Work Area	ORA_ENQ_MNT_WORK_AREA	1286
Maintenance - Work Center	ORA_ENQ_MNT_Work_CENTER	1282
Manufacturing - Component	ORA_ENQ_COMPONENT	1210
Manufacturing - Miscellaneous	ORA_ENQ_MISCELLANEOUS	1250
Manufacturing - Resource	ORA_ENQ_RESOURCE	1220
Manufacturing - Supplier Operation	ORA_ENQ_SUPPLIER_OPERATION	1215
Manufacturing - Work Area	ORA_ENQ_WORK_AREA	1240
Manufacturing - Work Center	ORA_ENQ_WORK_CENTER	1230
Non-Conformance - Audit Finding	ORA_ENQ_QI_NC_AUDIT_FINDING	1112
Non-Conformance - Development Item	ORA_ENQ_QI_NC_DEV_ITEM	1108
Non-Conformance - Inventory	ORA_ENQ_QI_NC_INVENTORY	1116
Non-Conformance - Outside Processing	ORA_ENQ_QI_NC_OSP	1104
Non-Conformance - Receiving	ORA_ENQ_QI_NC_RECEIVING	1120
Non-Conformance - Resource	ORA_ENQ_QI_NC_RESOURCE	1124
Non-Conformance - Work in Process	ORA_ENQ_QI_NC_WORK_IN_PROCESS	1128
Problem Report - Item	ORA_ENQ_QI_PR_ITEM	1404
Problem Report - Document	ORA_ENQ_QI_PR_SUPPLIER	1408

Name	Code	ID

Quality Actions

Name	Code	ID
CAPA - Corrective Action	ORA_ENQ_CORRECTIVE	5110
CAPA - Preventive Action	ORA_ENQ_PREVENTIVE	5120
CAPA - Supplier Corrective Action	ORA_ENQ_SUPPLIER_CORRECTIVE	5130
CAPA - Development Action	ORA_ENQ_DEVELOPMENT	5140
Failure Analysis - Design	ORA_ENQ_DESIGN	5210
Failure Analysis - Manufacturing	ORA_ENQ_ACTIONS_MANUFACTURING	5220
Audit - Audit Finding	ORA_ENQ_AUDIT_FINDING	5310

d. Click **Save and Close**

Let's create an action link, since the created object functions work only if attached to an action.

6. Click **Action and Links**.

a. On the Quality Issue: Create Action or Link page, click the **Create** button. The display label is what you see on the button that you click. Enter a value. For example: Create CAPA. Internal names are automatically filled.
b. Select the Method Name that you created earlier in the Server Script node.
c. Click **Save**.

The action works only if it's added to a page layout. So, let's add an action to the page.

7. Click **Pages**.
    a. On the Quality Issue: Pages page, go to Details Page Layouts panel and click the **Duplicate** button to create a duplicate layout.
    b. Enter a name in the **New Layout Name** field. For Example: Default custom layout.
    c. Click **Save and Edit**.
    d. On the Details Layout: Default custom layout page, click the **Edit** icon next to the **Actions** menu. You see that the action name is available as a button and as an action. Here's where you choose how the action name must appear - as a button or as an option on the **Actions** menu.
    e. Move the action name to the **Selected Buttons** or **Selected Actions** panel.
    f. Click **Save and Close**.
    g. Click **Done**. You can see that the action name appears as a button next to the **Actions** menu or as an option on the **Actions** menu.

   Now, let's check if the settings that were set works.
8. Navigate to the **Quality Management** work area.
    a. Create a new quality issue, filling in the required fields. Once the issue is created, click the new CreateCAPA button.
    b. Run a search for quality actions and the newly created quality action appears in the search results. You can see that the button created the new CAPA using the groovy script in the quality issue.

Related Topics

- Groovy Scripting

Add a Child Object as a Subtab to an Idea

In this example, you can create and add your own tab to display a child object or related object.

Standard objects come with tabs, and each of the standard objects lets you create and add multiple instances of the child object to one instance of the parent. You can add the child object as a tab to the parent object. Let's look at how we can create a child object and have this child object appear as a subtab on an idea.

First, create a child object.

Create a Child Object

1. Navigate to the Application Composer work area.
2. Select **ERP and SCM Cloud** from the Application menu.
3. Expand **Standard Objects** and click the **Idea** link.
4. Click the **Create Child Object** button.
    a. Fill in the required information in the Create Child Object window.
    - **Display Label**: This will appear as node name in Application Composer - let's call the child object Market Research.
    - **Plural Label**: This will appear as the page title for the object's work area and the icon name in Navigator - we will keep this also as Market Research.
    - **Record Name Label**: This will be the name of the field where you enter the record name. Here it will be Market Research Name.

- **Record Name Data Type**: This can either be user-entered text or an autogenerated sequence to identify a record.
  - Use "Text" for users to enter their own names for each record.
  - Use "Automatically Generated Sequence" to automatically create the name for each record.

    > **Note:** For more information on Automatically Generated Sequence, refer to Set up Autonumbering Schema for Quality Objects and How You Set Up the Number Generation Method for a Quality Issue or Action Type. The links to these related topics are provided.

- **Object Name**: This is the internal name of the object. In this example: MarketResearch.
- **Child Collection Name**: This is automatically created and used to assign a generic function across the collection or list. Here we refer to it as MarketResearchCollection.
- Click **OK**.

  The Idea: Overview page opens. To change the existing icon, click **Change Icon** and select an icon from the Icons page.

5. Next, configure fields, page layouts, server scripts, actions and links, and assign roles to secure the object.
   - **Fields** - though the idea object comes with standard fields, you can create additional fields as per your need.
   - **Security** - define who can use the child object. In the Security node, set up which roles have the privilege to use the object.
   - **Pages** - since a new object has no page layouts when created, you must generate default pages. Once pages are created, you can edit them and also add new pages. To do this:

   a. Go to **Pages** (Standard Objects > Idea > Pages) and duplicate a standard layout for these pages.
   b. In the Details Page Layouts region, select the standard layout and click **Duplicate** from the **Actions** menu to duplicate it, or select another layout.
   c. In the Duplicate Layout window, enter the **New Layout Name** and click **Save and Edit**. Now you can add a subtab to this page layout.

Add a Child Object as a Subtab

1. On the Details Layout page of the parent object, click the **Add** icon in the Subtabs region.
2. In the **Create Subtab** page, you can add the following types of subtabs:
   - Related Object
   - Child Object
   - Context Link
   - Common Component
   - Mashup Content

   Let's select the **Child Object** option and click **Next**.
3. Enter the basic information. Select the data object from the list of options.
   - Data Object: The child object you created earlier - Market Research.
   - Display Label: Takes the name from the child object you created.
   - Display Icon: If you have specified a display icon while creating the child object, it appears here.
   - Select the fields you want to display on the subtab summary table at runtime and click **Next**.

