Oracle® Cloud

Extending Oracle Cloud Applications in Visual Builder Studio Express Mode

Release 25.07
G26868-01
April 2025

Oracle Cloud Extending Oracle Cloud Applications in Visual Builder Studio Express Mode, Release 25.07

G26868-01

Primary Author: Oracle Corporation

Contents

Preface

Extending Oracle Cloud Applications in Visual Builder Studio Express Mode describes how to use a web-based visual development tool to create and configure extensions to customize Oracle Cloud Applications in Express mode. This mode is available for a subset of Oracle Cloud Application pages, as determined by Oracle. If you don't see "Express" in the Designer's header,this mode is not available for the current page.

Topics:

Audience

This document is intended for Oracle Cloud Applications users who want to create, edit, and publish their extensions in Express mode..

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://support.oracle.com/portal/ or visit Oracle Accessibility Learning and Support if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud

  http://cloud.oracle.com

- Extending Oracle Cloud Applications with Visual Builder Studio (Developer Mode)

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

What Can You Do with Visual Builder Studio in Express Mode?

Visual Builder Studio (VB Studio) provides an easy, simple-to-use interface called *Express mode*, which helps you tailor your Oracle Cloud Applications pages so your end users are presented with only the data they need.

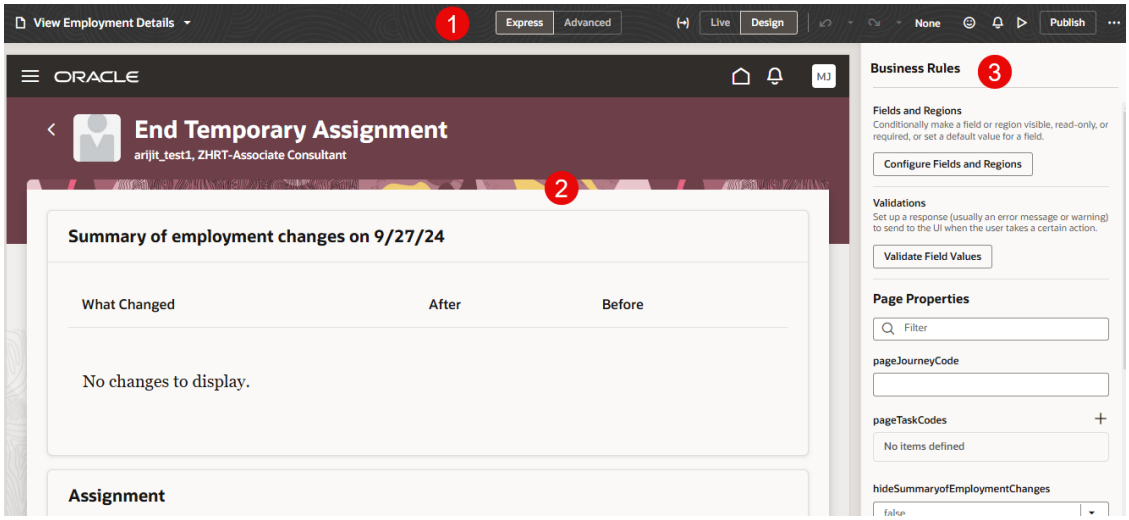
In fact, VB Studio provides two entirely different modes—Express and Advanced—so that you, the Oracle Cloud Applications functional administrator, can work in a curated design experience where you're not distracted by features you don't need, while hard-core developers can use Advanced mode to tackle more complex use cases. Toggling between the two modes is as simple as clicking a button in the header. If you ever encounter a scenario that you can't achieve in Express mode, you can enter Advanced mode to access everything that VB Studio has to offer. Advanced mode is described in Extending Oracle Cloud Applications with Visual Builder Studio.

> **Note:**
>
> If you're an HCM or SCM user, you may also want to refer to Overview of Redwood Application Extension in *Extending Redwood Applications for HCM and SCM Using Visual Builder Studio* for content specific to that Oracle Cloud Application.

While in VB Studio, you'll do most of your work in *the Designer*, VB Studio's built-in editor. Let's look at the Designer in a bit more detail.

What Is the Designer?

In Express mode, the Designer—which is essentially your working area—is divided into three main areas:

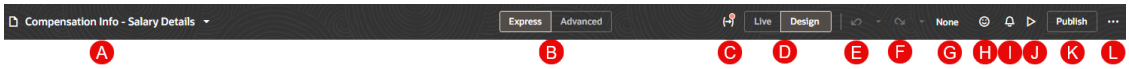


1. Header
2. Canvas/Editors
3. Properties pane

Read on to discover what's contained within each area.

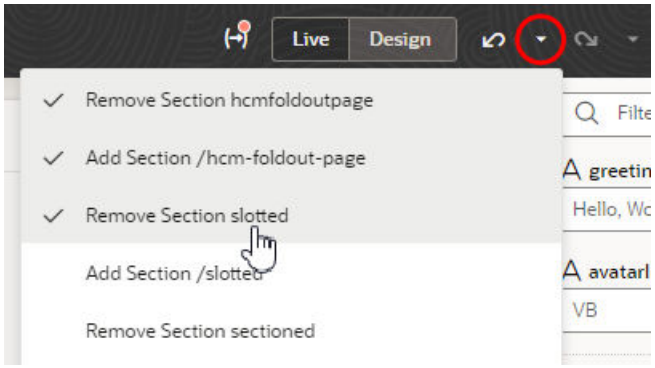
The Header

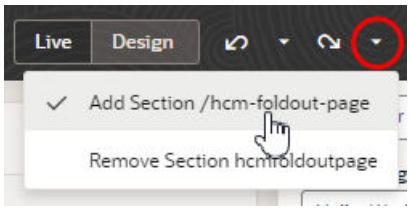
The header contains information about your current VB Studio session, and provides access to tools that help you share and publish your changes when you're ready.



Here's what each element does:

Label	Element	Description
A	Page title	Displays the title of the page open in the Designer. Click the page title to open a list of pages in the Navigator, which you use to locate and open Oracle Cloud Application pages.
B	Express / Advanced	Toggles between the full view of Visual Builder Studio (called "Advanced" mode) and Express mode. Express mode is tailored for the needs of Oracle Cloud Applications functional administrators.
C	Set Page Properties	Opens the Set Page Properties dialog box where you can set the page's input parameters, if any. When a page has a required parameter but a value isn't assigned to it, you'll see a red dot on the Set Page Properties icon in the header, like this:  You can click a property's help icon (if available) in the Set Page Properties dialog for guidance on how best to set the property's value.

Label	Element	Description
D	Live / Design	Toggles between Live and Design views. • **Live**: Provides a quick way of checking how your pages will look and behave when they are published without leaving the Designer. You can interact with pages in Live view, however, some behavior, such as how pages are rendered or how navigation links work in the app, might be different from the published app. To fully test your application, you should open it in Preview mode in a new browser tab, instead of using Live view. • **Design**: The view you use most frequently, to select components on the page.
E	Undo	Undo one or more of your changes. To undo your most recent change, click the **Undo** icon (hover your cursor over the icon to view the action that will be undone). To undo multiple changes, click the Undo drop-down list and select the actions you want to undo. For example, selecting the Remove Section slotted action in this image will re-add the section and undo the two other changes you made after removing the section:  **Tip:** You can undo up to 10 of your changes at a time (your last 500 actions are stored in the browser and will be lost if you clear the browsing cache). To undo more than 10 actions, simply undo a few items, then open the drop-down list again.

Label	Element	Description
F	Redo	Redo one or more changes after undoing them. To redo your most recent change, click the **Redo** icon (hover your cursor over the icon to view the action that will redone). To redo multiple changes, click the Redo drop-down list and select the actions you want to redo. For example, selecting the Remove Section hcmfoldoutpage action in this image will also revert the previously undone action:  **Tip:** You can redo up to 10 of your changes at a time (your last 500 actions are stored in the browser and will be lost if you clear the browsing cache). To redo more than 10 actions, simply redo a few items, then open the drop-down list again.
G	Sandbox	Opens a menu with options for working with a sandbox in your extension. The Switch Sandbox option opens a dialog box where you can select a sandbox. If a sandbox is selected, its name is displayed in the header, and you can select additional options in the menu: Refresh, Disassociate, and Publish. For more about these options, and associating sandboxes with Git branches, see How Do Sandboxes Relate to Git Branches? in *Extending Oracle Cloud Applications with Visual Builder Studio*. **Note:** You need to switch to Advanced mode if you want to work with Git repositories.
H	Feedback	Submit your feedback about VB Studio to Oracle.

Label	Element	Description
I	Notifications	View notifications received over the past two days. To get your attention, notifications pop up instantly in the bottom right corner of your window, where info, confirmation, and warning messages clear automatically after five seconds but error messages stay until they are manually cleared. If you've missed a notification or want to review one again, click **Notifications** in the header to view all notifications from the last two days. Notifications older than two days are automatically cleared, or you can click **Clear all** to clear them yourself. When you've got unread notifications, the Notifications icon is badged according to the type of notification received. Missed info, confirmation, and warning messages are badged with a blue dot  ; missed error messages are badged with a red dot  . Any time an error notification is unread, you'll see the red dot over Notifications—even if the last notification was of a different type—indicating that an error condition requires your action. You can act on notifications from within the Notifications panel just as you would when the notification pops up in the bottom corner of your window. You can also track the progress of tasks that are running in the background right from the notification.
J	Preview	Use the Preview action to test your app and see how your pages look and behave, including the navigation between pages and to external addresses. Preview represents how your app will behave when published more accurately than using the Designer's Live view. Preview opens your page in another browser tab. You may be asked to re-enter your credentials for the Oracle Cloud Applications instance before the tab opens.
K	Publish	After you're done testing your changes, use the Publish action to apply them to your development environment.
L	Menu	Open a menu containing the Share action, so you can make your changes available to others before publishing. The menu also provides options for navigating to the Visual Builder Studio Help Center and discussion forums, changing the Designer theme, and signing out.

Your changes are saved as you work, so there's no need to save them explicitly—but when collaborating with others, it's important to include their saved changes in your workspace, so you're not overwriting their work. To help you do this, VB Studio notifies you when other people save their changes (which is when their work is published). All you then need to do is click the **Refresh Workspace** link in the notification to pull in their updates. This way you keep your work up-to-date with everyone's changes and avoid changes that overlap or conflict with each other. See Refresh Your Source Files.

Canvas/Editors

When you open a page, it opens in the canvas area. When you're in Design mode, you can select configurable items on the canvas to view their details in the Properties pane. When you open the VB Studio editors, for example, the business rule editors, they open in this area, on top of the page on the canvas. In wide browser windows, you can see the page to the left of the editors.

Properties Pane

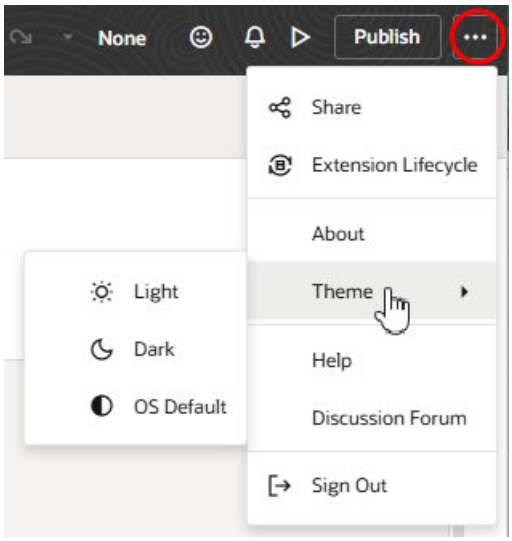
The content of the Properties pane changes, depending on how your page was configured. If your page was configured with business rules, you'll have access to the business rule editors from the Properties pane. For pages controlled by rule sets, a link to the rule sets editor is provided.

If an Oracle Cloud Application page has configurable containers or page properties, you can edit them from the Properties pane.

> **Note:**
>
> If the page title is a translation, the Properties pane displays the page's file name instead of the translated title.

Select the Designer Theme

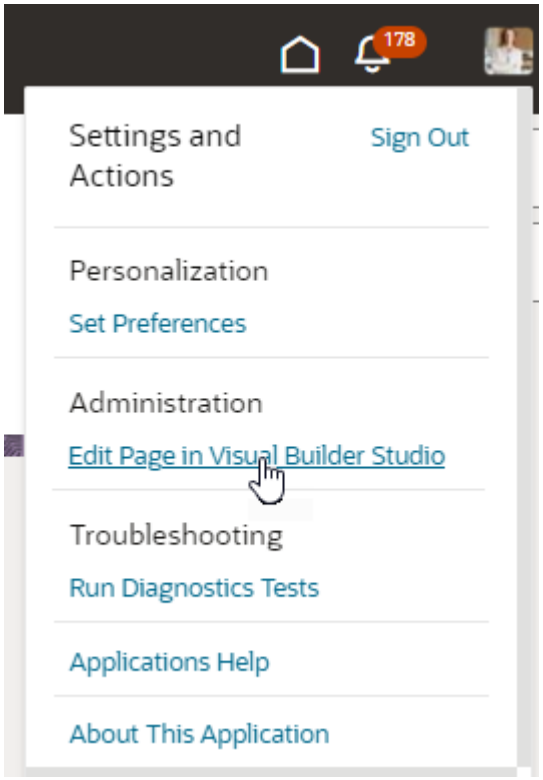
By default, the Designer uses a light theme to set the color palette for your work environment. To personalize your environment to use a dark theme or sync with your OS settings, click **Theme** from the menu in the header and choose an option:



- Select **Light** to use the default theme.
- Select **Dark** to use a dark color display, more suited for low-light conditions. This option switches the background and text used in all the editors, except the canvas, where application pages continue to display against a lighter background with dark text.
- Select **OS Default** to inherit the theme used in your operating system's settings. If your system settings are configured to use dark mode, the Designer also uses those settings.

Access Visual Builder Studio

The simplest way to enter VB Studio is to view the Oracle Cloud Applications page you want to configure, click **Settings** in the upper right corner, then click **Edit Page in Visual Builder Studio**:



Note:

If you don't see the **Edit Page in Visual Builder Studio** option in your Oracle Cloud Application, it could be because:

- You don't have the right privileges to access VB Studio. Check with your Oracle Cloud Application administrator if you're not sure.
- You may not be working in an environment that has an instance of VB Studio associated with it. Again, check with your Oracle Cloud Application administrator to see if this is the case.
- Your Oracle Cloud Application has not yet adopted Oracle's new Redwood design pattern, so this page is not extensible using VB Studio. In that case, refer to *Oracle Applications Cloud: Configuring and Extending Applications* for instructions on how to customize your Oracle Cloud Applications with App Composer to meet your business needs.