4. Select additional layouts if required.
5. Click **Save and Close**.
   The new subtab appears on the layout page.

Related Topics

- Overview of Subtabs
- Set Up Autonumbering Schema for Quality Objects
- How You Set Up the Number Generation Method for a Quality Issue or Action Type

Add Custom Object in Quality Issue

You can configure a custom object to appear as a subtab (or side tab) on a quality issue.

To do this, create the object as a child object of the standard quality issue object. In Application Composer, the child object appears in the Standard Object node hierarchy. Each child object you create can appear as separate subtabs on the quality issue.

You can also create a custom object at the top level (same level as the standard objects), and use it in multiple objects. For example, you could have it appear as a subtab on a quality issue and also on a quality action. You could also have multiple custom objects appear within a single subtab on a quality issue. To do this, you must create relationships between the custom object and the standard objects.

The following example illustrates how you can create a custom object called Surface Defect and have it appear in a quality issue.

Here's an outline of how you configure the custom object:

1. Create a custom object with tasks and fields.
2. Define a relationship between the custom object and the quality issue.
3. Define the custom object in the standard quality object.
4. Verify the custom object in quality issue.

Create Custom Object

1. Within a sandbox, navigate to Application Composer work area.
2. Select **ERP and SCM Cloud** option from Application Composer.
3. Click **Custom Objects**.
4. In the Custom Objects page, click **Create** from the **Actions** tab.
5. In the Create Custom Object page, enter the Display Label as Surface Defect. Click **OK**.
   A custom object with the name Surface Defect is created.
6. For the Surface Defect custom object, click the **Fields** tab and create two custom fields - a text type field called Severity Description and another number type field called as Severity Code.
7. For the Surface Defect custom object, click the **Pages** tab, and then click **Create Default Pages**. This creates the following: Landing Page Layouts, Creation Page Layouts, Details Page Layouts, and Reusable Regions.
8. In the Creation Page Layouts, click the **Default custom layout** to edit it. The Creation Layout: Default custom layout page appears.
9. In the Creation Layout: Default custom layout page, click the **Edit** icon next to FuseSurface DefectCreate.
10. In the Configure Detail Form, add the Surface Defect Name, the Severity Code, and the Severity Description fields in the Selected Fields column. These fields will appear in the custom object page.

11. Click **Save and Close**.
12. Click **Done** to exit the Creation Layout: Default custom layout.
13. Select the Security node and verify that the Custom Objects Administration role is enabled. To view, create, or modify instances of this custom object, your user account must have the Custom Object Administration role. Click **Save and Close**.

Define Relationship between Custom Object and Quality Issue

1. In Application Composer, in **ERP and SCM Cloud**, select **Relationships** in the Common Setup pane.
2. On the Relationships page, click the **Create** icon.
3. On the Create Relationship page, select the Source Object as Quality Issue and the Target Object as Surface Defect. Set the Cardinality as 1:M.
4. Enter a name such as QualityIssueSubtab, and a Display Name such as Quality Issue Subtab.
5. Click **Save and Close**.

   With this, you have created a 1:M relationship between the Surface Defect custom object and the Quality Issue standard object.

Define Custom Object in Standard Quality Issue

1. In Application Composer, in **ERP and SCM Cloud**, select **Standard Objects** > **Quality Issue**. Click **Pages**.
2. In the Details Page Layouts region, click the Duplicate Details Page Layout icon to create a new layout and enter a name as surface defect for the layout.
3. Click **Save and Edit**.

   You can also click the link of an existing layout (not Standard layout) to edit it.
4. In the Details Layout: Default custom layout page, in the Subtabs Region, click the **Add** tab.
5. In the Details Layout: Default custom layout: Create subtab page, select **Related object.** Click **Next**.
6. In the Create Subtab page, select the related object that's to be exposed on the subtab as the Data Object.
7. In the Configure Summary tab, select the fields and links you want to display at runtime, including the Severity Description and Severity Code fields that you created in the first step of this process.
8. Click **Next**.
9. Click **Save and Close**. In the Details layout: surface_defect page, you can see the Surface Defect subtab in the Subtabs Region.
10. Click **Done**.

Verify Custom Object in Quality Issue

1. While still in the sandbox, navigate to the Quality Management work area.
2. Search for and open a quality issue or create a quality issue.
3. Select the newly created subtab from the page.
4. Click **Create** and enter values in the Severity Description and Severity Code custom fields.
5. Click **Save and Close**.

   When you open the quality issue to which the subtab is linked, you can see that the custom fields display the updated values.

Create a Project from a Proposal

Your company is using proposals in Innovation Management Cloud to capture proposal data and approve the proposals to implement.

You can roll out Project Management Cloud and set up an automation such that when a proposal is approved, a project of the same name is created automatically.

The following table summarizes implementation challenges and solutions to solve them:

Challenges	Solutions
Groovy scripting hasn't been implemented for Projects yet	Use REST web services instead of Groovy to create the project
When you create a project you must assign a user as project manager or the project isn't available in the UI	Use two REST web service calls, one to create the project and the other to assign a project manager
Users who create and manage proposals aren't the same users who manage projects	Add a new list field with the names of users that have the privileges to be project managers

Here are some additional factors to consider:

What to consider	Action to take
You must access the Team table to assign a user as a project manager and to access the Team table you must use a REST URL that won't come into existence until you create a project.	Use Groovy scripting to access the Project ID from the response that returns after creating the project, and pass that Project ID to another web service call that contains a dynamic parameter. **Note:** When establishing a web service connection in Application Composer, you can set dynamic parameters or "variables" in the REST URL. Using these "variables", send data from a Groovy script into the URL so that the URL calls the correct REST API information from within an object
You must use the user's email address to assign the project manager, but the number of characters is too long to put in a fixed list.	Create a fixed list with the user names of potential project managers and use a formula field to convert the user name to an email address.

Tasks Summary

1. Create a web service connection to add a project.

2. Create a web service connection to add a project manager.
3. Create a fixed list and formula field.
4. Create an object trigger to create a project and add a project manager.

   These are the tasks that you must perform to automatically create a project once a proposal is approved. Let's go through each of them in detail.