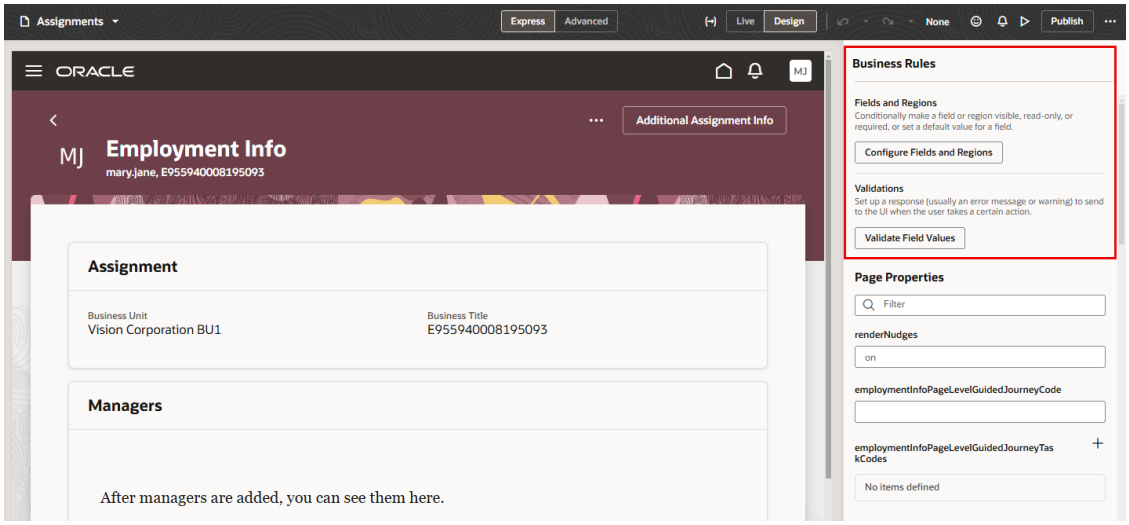
At this point you'll need to join a *project*, which provides all the infrastructure you need to change your Oracle Cloud Application and deploy it to a production environment.

- If you're the first person in your organization to edit an Oracle Cloud Application page—any page, regardless of the pillar it belongs to (HCM, SCM, etc.)—you'll become the owner of a new project called `Application Extensions`, which VB Studio creates for you. Thereafter, anyone who uses **Edit Page in Visual Builder Studio** must request access to this project and you, as project owner, will need to approve those requests. (If you don't want to serve as the project owner, you can transfer ownership by following the instructions here: Assign a Project's Ownership to Another User. A project must have at least one owner, although there can be more than one.)

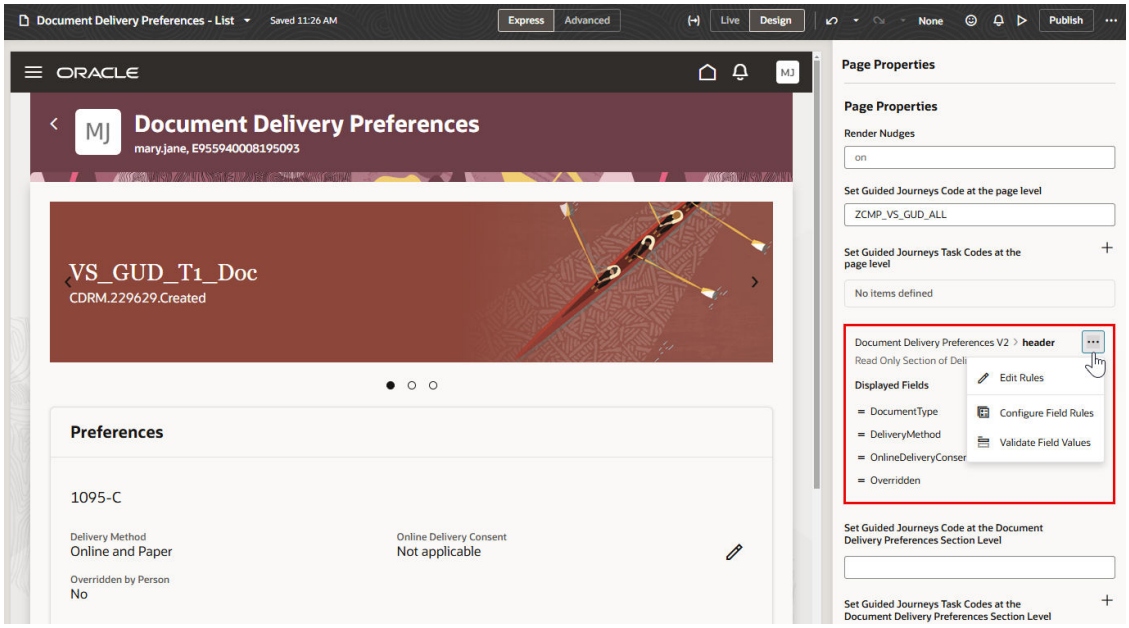
- If you're not the first person to edit an Oracle Cloud Application page, you'll have to ask to become a member of `Application Extensions` on the dialog that appears. You'll receive an email once the approval process is complete. When you do, go back to the page you want to edit and click **Edit Page in Visual Builder Studio** again.

You'll then land in the Designer, with all the tools you need at your disposal.

Depending on how your page was originally built by Oracle, your view of the Designer may differ. For example, if you see a Business Rules pane to the right of the page in the Designer, you'll use *business rules* to control the logic that determines what is displayed on the page at runtime. In the Business Rules pane, notice the link to Configure Fields and Regions, which gives you direct access to the business rules editor (see Control Your Display with Business Rules for information on working with this editor):



If, however, you don't see a Business Rules pane in the Designer, it means the page is not set up to use business rules. In this case, any dynamic forms and tables in the page are governed through *rule sets*. Like business rules, rule sets allow you to configure how a dynamic table or form appears at runtime through the use of rules and conditions. When the page is open in the Designer, the Properties pane on the right lists the rule sets used in the page instead of displaying the Business Rules pane:



In pages that use rule sets, in Express mode it's likely that your tasks will be limited to creating rules, as well as the layouts that are applied when the condition for the rules are satisfied. For more information on using rule sets in Express mode, see Control Your Display With Rule Sets.

Other Views of the Designer

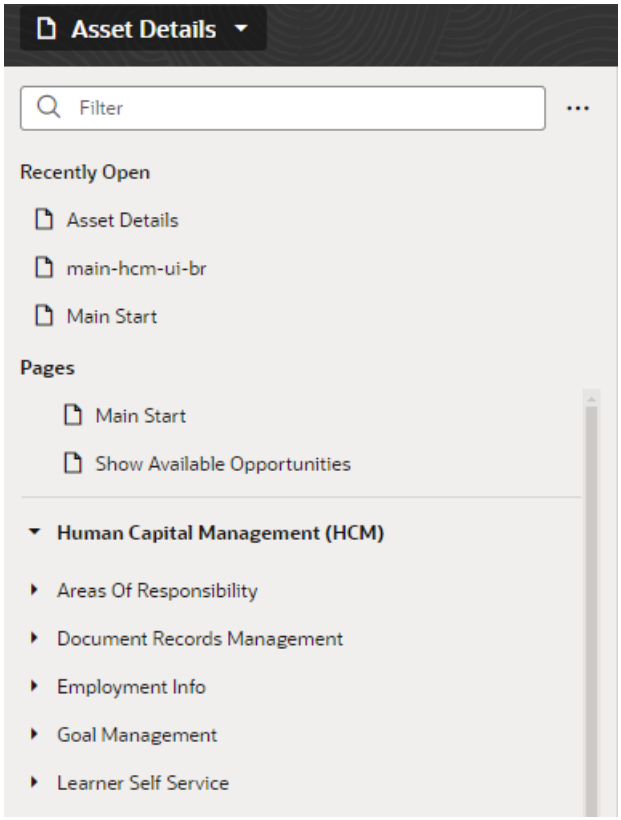
Depending on the page you're trying to edit, you may see something entirely different when you land in VB Studio than what's shown in the preceding topic.

For example, you may see a notice in the Properties pane that says your page is not extensible, which means that neither the business rules editors nor the rule sets editor are appropriate for this page. If you see this message, it's best to return to the browser tab with your Oracle Cloud Application and choose a different page to work on, assuming you have one.