Create a Web Service Connection to Create a Project

1. Navigate to **Application Composer (Sandbox bar > Tools > Application Composer)**.
2. Select **ERP and SCM Cloud** from the Application list.
3. From Common Setup, choose the Web Services node and click **Create Web Service Reference** icon.
4. Select **REST** and click **OK**.
   a. Enter the following details:
      - **Name**: Project01
      - **URL**: https://<your host name>/fscmRestApi/resources/latest/projects
      - **Security scheme**: Call with basic authentication
      - **Credential key**: click the + icon:
        - CSF Key Field: PROJECT_ACE
        - User Name: Jane Doe (User must have Create Project privileges)
        - Password: Enter appropriate password and click **OK**.
   b. In Select and configure Methods against the Resource:
      - Select the **Post** option.
      - For the Request Payload, select the **Code Sample** option, and enter the following code: `{ProjectName:"Name"}`. This sample defines which parameters you need to use when you create the trigger and to pass the proposal name to the project.
      - For the Response Payload, select the **Code Sample** option and enter the following code: `{ }`

        > **Note:** The empty braces indicate that you want a response, otherwise it returns null.
5. Click **Save and Close**.

Create a Web Service Connection to Add a Project Manager

1. Click the **Create Web Service Reference** icon, select REST and click **OK**.
2. Enter the following details:
   a. **Name**: Project Manager 01
   b. **URL**: https://<host>/fscmRestApi/resources/latest/projects/##projectId##/child/ProjectTeamMembers
   c. **Security scheme**: Call with basic authentication
   d. **Credential key**: Password

      > **Note:** You can use the same credential key that you created in the prior section or create a new one, just make sure that the account has the privilege to assign project managers.

   e. In the Select and configure Methods against the Resource region, click the **Post** option.

- In Request Payload, click Code Sample, and enter the following code: `{PersonEmail: "jane.doe@example.com", ProjectRole: "Project Manager 01", StartDate: "2019-09-25"}`

  Note: This sample defines which parameters you need to use when you create the trigger.

- For the Response Payload, select the **Code Sample** option and enter the following code: `{ }`

  Note: The empty braces indicate that you want a response, otherwise it returns null.

3. Click **Save and Close**.

Create a Fixed List with User Name and Email Address

1. Create a fixed choice list.
   - Expand **Standard Objects > Proposal > Fields**.
   - Click the **Create a Custom Field** icon.
   - Select the **Choice List** (Fixed) option and click **OK**.
     - Enter a name for the Display Label: Set Project Manager.
     - In List of Values, click the **Create a New Lookup Type** icon.
       - **Meaning**: Set Project Manager
       - **Lookup Type**: PROJECT_MANAGER_01
       - Click the Create Lookup Code icon three times to create three rows.
   - The following table lists the lookup codes. You need to replace values in the table with actual values pertaining to your scenario.

Meaning	Lookup Code	Enable	Display Sequence
Martin Jones	martin.jones@example.com	Check	3
Sara Smith	sara.smith@example.com	Check	2
Alex Abraham	alex.abraham@example.com	Check	1

   - Click Save.
   - Click **Save and Close** to complete the new field.

2. Now, create a page layout and add the Set Project Manager field.
    - Click **Proposal > Pages**.
    - In the Details Page Layout region, click the **Duplicate** icon.
    - Enter a name and click **Save and Edit**.
    - Click the **Edit** icon (pencil) in the Edit Summary Subtab region.
    - Move the Set Project Manager field to the Selected Fields region.
    - Click **Save and Close**.

Configure Tabs and Subtabs

In this example, you can configure the tabs and subtabs of a proposal based on user roles in a page layout.

Scenario

To configure tabs or fields:

1. Navigate to Application Composer.

    Note: You must have activated the Sandbox.

2. Select **ERP and SCM Cloud** from the Application list.
3. In the Objects node, expand **Standard Objects**. In Standard Objects, from the Proposal node, click the **Pages** link.
4. Click the **Duplicate Details Page Layout** icon in the Details Page Layouts section.
5. Enter a name for this new layout, and select **Standard Layout** as the Source Layout.
6. Click **Save and Edit**.
7. To set a default subtab:
    a. Click the **Edit** icon next to the **No default subtab configured** text of the Subtabs region.
    b. In the Edit Default Subtab dialog box, select **Details** as the Default Subtab.
    c. Click **OK**.
    d. The Subtabs Region now displays the Default Subtab as Details.
8. To hide subtabs, in the Subtabs region, click the **Hide, Show or Reorder subtabs** icon.
    a. In the Configure Subtabs dialog box that appears, move Cost, Resources, and Revenue to the Available Subtabs panel.
    b. Click **OK**.

    The Proposals page doesn't display the Cost, Resources and Revenue subtabs.
9. To configure the fields in the subtabs, click the **Edit** icon in the Edit Subtab : Summary panel.
    a. In the Details Layout: <name_layout>: Edit Summary, in the Configure Detail Form area, select the fields - Currency and Business Objectives from the Selected Fields panel, and move it to the Available Fields panel.
    b. Click **Save and Close**.
    c. Click **Done**.

The Proposals page doesn't display the Currency and Business Objectives in the fields.

Validating the New Configuration

Here's how you can test the tabs and subtabs are hidden in the proposals:

1. Navigate to the Concept Design work area (**Navigator** > **Product Management** > **Concept Design**).
2. Click the **Search** icon.
3. Search for Proposals. In the Manage Proposals page that appears, click any proposal. You can see that the tabs and subtabs that you had hidden aren't displayed in the General Information tab of the Edit Proposal page.

In this example, the configuring of tabs and fields is very useful when you have to expose only certain fields to the end user based on their job description.

Hide Custom Subtabs in Supplier Portal

Use the isSupplier() method to control the visibility of the newly added tabs and layouts during runtime. Using this method, you can write Groovy Expressions in custom page layouts for change objects to hide or show custom subtabs to suppliers.

To limit a page layout so that only supplier users can see and use it, use the following as an advanced expression for the layout:

```
return this.isSupplier
```

To set a condition so that a button or action only works if the user is a supplier, use the following expression:

```
{if {this.isSupplier()} {
// do something
}
```

Use Global Function to Set Change Order Exit Criteria

You can use global functions to write groovy scripts that check the exit criteria for object types. In this case, the global function checks if affected item in change order has an Approved Manufacturer List (AML) associated with it.

In this example, we create a function to define the exit criteria in the approval step of the change order. The change order is approved only if an AML is added to each affected item. Defining the exit criteria in the function involves the following four steps:

1. Create two web services - GetAffectedItem, and GetAIAML.
2. Create global function - hasAML.
3. Create item rule.
4. Configure exit criteria for change order.

Create Web Services

In this example, let's create two Web Services - GetAffectedItem, and GetAIAML.

1. Navigate to Application Composer.

   Note: Ensure that you're in a publishable sandbox.

2. Select **ERP and SCM Cloud** from the Application list.
3. Select **Web Services** from the Common Setup pane.
4. In the Web Services page, click **Create a new Web Service reference that can be accessed within object scripts**.
5. Select REST from the Select Connection Type dialog.
6. In the Create REST Web Service Connection, enter the name as GetAffectedItem. Enter the URL: `https://<hostname>/fscmRestApi/resources/11.13.18.05/productChangeOrders/##changeId##/child/AffectedObject`
7. Select the option - **Propagate user identity using SAML** as the Authentication Scheme.
    a. In the Select and configure Methods against the Resource area, select the **GET** check box.
    b. Set the Request Payload to **Schema URL**.
    c. Set the Response Payload to **Code Sample**, and enter the code `{}`.
8. Click **Save and Close**.
9. Similarly, create another Web Service - GetAIAML with the URL: `https://<hostname>//fscmRestApi/resources/11.13.18.05/productChangeOrders/##changeId##/child/AffectedObject/##itemId##/child/AffectedItemAML`.

Create Global Function

Use Global Function in the Common Setup area to create a function that will check if the affected item in the change order has an AML.

1. In Application Composer, navigate to the **Common Setup** pane.
2. Click **Global Functions**.
3. In the Global Functions page, click **Add a Global Function** button.
4. In the Create Global Function page, enter hasAML as the function name. Set the Returns to **Boolean**.
5. In the Parameters area, click **Add Parameter**.
6. Enter the name as **changeId** and set the type to **String**.
7. In the Edit Script area, enter the following script:

```
def itemId = ""

def rtrnVal = true

changeId = String.valueOf(changeId)

def aitems = adf.webServices.GetAffectedItem.GET(changeId)

def ailnk = aitems["items"]

for(affectedItem in ailnk){

def links = affectedItem["links"]

for(lnk in links){

if(lnk["rel"].toString() =="self" && lnk["name"].toString()=="AffectedObject"){
```

//Extracting Item Unique ID by replacing the below url with blank

itemId = String.valueOf(lnk["href"].toString().replace(https://<host>:443/fscmRestApi/resources/11.13.18.05/productChangeOrders/+changeId+"/child/AffectedObject/",""))

```
def attResult = adf.webServices.GetAIAML.GET(changeId, itemId)

def aml = attResult["count"]

if(count==0){

rtrnVal = false

}}}}

return rtrnVal
```

8. Click **Validate**. You get the message that the script was parsed successfully.
9. Click **Save and Close**.

After creating the global function, publish the sandbox.

Create Item Rule

Create a rule and associate it to a change order type.

1. In the **Setup and Maintenance** menu, click the **Product Management** offering.
2. Click **Product Rules**.
3. In the Product Rules page, click **Manage Item Rule Sets** from Task.
4. In the Manage Rule Sets page, click **Create**.
5. In the Create Rule Set page, enter the following values:
    - Enter Display Name as hasAML.
    - Select **Validations** from the Type list.
    - Set the Association Type to **Change order type**.
    - Set the Association Name to **Engineering Change Order** or any other change order type that you want.
6. Click **Save and Continue** in the Create Rule Set page.
7. In the Edit Rule Set: hasAML page, click **Create**.
8. In the Create Rule page, set the following values:
    - Sequence=10
    - Name=Check AML
    - Set the Severity to Reject.
    - Select the check box **Stop further processing when rejected**.
9. Click **OK** in the Create Rule page.
10. In the Check AML: Details area of the General Information page, enter the following values:
    - IF Expression == true
    - Validation Condition == InvokeGlobalFunction("hasAML", [ChangeHeader].[Change Header Main].[Change ID])
    - User Message: You must add an AML to the item.
11. Click **Validate**. You get the message that the **Rule Check AML is valid**.
12. Click **Save and Close**. The hasAML rule is available in the Manage Rule Sets page.
13. Click **Done**.

Configure Exit Criteria for Change Order

Now, we configure an exit criteria for the change order to prevent status change if the hasAML rule is not met.

1. In the **Setup and Maintenance** menu, click the **Product Management** offering.
2. Click **Manage Change Order Types**.
3. In the Manage Change Order Types page, select the change type that you set in the rule, and click the **Edit** button.
4. In the Edit Change Order Type, select **Workflow** tab. Select a status such as Open. Click the Exit Criteria list, and you can see the rule that you created.
5. Set the Exit Criteria to hasAML.
6. Click **Save and Close**.
7. Click **Done** in the Manage Change Order Types page.
8. Navigate to **Product Management** > **Product Development**.
9. Select **Tasks** > **Create Item**.
10. In Create Item page, set the following:
    - Class = Root Item Class
    - Name = XJ500
    - Description = Radio
11. Click **Save and Close**.
12. In Edit Item: <item_name> page, select the AML tab, and note that it's empty. Now, select **Actions** > **Assign to Change Order**.
13. In the Assign to Change Order page, set the following values:
    - Change Type = Engineering Change Order (or the change order type where you assigned the rule as the exit criteria.)
    - Number = ECO500
    - Name = Update XJ500
14. Click **Save and Edit**.
15. Change the status of the change order from draft to open.
16. Click **Save**.
17. In the Edit Change Order page, change the status of the change order from Open to Approval.
18. You get the error message that you set in the Create Item Rule Setup step - **You must add an AML to the item**, and that the affected item has failed a validation related to a rule.

Update Attributes Using REST Web Services

Let's look at an example which uses REST web service to get the ID of a configured object and update the field.

Let's say that you have created an object entitled Research Log that records information from product research, and relates it to the Idea object through a Dynamic List. You then create a groovy expression that updates the Reference field on a Research Log, based on what log is selected in the Idea. The groovy expression uses web services to get the ID of the Research Log and update the field using the PATCH method.

REST Web Service APIs are automatically generated when you create a new custom object. All the attributes and properties of the custom object service are available in the Describe URL field. To find the service URL, click the **Custom Objects** link in the list of custom objects.

> **Note:** Ensure that you're in a sandbox to work with custom objects and REST APIs.

Here are some of the tasks that you must perform to update attributes using REST APIs.

Tasks Summary

1. Add the configured object to a standard object (such as Ideas).
2. Connect to the web service.
3. Update attributes using REST web services.

Add the Configured Object to a Standard Object

1. After creating a custom object called Research Log, navigate to **Standard Objects > Idea** and select the **Fields** node.
2. Click the **Create** icon, select **Choice List (Dynamic)**, and then click **OK**.
3. Name the Dynamic List as **Key Research** and then click **Next**.
4. Select **Research Log** as the Related Object and make sure that **Log Title** is set as the List Selection Display Value.
5. Click **Submit**.
6. Click the **Create** icon again, select **Text**, and then click **OK** .
7. Name the Text field: **Update Reference**.
8. Click **Save and Close**.
9. Select the **Pages** node in Idea and click the **Default custom layout** from Details Page Layouts (if there's no default layout, then create one).
10. In the Edit Subtab: Summary region, click the **Edit** icon and move **Key Research** and **Update Reference** to Selected Fields.
11. Click **Save and Close**.
12. Click **Navigator > Others > Research Log** to test the custom object.
13. Create two Research Logs.
14. Now to test the dynamic list in Ideas, click **Navigator > Product Management > Ideas**.
15. Click the **Manage Ideas** tab.
16. Click the link to any Idea and select a Research Log in the Key Research field.
17. Click **Save and Close**.