However, if you know the name of the page you want to work on, you can open it directly from within the Designer using the Navigator:

- In the Designer, click the page name in the header to open the Navigator, then select the page in the list.

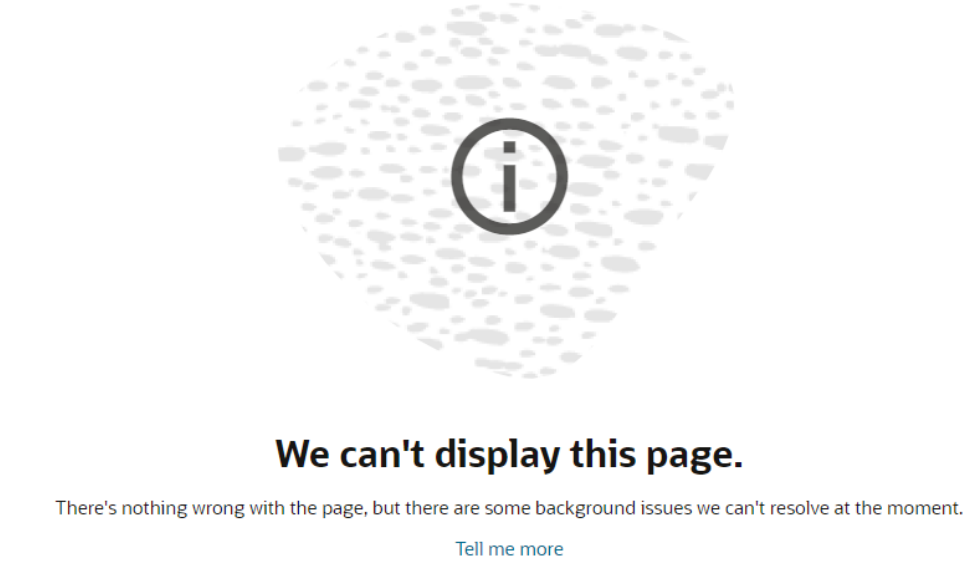
The Navigator lists all the pages in your Oracle Cloud Application instance that you can access. The list can be long, so you might try using the Filter field to help locate the page you want to work with. If you have created any pages in an App UI, they are listed under Pages, and recently opened pages are listed at the top of the Navigator:



Pages are listed in the Navigator by the title reflected on the screen, if available. If a page doesn't have a title, the list will show the page's *internal* name (like `activity-details`).

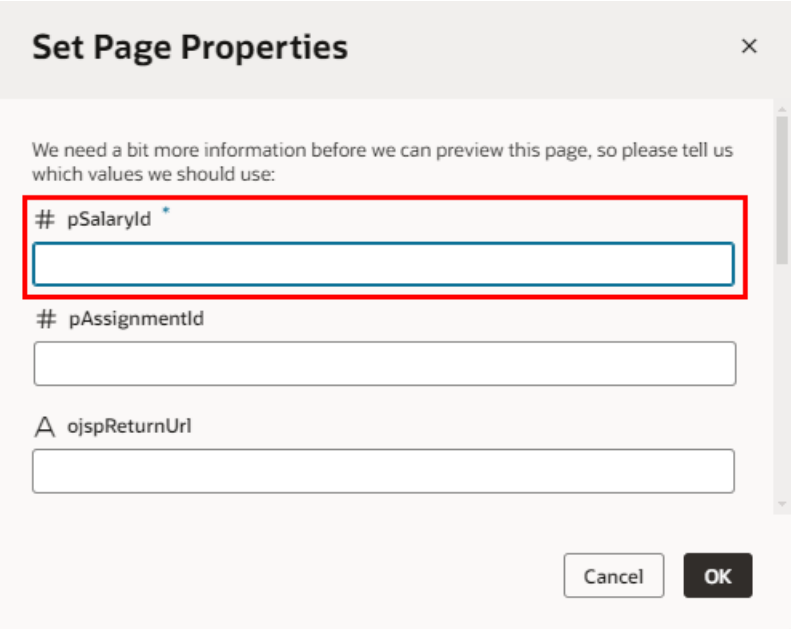
> **Note:**
>
> If you want the Navigator to always be visible in the Designer, open the menu next to the Filter field and select **Keep this panel visible**.

While it's possible to configure a page by choosing it directly in the Navigator—as opposed to starting in Oracle Cloud Applications and using **Edit Page in Visual Builder Studio**—this method may sometimes result in an error that looks like this:



This occurs when the page has required input parameters that can’t be set when you access the page directly from VB Studio. The best thing to do in this situation is return to your Oracle Cloud Application, find the page, then click **Edit Page in Visual Builder Studio**. This ensures that the page will be able to gather the required data for the input parameters and pass them to VB Studio during the transfer process, which enables the page to render correctly.

If you know the values for parameters required in the page, you can manually set the parameters in the Set Page Properties dialog box. When a page has a required parameter but a value isn't assigned to it, you'll see a red dot on the Set Page Properties icon in the header, like this:  . Click the icon to open the dialog box and set the missing parameters. Required parameters in the dialog are marked with an asterisk (*):


Set Page Properties
We need a bit more information before we can preview this page, so please tell us which values we should use:
pSalaryId *
pAssignmentId
ojspReturnUrl
Cancel
OK

2

Control Your Display with Business Rules

Whether you're a functional administrator tailoring your end users' interactions or an end user yourself, *business rules* enable you to create rules to control how regions and fields in components are displayed at runtime, based on conditions such as the user's role and the employee's business unit or legal employer.

When the dependency developer has enabled business rules for a page, you can easily control what's displayed on the page as a whole, instead of configuring page components such as tables and forms individually. For example, you could create a single rule to conditionally hide every occurrence of a field on the page, in every form where it appears. If you were to do this using rule sets, you would need to configure the rule set for each form to hide the field. Business rules can handle simple and complex display needs that might otherwise require tens, if not hundreds, of individual layouts if the rule set approach was used instead.

If you don't see the Business Rules pane in your view of the Properties pane, that means your page is not enabled for business rules. Instead, use the rule sets editor as described in Control Your Display With Rule Sets to configure your page. (While it's possible to use both the rule sets and business rules editors for the same page, this is an advanced technique that is not generally recommended.)

What Are Business Rules?

Business rules allow you to override the appearance and behavior of fields in dynamic tables and forms, provided that the specified conditions in a rule are met at runtime.