Connect to the Web Service

1. Return to your Sandbox and select **ERP and SCM Cloud**.
2. Click the **Web Services** node in Common Setup.
3. Click the **Create** icon, select **REST**, and then click **OK**.
4. In the Name field enter: **Get_ResearchLog**.
5. Enter the following host information for the URL:

```
https://<host>/fscmRestApi/resources/latest/ResearchNotesXX_c/##recordId##
```

6. Select **Call with basic authentication** and either use a Credential Key that you already created or create one with your user name and password.

7. Select **GET** and make sure that the Response Payload is set to **Code Sample** with the following as the code sample: { }
8. Click the **Create** icon again, select **REST**, and then click **OK**.
9. In the Name field enter: Patch_ResearchLog
10. For the URL enter: `https://<host>/fscmRestApi/resources/latest/ResearchLogXX_c/##recordId##`
11. Select **Call with basic authentication** and use the Credential Key that you already created.
12. Select PATCH and set the Request Payload to Code Sample with the following code sample: {Reference_c: "UpdateReference_c"}
13. Make sure that the Response Payload is set to Code Sample with the following as the code sample: { }
14. Click **Save and Close**.

Update Attributes of the Configured Objects Using REST Web Services

Use REST APIs calls in Groovy expressions to update attributes between:

- Standard and Custom Objects
- Custom and Custom Objects
- Standard and Standard Objects

1. Click the **Actions and Links** node in Idea and then click the **Create** icon.
2. In the Display Label field enter: **Update Reference Project**.
3. Click the Plus button in the Script region.
4. In the Function Name field enter **updateReferenceViaREST**.
5. Use Expression Builder to create the following Groovy Expression based on your API references:

```
def recordName = KeyResearch_c
def recordId
def response = adf.webServices.Get_ResearchLog.GET(recordName)
def responseItems = response["items"]
for (att in responseItems){
recordId = att["Id"].toString()
}
def data = [Reference_c: UpdateReference_c]
def url = adf.webServices.Patch_ResearchLog.PATCH(recordId, data)
```

6. Select your new expression as the Method Name for the Script and then click **Save**.
7. Now click the **Pages** link from Idea.
8. Click **Default page layout** from Details Page Layouts and click the **Edit** icon next to **Actions**.
9. Move Update Reference Project to the Selected Buttons column.
10. Click **Save and Close**.

Configure Fields to Show Data from External Sites

In this example, we will configure a field to receive and display data from an external repository, such as a bug database, using a web service.

Here are the tasks that we will perform:

Tasks Summary

1. Connect to an external repository using a web service.

2. Create a field.
3. Add an action to update the field.
4. Validate the configuration.

Connect to an External Repository Using a Web Service

Identify the correct external site and create a connection using a web service.

1. Navigate to Application Composer work area and select **ERP and SCM Cloud**.
2. To create a connection, click the **Web Services** link in the Common Setup area of Application Composer.
3. Click the **Create Web Service Reference** icon.
4. Select **REST** and click **OK**.
5. Enter a Name such as Bug Status.
6. Enter the URL to a REST web service. For example: bug.<xyzcomp>.com
7. For Authentication Scheme, if the web service doesn't require authentication, then click **None**, otherwise, setup authentication.
8. Select **GET** and leave the Format as **JSON**.
9. Leave the Request Payload blank.
10. For the Response Payload, select **Code Sample** and enter: { }

    The empty curly braces signifies that you want it to return data. Otherwise, it returns null.
11. Click **Save and Close**.

Create a Field

Create a field to receive the data from an external site.

1. In Application Composer, expand **Standard Objects > Idea** (or another object as required) and then click the **Fields** link.
2. Click the **Create** icon.
3. Select **Long Text** and click **OK**.
4. For the Display Label, enter **Update Status**.
5. Click **Save and Close**.

Add an Action to Update the Field

To call the web service, create an action and add it to the page layout.

Note: It can be a button or an action menu item.

1. Click the **Actions and Links** link in the Idea node.
2. Click the **Create** icon.
3. In the Display Label field, enter a name such as **Update Bug Status**.

   You can call a web service connection from any property that uses Expression Builder like Required field property, Updatable field property, Default Value Expression property, Layout Advanced Expression, Actions, Validations, and Triggers. Use Expression Builder to help you identify the correct web service and function you need to call to update the field.
4. Click the **New** (+) icon in the Script region.
5. In the Function Name field, enter **Bug Status**.
6. Enter the following script:

```
def response = adf.webServices.BugStatus.GET()
def properties = response["properties"]
```

```
def periods = properties["periods"]
for (att in periods){
if (att["name"] == "Today"){
setAttribute("Bug_Status", att["detailedStatus"])
}
}
```

> **Note:** This is a non-working example. Each system has a unique or different structure for the REST web service calls.

7. Click **Save and Close**.
8. Select **Bug Status** as the Method Name and then click **Save**.

Follow these steps to add the action to a page layout.

9. Click the **Pages** link.
10. Click the **Default custom layout** link.
11. Click the **Edit** (Pencil) icon next to Actions.
12. Move **Update Bug Status** to the Selected Buttons column.
13. Click **Save and Close** and then click **Done**.

Validate the Configuration

1. Navigate to **Product Management** and select **Ideas**.
2. Click the **Manage Ideas** tab.
3. Click the name of an Idea.
4. Click **Update Bug Status** button to view the updated status of the field.

Create a Record Field

In this example, use a Record Type field to trigger the display of a specific page layout.

In this example, create two fields called Voltage and Weight. Create two pages called Electrical Layout and Mechanical Layout. Create a record field and associate multiple values to it. Select a list item from the Record Type field, and that selection triggers a specific page layout.

1. Navigate to Application Composer (**Sandbox bar** > **Tools** > **Application Composer**).
2. Select **ERP and SCM Cloud** from the Application list.
3. Select the Quality object tag.
4. Expand the Quality Issue node, and click **Fields**.

   Create two fields - Voltage and Weight.

   a. In the Fields page, click **Create a custom field** icon.
   b. In the Select Field Type page, select the **Number** field type and click **OK**.
   c. In the Create Number Field page, enter **Voltage** as Display Label and enter the number of characters in the Display Width field.
   d. Click **Save and Close**.