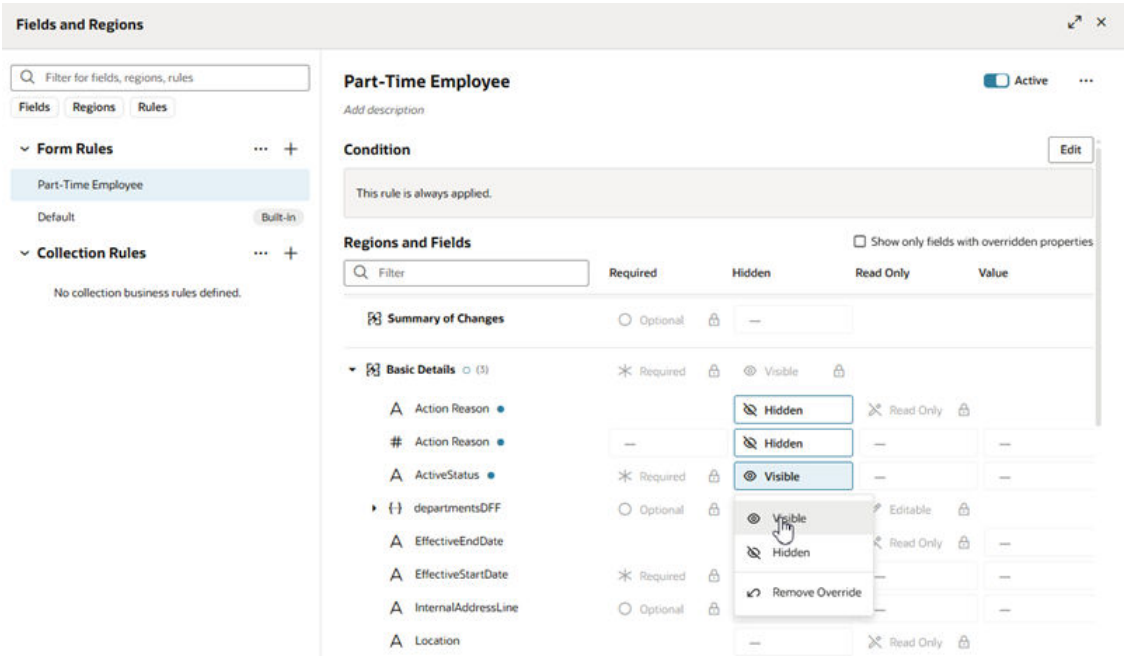
> **Note:**
>
> If you're looking for information on business rules as they pertain to business objects, see Create Rules for Business Objects.

> **Note:**
>
> In some applications, where you see fields grouped into *regions* in the business rules editor, a region is simply a dynamic table or form.

For example, suppose you had a data object called Person, that was used in two dynamic forms. Now suppose you want to hide the **Organization Name** field when the user is a manager, as only HR specialists should see this data. With business rules, you can create a single rule that defines the condition, then overrides the setting for the Hidden property to ensure that **Organization Name** is hidden when the user is a manager. With this one action, the **Organization Name** field will be hidden for every region that includes that field.

In other words, business rules let you define a rule at the *object* level. Let's look at a page with several dynamic forms. These are listed in the Regions and Fields section in the image below. Some components use the same data object, and you can quickly see every field used in each

component. When you set a property in a business rule you are setting it at the object level, so it can be applied to every occurrence of that field, in each form or table using that data object.



There are two types of rules: *form rules*, which control the properties of fields in dynamic forms, and *collection rules*, which determine the columns displayed in tables. Form and collection rules defined in a dependency are read-only, and you can identify them by their *Built-in* label visible in the list of rules. For more, see Create a Rule For Forms and Create a Rule For Tables.

Work with Business Rules

When configuring regions and fields, the main components of a business rule are *conditions* and *properties.* Conditions determine the circumstances under which the rule is applied, while properties override the value set for a given field or region by Oracle (or by your functional administrator).

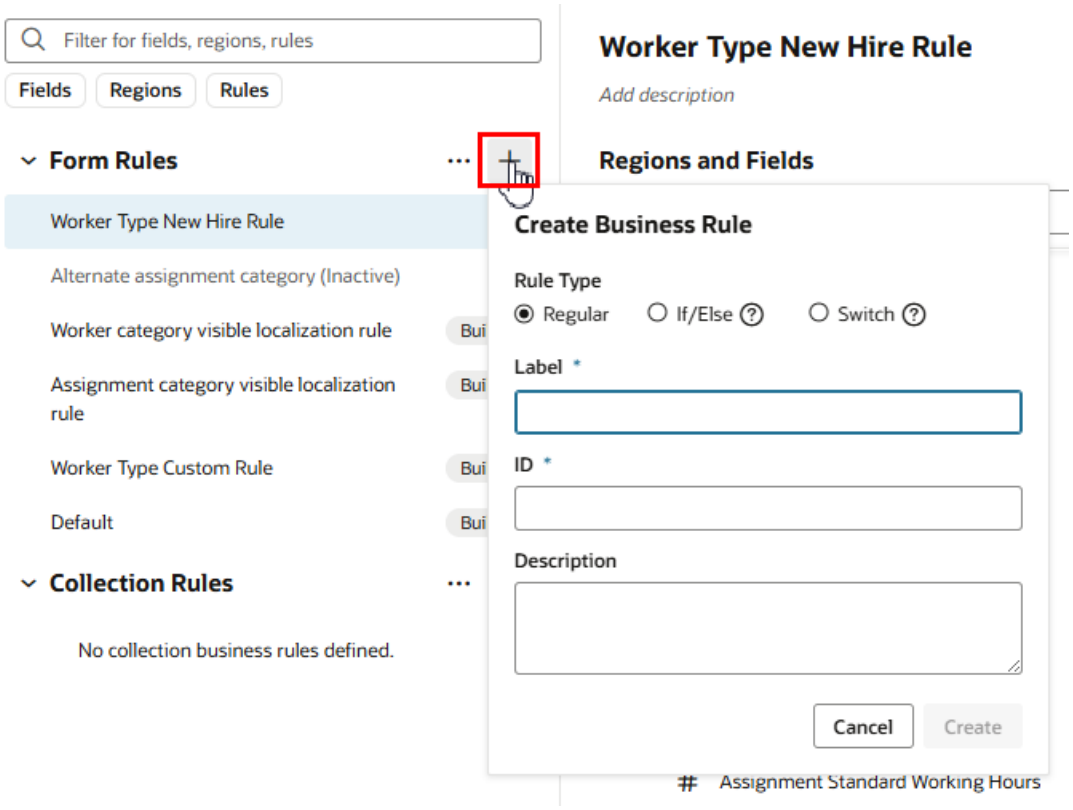
When validating fields with business rules, you configure *messages* that are displayed on the page when the rule's conditions are met.

Create a Rule For Forms

A form rule allows you to override certain properties for a dynamic form, assuming that certain conditions are met at runtime.

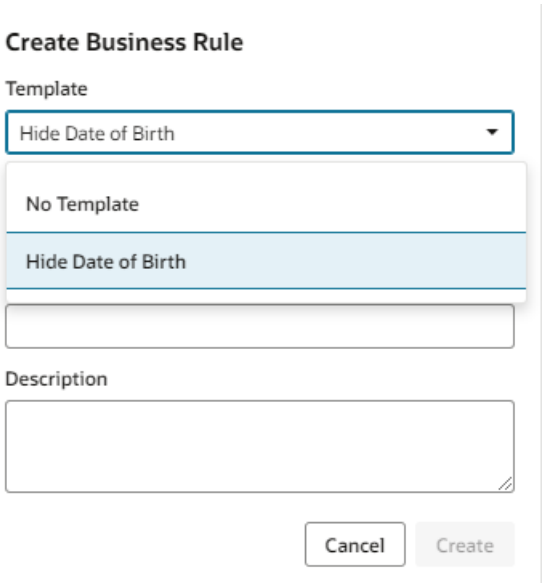
To create a form rule:

1. Open the page you want to configure, and then click **Configure Fields and Regions** in the Properties pane to open the business rules editor.
2. In the business rules editor, click + next to Form Rules to open the Create Business Rule dialog box.



3. Do one of the following.

   - Select the Rule Type in the dialog box.
     Select **Regular** to create a standard business rule. The other rule types (If/else and Switch) are special types of nested rules. For more about these nested rules, see Add an If/Else Rule and Add a Switch Rule.
   - Select a rule template.
     If there are any business rule templates defined in the app, you can select one using the Template dropdown list in the Create Business Rule dialog box:



A template can provide examples of how to configure business or validation rules, for example, how to specify rule conditions, how to write advanced expressions, or examples of how to override properties.

When you create a rule from a template, the new rule contains all the examples in the template. You can edit the examples in the new rule as needed.

4. Enter a label, id, and description for the rule.

   The id is generated automatically based on the label you enter, but you can modify the id if you wish. The description field is not required, but it can be helpful when you later try to understand what a rule is doing, especially when there are many rules.