   Similarly create a Weight number type field.
5. Add a record type field called Product Type.

a. In the Quality Issue node, click **Fields** link.
b. In the Fields page, click the **Create a custom field** icon.
c. In the Select Field Type page, click the **Record Type** field and click **OK**.
d. In the Create Record Type field page, in the Appearance area, enter **Product Type** for Display Label field.
e. In the same page, scroll to the List of Values area, and click the **Create a New Lookup Type** icon.
f. In the Create Lookup Type page, enter **Prod Type** in the Meaning field, and **PROD_TYPE** in the Lookup Type field.
g. In the same page, in the Lookup Codes section, click the **Create Lookup Code** icon.
h. Enter **Electrical** in the Meaning column, **ELECTRICAL** in the Lookup Code column, and **10** in the Display Sequence column.
i. Again, click the **Create Lookup Code** icon, and enter **Mechanical** in the Meaning column, **MECHANICAL** in the Lookup Code column, and **20** in the Display Sequence column.
j. Click **Save**.
k. In the Create Record Type Field, click **Save and Close**.

6. Click the **Pages** link in the Quality Issue node.
7. To add the Product Type field to the Record Type layout, do the following:

a. In the Quality Issue: Pages user interface, scroll to the Details Page Layouts area, and click **Duplicate Details Page Layout** icon.
b. In the Duplicate Layout dialog that appears, enter **Record Type** as the new Layout Name.
c. Click **Save and Edit**.
d. In the Details Layout: Record Type page, click the **Edit** icon in the Edit Subtab: Summary of the General Information tab.
e. In the Details Layout: Record Type: Edit Summary page, move Product Type from the Available Fields column to the Selected Fields column.
f. Click **Save and Close**.
g. Click **Done**.

8. Create a layout called Electrical Layout and add the Voltage text field.

a. In the Quality Issue: Pages user interface, scroll to the Details Page Layouts area, and click **Duplicate Details Page Layout** icon.
b. In the Duplicate Layout dialog that appears, enter **Electrical Layout** as the new Layout Name.
c. In the Source Layout field, select **Record Type**.
d. Click **Save and Edit**.
e. In the Details Layout: Electrical Layout page, click the **Edit** icon in the Edit Subtab: Summary of the General Information tab.
f. In the Details Layout: Electrical Layout: Edit Summary page, move Voltage from the Available Fields column to the Selected Fields column.
g. Click **Save and Close**.
h. Click **Done**.

9. Similarly, create a layout called Mechanical Layout and add the Weight text field. Again, while creating the page layout, select **Record Type** as the Source Layout
10. Set the record type for the specific layouts.

a. In the Quality Issue: Pages, scroll to the Details Page Layouts, click the **Available Record Types** in the Type column of the Mechanical Layout row.
b. In the Available Record Types dialog that appears, Select the **Specific Type** radio button.
c. Move **Mechanical** to Selected.

d. Click **OK**.
e. In the Quality Issue: Pages, scroll to the Details Page Layouts section, click the Available Record Types icon (arrow) in the Type column of the Electrical Layout row.
f. In the Available Record Types dialog that appears, select the **Specific Type** radio button.
g. Move **Electrical** to Selected.
h. Click **OK**.

Validating the New Fields

1. Use Navigator to open **Quality Management** (**Navigator** > **Supply Chain Execution** > **Quality Management**).
2. Click **Create** icon, and select **Quality Issue** from the list.
3. In the Create Quality Issue page that appears, select the **Problem Report - Item** as the Type.
4. Enter **Product Issue** as the Name.
5. Set an appropriate organization.
6. Click **Save and Close**.
7. In the Problem Report - Item: Product Issue page, notice that a Product Type field is displayed.
8. Select Electrical from the list in Product Type. The page layout with field Voltage is displayed.
9. In case you select Mechanical from the Product Type list, the page layout with field Weight is displayed.

Create a Conditional Layout

In this example, you will create a new conditional field that will become available only if there's a product enhancement, and a particular customer has to be informed about it.

Create a Conditional Field

Create a field called CAD Operator that's displayed only if the Idea type is Product Enhancement and DA Software is the customer. Next, add the CAD Operator field to a new layout. Then, create the condition expression such that the layout with the CAD Operator field is displayed only when the condition is met.

1. In Application Composer, select **ERP and SCM Cloud** from Application.
2. Select **Innovation** in the Object Tags.
3. Expand the **Standard Objects** node, and the **Idea** nodes.
4. To create a new field, do the following:
    a. Click **Fields** link in the Idea node.
    b. In the Fields page, click the **Create a custom field** icon.
    c. In the Select Field Type page, select the **Text** type, and click **OK**.
    d. In the Create Text Field page, enter **CAD Operator** for Display Label field, and enter the number of characters as 50 in the Display width field.
    e. Click **Save and Close**.
5. To create a new layout, do the following:
    a. Click the **Pages** link in the Idea node.
    b. In the Idea: Pages user interface that appears, scroll to the Details Page Layouts area and click the **Duplicate Details Page Layout** icon.
    c. In the Duplicate Layout dialog box that appears, enter **Conditional Layout** as the New Layout Name, and select **Standard Layout** as the Source Layout.

    d. Click **Save and Edit**.
6. After you click Save and Edit in the Duplicate Layout dialog box, the Details Layout: Conditional Layout page appears. In this page, you can add the newly created field to the Conditional Layout page. To add the CAD Operator field to the Summary section of the newly created Conditional Layout page, do the following:
    a. Click the **Edit** icon in the Edit Subtab : Summary region.
    b. In the Details Layout: Conditional Layout: Edit Summary page that appears, select **CAD Operator** in Available Fields, and click the button to move CAD Operator to Selected Fields.
    c. Click **Save and Close**
    d. Click **Done**.
7. In the Details Page Layouts area of the Idea: Pages user interface, in the Conditional Layout row, click the **XYZ** icon. Click this icon to enter the expression to determine which page layout has to be displayed.
8. In the Advanced Expression page that appears, in the Edit Script area, enter the following expression: `(Type == "Product_Enhancement") && (Customers == "DA Software").`
9. Click the **Validate** icon. If the expression is correct, a message is displayed: Script parsed successfully.
10. Click **OK**.
11. Test the condition layout expression.
    a. Navigate to Ideas (**Navigator** > **Product Management** > **Ideas**).
    b. Click the **Post Idea** button.
    c. Enter the following in the Idea field: Drawing Re-modeling.
    d. Select **Product Enhancement** as the Type.
    e. Click **Save and Close**.
    f. Click the plus (+) icon next to the Customers field.
    g. In the Associate Customers dialog box that appears, select **DA Software** from the Available Customers list to the Selected Customers panel.
    h. Click **OK**.

    Note that the CAD Operator field appears in the Conditional Layout page.