5. Click **Create**

Your new rule is added to the top of the list of extension rules under Form Rules.

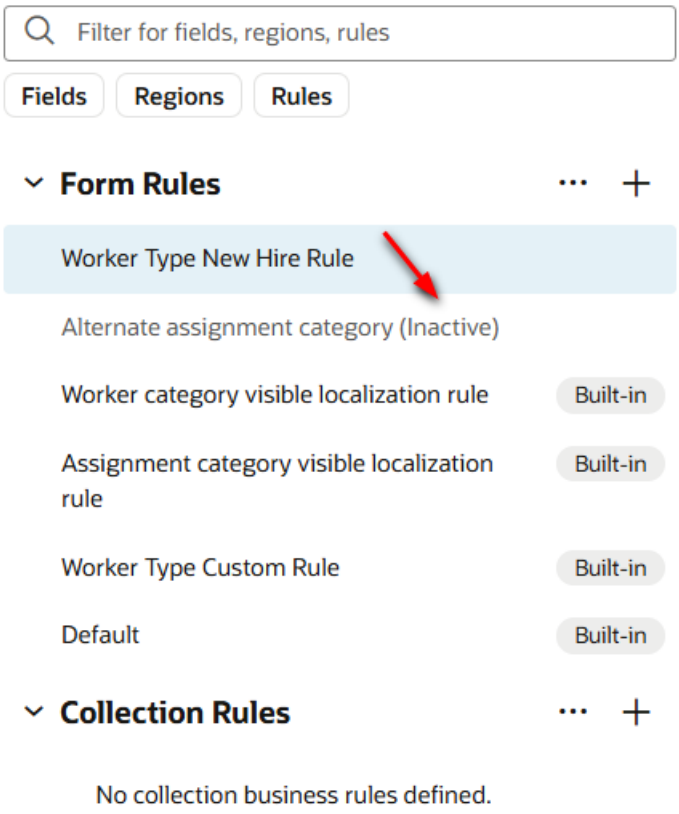
Rather than creating a rule from scratch or from a business rule template, you can duplicate an existing rule and use it as the basis for a new rule. To duplicate a rule, right-click the rule in the list, and then click **Duplicate**.

> **Note:**
>
> When duplicating rules defined in extension dependencies (the rules labeled "Built-in"):
>
> - You cannot duplicate a rule if it uses advanced expressions, for example, in the rule's condition or to override a field's value.
> - You cannot duplicate a rule if it uses any global functions that cannot be referenced in the extension. If the global functions are referenceable, when the rule is duplicated they are automatically imported, and the references to the function are updated in the rule.
> - When duplicating nested rules, you can only duplicate the top-level rule. When you duplicate the rule, the child rules are also duplicated.

To delete a rule, right-click the rule in the list to open the popup menu, and then click **Delete**.

Rules are evaluated in order, from bottom to top, so all extension rules are evaluated after the built-in rules. As you create more rules, make sure you position each one in the order you want them evaluated. To change a rule's position, right-click the rule and use the Move up and Move down items in the popup menu. To find out more about how rules are evaluated in Understand What Will Be Shown at Runtime.

If you decide you don't want to include a rule in the evaluation order, select the rule, then use the **Active** toggle switch in the upper right corner to deactivate it. (You can also right-click a rule and deactivate and activate it in the popup menu.) This enables you to still keep the rule so you can re-activate it later. You can tell at a glance if a rule is inactive because "(Inactive)" is appended to the name, like this:



Inactive rules are not included in the rule evaluation process.

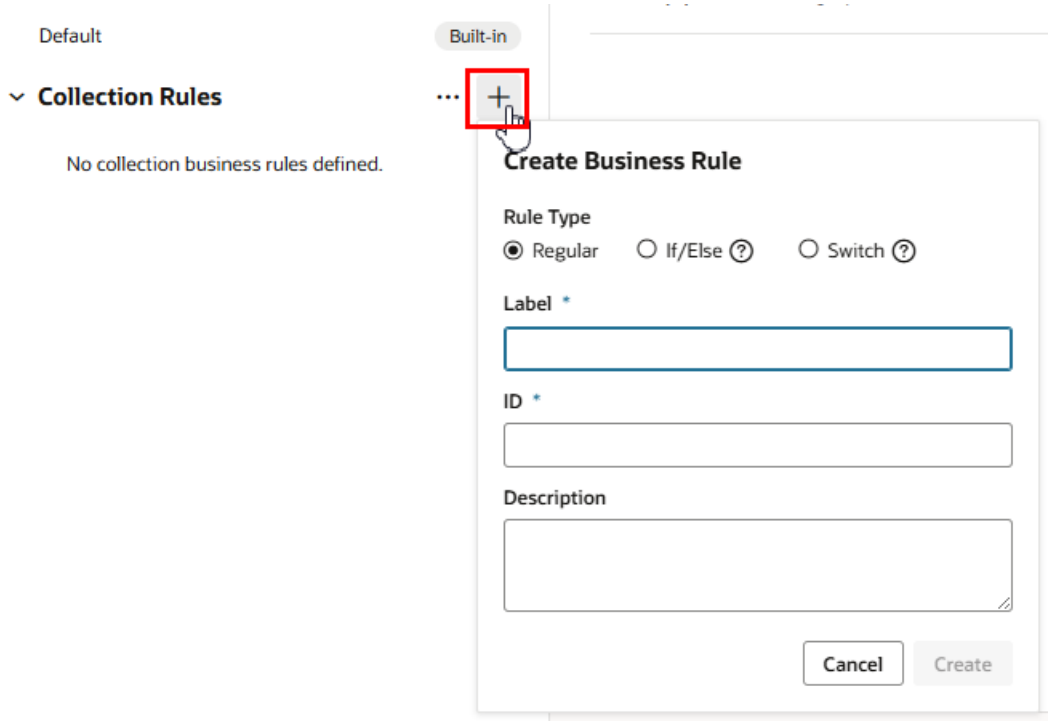
> **Note:**
>
> You can deactivate all the extension rules at once by clicking the three dots next to the Form Rules and Collection Rules headings, then clicking **Deactivate All**. This can be useful when debugging a page, allowing you to see the page with only the built-in rules applied. Use **Activate All** to reinstate all the rules at once, or use the **Active** toggle to selectively activate them as you work through your debugging process.
> You cannot deactivate built-in rules, as those have been locked into place by Oracle. In other words, a page's out of the box behavior is determined by the default rule, **plus** any built-in rules that may exist.

Create a Rule For Tables

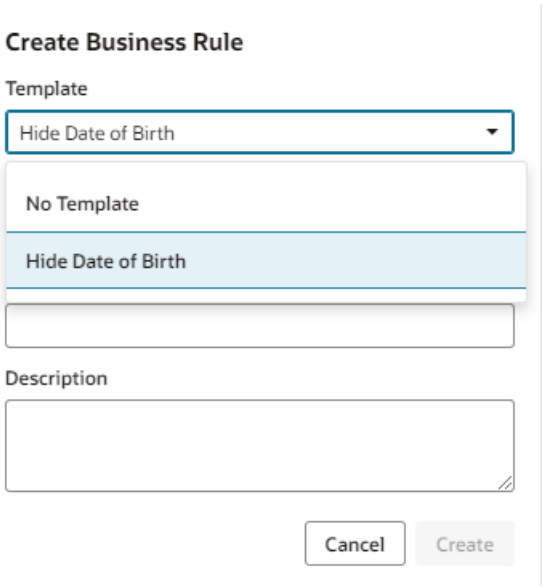
You use a collection rule when you want to hide or display columns in a dynamic table, assuming that certain conditions are met at runtime.