Set a Field Condition

In this example, the Occurrence field in Application Composer is used to check how often a quality issue occurs. Let's create a field in Application Composer. Set a property to this field such that it's updated whenever a quality issue occurs.

Create a new percentage field called Probability and add an expression for the Updatable property that activates the field for use if the Occurrence field is set to High. Here's how you can do it.

1. Navigate to Application Composer (**Sandbox bar** > **Tools** > **Application Composer**).
2. Select **ERP and SCM Cloud** from the Application list.
3. In the **Quality Object** tag, select the **Quality Issue** node.
4. Let's create a field.
    a. Expand the Quality Issue node, and click the **Fields** link.
    b. In the Fields page that appears, click the **Create a custom field** icon.
    c. In the Select Field Type page that appears, select the **Percentage** field and click **OK**.

   d. In the Create Percentage Field page that appears, name the field as **Probability**.
   e. In the Constraints area of the same page, click the **XYZ** icon next to the **Updatable** property.
   f. In the Edit Script area of the Configure Expression page, enter the following script:
   ```
   LikelihoodOfRecurrence == "ORA_LIKE_RECUR_HIGH"
   ```
   g. Click **OK**.
   h. Click **Save and Close**.
5. Click the **Pages** link in the Quality Issue tag.
6. In the Quality Issue: Pages user interface that appears, click the **Duplicate Details Page Layout** icon in the Details Page Layouts area.
7. Enter **Field Condition** for name in the Duplicate Layout dialog. Select **Standard Layout** as the Source layout.
8. Click **Save and Edit**.
9. Now, include the Probability field to the Summary Subtab by doing the following:
   a. Click the **Edit** icon in the Summary Subtab region.
   b. Select **Probability** and move it to the Selected fields.
   c. Click **Save and Close**.
10. Click **Done**.

Here's how you can test the scenario:

1. Navigate to the Quality Management work area (**Navigator** > **Supply Chain Execution** > **Quality Management**).
2. Click the **Create** icon and select **Quality Issue**.
3. Select **Problem Report - Item** as the Type.
4. Enter **Power Failure** as the Name.
5. Set an appropriate organization.
6. Leave the Severity and Source at their default values.
7. Click **Save and Close**. Note that you can't enter a value in the **Probability** field.
8. Select **High** for the Occurrence field.
9. Click **Save**. Now, notice that you can enter a value in the Probability field.

Configure a Formula Field

This example illustrates how to build an expression that represents a formula used to calculate values. In this example, a multiplication function called NVL () is used to calculate the total cost of the products.

In this example, we will build a simple expression for calculating the total cost for a number of products given the cost of a single product. To do this, create a field to enter the cost of one product and another to enter the number of products. In addition, create a formula field to display the result of calculating the total cost of the products. An Expression Builder is used to create the formula, using the attributes in the newly created custom fields.

> **Note:** Concept structures don't support formula fields.

To calculate the formulas, do the following:

1. Launch Application Composer.

2. Select **ERP and SCM Cloud** from the Application list.
3. Select the **Innovation** object tag.
4. In Standard Objects, expand the **Idea** node. Here you add two fields in which you can enter the values and one formula field, in which the result of the expression is displayed.
    a. Expand the Idea node, and click the **Fields** link.
    b. In the Fields page that appears, click the **Create a custom field** icon.
    c. In the Select Field Type page that appears, select the type of field you want, in this example, select **Number** and click **OK**.
    d. In the Create Number Field page that appears, enter a name such as Price in the Display Label field and enter the number of characters in the Display Width field.

    > **Note:** In case you want the field to be conditional you can set the conditions in the Constraints area of the Create Number Field page.

    e. After creating the field, click **Save and Close**.
    f. Similarly, create another field called Quantity.
5. To create a formula field, do the following:
    a. Click the **Create a custom field** icon.
    b. In the Select Field Type, select **Formula** and click **OK**.
    c. In the Create Formula Field : Describe Field page that appears, select the Number data type for this example.
    d. Enter Total Amount as the Display Label, and a number for Display Width to indicate the number of characters the field can hold.
    e. In the Constraints area, in the **Depends on** field select Price and Quantity from the list to indicate that this field value is dependent on Price and Quantity.
    f. In the Create Formula Field : Configure Expression, enter the formula to calculate the total amount.
    g. Click the **Show/Hide Expression Palette** icon, click **Fields** tab.
    h. The two newly created fields - Price and Quantity are displayed in the Fields area of the Idea page. To, build the expression, select the API Name. The custom fields are indicated with an underscore "c" added to the end of the field name. In this example, the API Name for Price is Price_c.
    i. For this example, enter the following function in the Edit Script field:`nvl (Price_c, 0.00)*nvl (Quantity_c, 0.00)`

    > **Note:** `nvl` is a function that compares the values in the parenthesis. If the first value is null then the second value is used in the equation. Using this function prevents you from having a null value error in your equation results.

    j. Click the **Validate** icon to verify if the script is correct.
    k. Click **Submit**.
6. To create a Function Field layout to include the three new fields,
    a. Click **Pages** in the Idea tag. The Idea: Pages screen appears.
    b. Move to the Idea : Pages screen to the Details Page Layouts area, and click the **Duplicate Details Page Layout** icon.
    c. Enter a name for the new layout in the Duplicate Layout dialog. Let the Source Layout be Standard Layout.
    d. Click **Save and Edit**.
    e. In the Details Layout: Function Field page, click **Add** button present at the end of the page, and select **Field Groups**.

f. In The Details Layout: Function Field: Create Field Group page, you can configure the field group details. Enter the Display Label as Total Amount and click **Next**.
g. In the Add Fields page that appears, select the Price, Quantity, and Total Amount fields and move them to the Selected Fields list.
h. Click **Next**.

7. Click **Save and Close**.
8. Click **Done**.

To test the creation of the two fields, and the formula field, do the following:

1. Navigate to Ideas.
2. Click the **Manage Ideas** tab.
3. In the Manage Ideas page, click the row of any idea in the Idea panel.
4. The Edit Idea: <name_idea> displays the two newly created fields - Price and Quantity, and the Formula Field - Total Amount.
5. Enter a number value for the Price and Quantity.
6. Click the page to refresh it. The Total Amount displays the value as per the expression. In this example, the Total Amount field displays the product of the values entered in Price and Quantity fields.

Add a Validation Rule for a Field

You can create a validation rule using a groovy script to ensure that a particular field is filled in before an object is saved. You can also define the message that should be displayed if the rule is not met.

In this example, we create a field-level validation rule to ensure that the Priority field is filled in before a new change order is saved.

1. Navigate to Application Composer work area.

   > **Note:** Ensure that you're in a sandbox.

2. Select the ERP and SCM Cloud option from the Application list.
3. Select **Standard Objects** > **Change Order** > **Server Scripts**.
4. In the Server Scripts Change Order page, in the Validation Rules tab, click the **Add a new validation rule** button in Object Rules.
5. In the Create Object Validation Rule dialog, enter a name for the rule, Priority_Field_Validation_Rule.
6. Enter the error message you want to show when the condition is not met. “You must set a priority level.”
7. In the Rule Definition area, enter the following script:

```
if(PriorityCode == null){
return false;
}
else{
return true}
```

8. Click **Validate**. You get the message that the Script was parsed successfully.
9. Click **Save and Close**.

Test Rule

Let's test the rule in the Product Development application.

1. Navigate to **Product Management** > **Product Development**.

2. Create a new change order and save it without entering a value in the Priority field.

   You should see the error message that you configured.

Use Triggers to Update Descriptive Flexfields

Use field-level triggers to create a dependency between two fields, and ensure that the value entered in the first field is automatically added in the second field.

In this example, we add a trigger to create a dependency between two descriptive flexfields in the General Information tab of a change order. We have created and used descriptive flexfields - Implementation Reason, and PCN. When a value is added to the Implementation Reason field, the same value is automatically updated in the PCN field.

1. Navigate to Application Composer work area.

   > **Note:** Ensure that you're in a sandbox.

2. Select the **ERP and SCM Cloud** option from the Application list.
3. Select **Standard Objects** > **Change Order** > **Server Scripts**.
4. In the Server Scripts Change Order page, click the **Triggers** tab. Click the **Add a new trigger** button, and enter a name for the trigger.
5. In the Create Object Trigger page, you can select a trigger that is applicable to this scenario. For this example, we can choose the trigger from the list - **Before Update in Database** that is fired before an existing object is modified in the database.
6. In the Trigger Definition, enter the following script:

```
def changeDFF = ChangeObjectDFF
def reason = changeDFF.implementationReason
changeDFF.setAttribute('pcn', reason)
```

7. Click **Validate**. You get the message that the script was parsed successfully.
8. Click **Save and Close**.

Test the Trigger

Here's how you test this trigger in the Product Development application.

1. Navigate to **Product Management** > **Product Development**.
2. Open the change order that contains the two descriptive flexfields - Implementation Reason and PCN.
3. Enter a value in the Implementation Reason flex field. Click **Save**.

   You can see that the PCN field is automatically updated with the value entered in the Implementation Reason flexfield.

Use Expression Builder

In this example, use the Expression Builder to create a trigger that sets the Disposition field to Dispose as Scrap when the Occurrence field is set to High.

Scenario

To create a trigger using the Expression Builder

1. Navigate to Application Composer.
2. Select **ERP and SCM Cloud** from the Application list.
3. Click the **Quality** object tab and then expand the **Quality Issue** node.
4. Create a trigger stating that when Occurrence is set to High then Disposition will be set to Dispose as Scrap.
    a. Select the **Server Scripts** node.
    b. In the Server Scripts Quality Issue page, select the **Triggers** tab.
    c. In the Field Triggers region, click **Add a New Trigger** icon.
    d. In the Create Field Trigger page, select **Occurrence** in the list for the field name.
    e. Enter **OccurrenceHighDispositionScrap** as the Trigger Name.
    f. In the Trigger Definition area of the page, click the **Show/Hide Expression Palette** icon.
    g. Click the **Fields** tab.
    h. In the Quality Issue: Fields area, select **Occurrence** and click **Insert**.
    i. In the Edit Script area, enter the following script for the LikelhoodOfRecurrence: `if (LikelihoodOfRecurrence == "ORA_LIKE_RECUR_HIGH"){`
    j. Place the cursor on the next line, and then from the Quality Issue: Fields area, select **Disposition** and click **Insert**.
    k. The API_Name of Disposition appears in the Edit Script area. Enter the code around the API_Name: `setAttribute ('Disposition',"ORA_DISPOSE_AS_SCRAP")}`
5. Click the **Validate** icon. If the script is correct, you get the following message: *Script parsed successfully*. Else an error message is displayed showing where the error is.
6. Click **Save and Close**.

Test the Script

Here's how you test the script:

1. Navigate to Quality Management (**Navigator** > **Supply Chain Execution** > **Quality Management**).
2. Click the **Create** icon, and select **Quality Issue** from the list.
3. In the Create Quality Issue page that appears, do the following:
    a. In the Type field, select **Problem Report - Item** from the list.
    b. In the Name field, enter **Electrical Failure**.
    c. Select the appropriate value for Organization from the list.
    d. Set the Severity to 4 Low.
    e. Set the Source to **Manufacturing**.
    f. Click **Save and Close**.
4. In the Problem Report - Item: Electrical Failure page, General Information tab, set the Occurrence to **High**.
5. Click **Save**.

    Note that the Disposition field displays **Dispose As Scrap**.

Expression Builder is an easy way to create formulas/expressions/triggers. Expression Builder makes expressions that are easier to build and to read.

3 FAQs

What job role must I have to create my own objects in Application Composer?

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

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

In Oracle CX Sales, provision the user with the Customer Relationship Management Application Administrator job role (for performing the configurations) and the Custom Objects Administration job role and Sales Administrator job role (for testing the configurations).

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.

For a fixed choice list, you define the specific values that will appear and the list doesn't change.

A dynamic choice list, meanwhile, is populated from an existing object's actual data, which you select and can add filters to. Based on how you define the field, the list is dynamically populated at runtime and its values can change depending on the context.

What Application Composer tasks are available only within a sandbox?

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

- Objects
  - Custom Objects
  - Standard Objects
- Common Setup

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

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

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

Can two objects have the same record number?

Yes, two objects can have the same record number. The record number is unique only within a configured object.

How frequently can I publish a sandbox?

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

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

When do I publish a sandbox?

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

You must test the following configurations outside a sandbox:

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

Can I delete a sandbox?

Yes. You can delete sandboxes. However, you can delete only those that aren't published.

Before you delete a sandbox, you must first confirm that the sandbox isn't active.

CAUTION: Deletion of partial content of a sandbox is risky. It's recommended that you don't use this option.

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

Can I delete unused custom attributes?

Yes, you can delete any unwanted or unused custom attributes in a sandbox. You can only delete the custom attributes that have never been published.

Here's how you delete the custom attribute:

1. Navigate to Application Composer work area.

   Note: Ensure that you're in a sandbox.
2. Select the **ERP and SCM Cloud** option from the Application list.
3. Select the **Quality** check box in the Object tags.
4. Expand **Standard Objects** > **Quality Issue** > **Fields**.
5. Select the custom attribute and click the **Delete** button.

   In case the attribute is still in use, a message appears restricting you from deleting it. If the attribute is in use, review and remove all usages before you try again.
6. The unused custom attribute is deleted.