To create a collection rule:

1. Open the page you want to configure, and then click **Configure Fields and Regions** in the Properties pane to open the business rules editor..
2. Click + next to Collection Rules to open the Create Business Rule dialog box.



3. In the Create Business Rule dialog box, do one of the following.
   - Select the Rule Type in the dialog box.
     Select **Regular** to create a standard business rule. The other rule types (If/else and Switch) are special types of nested rules. For more about these nested rules, see Add an If/Else Rule and Add a Switch Rule.
   - Select a rule template.
     If there are any business rule templates defined in the app, you can select one using the Template dropdown list in the Create Business Rule dialog box:



A template can provide examples of how to configure business or validation rules, for example, how to specify rule conditions, how to write advanced expressions, or examples of how to override properties.

When you create a rule from a template, the new rule contains all the examples in the template. You can edit the examples in the new rule as needed.

4. Enter a label, id, and description for the rule.

   The id is generated automatically based on the label you enter, but you can modify the id if you wish. The description field is not required, but it can be helpful when you later try to understand what a rule is doing, especially when there are many rules.

5. Click **Create**

   Your new rule is added to the top of the list of extension rules under Collection Rules.

6. Click **Edit**, and then create a condition for the rule.

   See Set Conditions for a Rule.

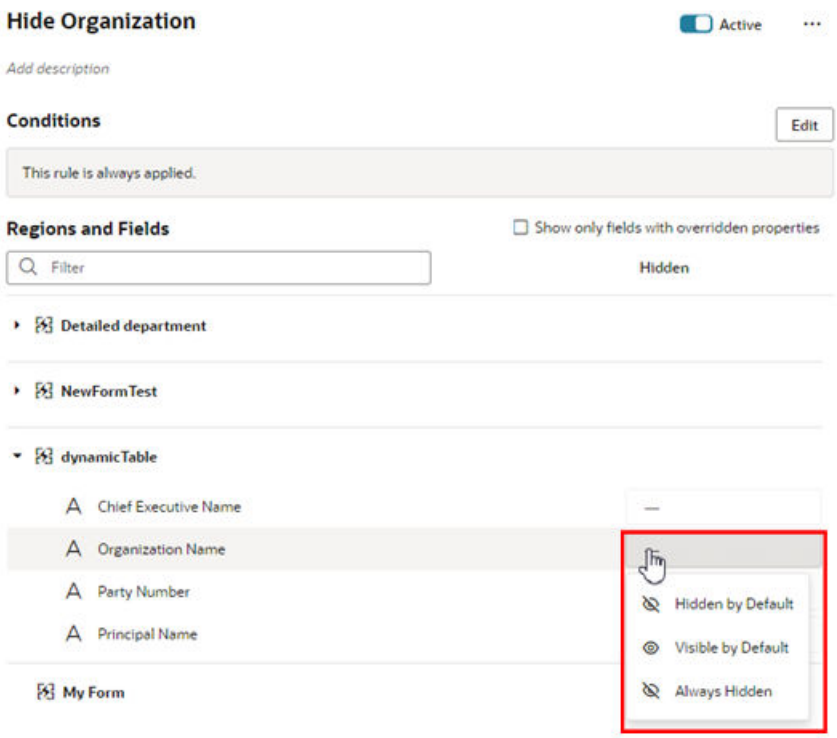
7. In the Fields area, locate the field corresponding to the column you want to configure.

   In a table, each field corresponds to a table column. This means that if a dynamic table is configured to display four fields, the table will have four columns.

   > **Note:**
   >
   > If you have trouble locating the field you want to modify, confirm that there were no errors loading services in the editor. The editor only lists fields from services that have loaded successfully. You'll see "Loading Completed with Errors" in the editor's status bar if any errors occurred when loading the editor.
   >
   > If you see the error message in the status bar, contact your project administrator about resolving the error.

8. Click the dash—or an existing value—for the field's Hidden property, and select a new value for the property in the dropdown list.

   In a collection rule, the only property you can override is the Hidden property:



The Hidden property has three options:

- **Hidden by Default** hides the table column when the page is rendered. A user can still use the table's popup menu to hide and show the column.
- **Visible by Default** displays the table column when the page is rendered. A user can still use the table's popup menu to hide and show the column.
- **Always Hidden** hides the table column. When this option is selected, a user cannot use the table's popup menu to display the column.

In a collection rule, you can set the Hidden property for a field even if it's not included in a table. The Hidden property in a collection rule is only applied to table columns. (The property does not affect fields in forms.)

Depending on how the app was configured, you might have options for filtering the list of fields displayed in the editor. The **Show only fields used by layouts** option filters out the fields not included in layouts. If you don't select this, the list contains all the object's fields. The **Show only fields with overridden properties** option limits the list of fields to those that have been modified by a rule.

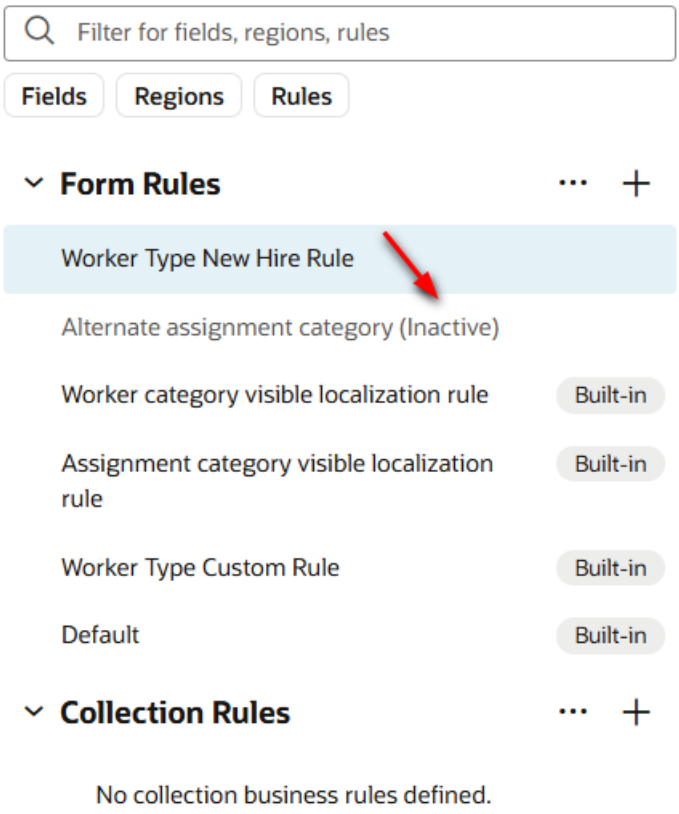
Rather than creating a rule from scratch or from a business rule template, you can duplicate an existing rule and use it as the basis for a new rule. To duplicate a rule, right-click the rule in the list, and then click **Duplicate**.

> **Note:**
>
> When duplicating rules defined in extension dependencies (the rules labeled "Built-in"):
>
> - You cannot duplicate a rule if it uses advanced expressions, for example, in the rule's condition or to override a field's value.
> - You cannot duplicate a rule if it uses any global functions that cannot be referenced in the extension. If the global functions are referenceable, when the rule is duplicated they are automatically imported, and the references to the function are updated in the rule.
> - When duplicating nested rules, you can only duplicate the top-level rule. When you duplicate the rule, the child rules are also duplicated.

To delete a rule, right-click the rule in the list to open the popup menu, and then click **Delete**.

Rules are evaluated in order, from bottom to top, so all extension rules are evaluated after the built-in rules. As you create more rules, make sure you position each one in the order you want them evaluated. To change a rule's position, right-click the rule and use the Move up and Move down items in the popup menu. To find out more about how rules are evaluated in Understand What Will Be Shown at Runtime.

If you decide you don’t want to include a rule in the evaluation order, select the rule, then use the **Active** toggle switch in the upper right corner to deactivate it. (You can also right-click a rule and deactivate and activate it in the popup menu.) This enables you to still keep the rule so you can re-activate it later. You can tell at a glance if a rule is inactive because "(Inactive)" is appended to the name, like this:



Inactive rules are not included in the rule evaluation process.

> **Note:**
>
> You can deactivate all the extension rules at once by clicking the three dots next to the Form Rules and Collection Rules headings, then clicking **Deactivate All**. This can be useful when debugging a page, allowing you to see the page with only the built-in rules applied. Use **Activate All** to reinstate all the rules at once, or use the **Active** toggle to selectively activate them as you work through your debugging process.

Use Nested Rules

Nested rules help you avoid having to write complicated conditions and can improve rule organization, especially if you have lots of rules.

A nested rule is simply a parent rule with one or more child rules, indicated in the UI by indentation. When you use nested rules, the children of a rule can override the properties set by its parent(s).

In addition to the typical nested rules, there are two special types of nested rules (If/Else and Switch) that behave slightly differently. These special types are described later, but first let's look at typical nested rules.

Here's a simple example of a nested rule:

This example shows two sets of nested rules:

- **Country is USA** and **Country is Canada** are at the same level
- **Full Time Employees** is nested beneath **Country is USA**
- **Hire Date before 2023** and **Hire Date after 2023** are both children of **Full-time Employees**.

The red numbers indicate the order in which these rules are evaluated at runtime. Let's take a closer look at what that means:

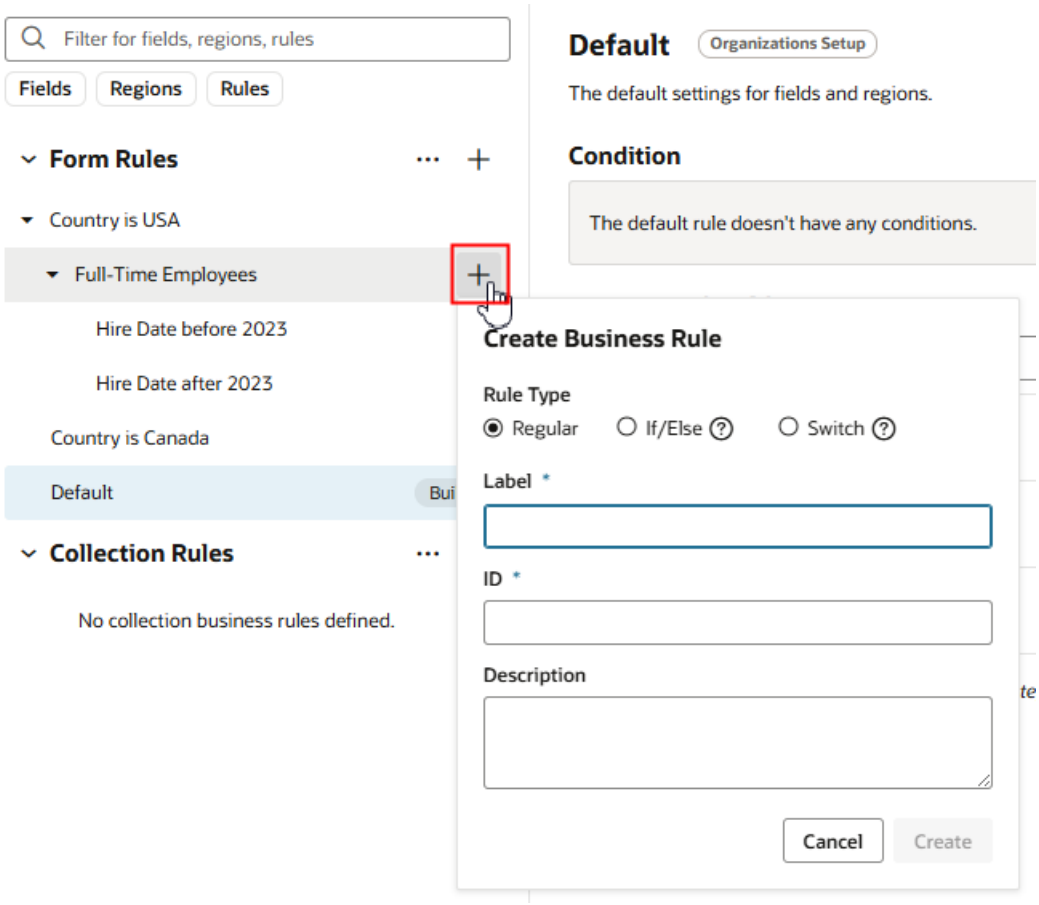
1. Evaluation begins at the bottom with **Country is Canada**. If the conditions for this rule are met, the rule's property overrides are applied. Next:
2. **Country is USA** is evaluated.
   - If the rule's conditions are NOT met, the rule doesn't override any properties. In addition, its child rules (**Full-time Employees**) are never evaluated.
   - If the conditions for **Country is USA** ARE met, the property overrides defined in the rule are applied, and **Full-time Employees** is evaluated.
3. If the conditions for **Full-time Employees** are NOT met, the rule does not apply any properties, and its children (**Hire Date after 2023** and **Hire Date before 2023**) are never evaluated.
4. If the conditions for **Full-time Employees** ARE met, the property overrides defined in the rule are applied, and its children are evaluated.
   - **Hire Date after 2023** is evaluated first, because it is lowest, followed by **Hire Date before 2023**.

Although you could write a more complex set of conditions to achieve the same outcome, nested rules make it much simpler to apply multiple property overrides at runtime.

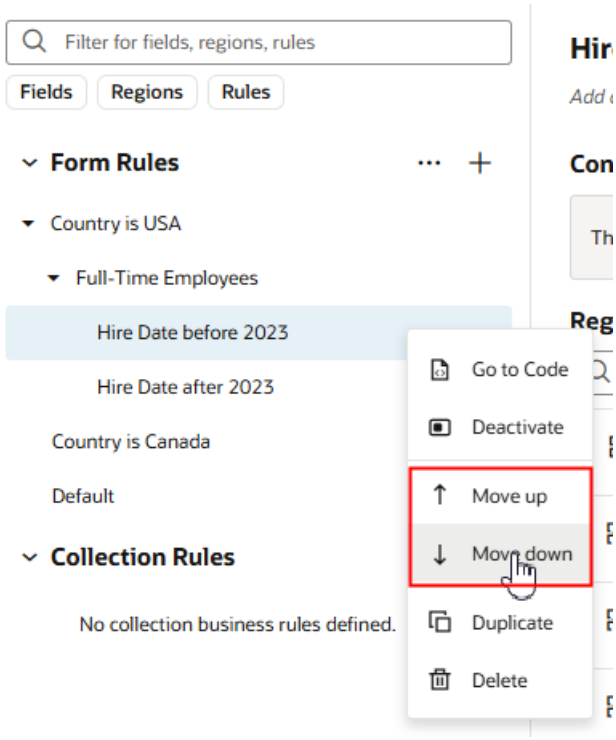
Add a child rule

You can add a child rule to any of the rules you've created:

1. Click + next to the rule that you want to be the parent rule.



2. Type the Label of the child rule. (The ID field is automatically populated based on the Label, but you can type a different ID if you want.) Click **Create**.
3. Right-click the child rule, and then use **Move up** and **Move down** in the options menu to move the rule into the position you want.
   Remember, child rules that are peers are evaluated from the bottom up, so the order is important.



When you duplicate a rule, the rule's children are also duplicated. In the example above, if you duplicated **Full-time Employees**, a new rule (**Full-time Employees copy**) will appear under the parent (**Country is USA**), along with its child rules.

> **Note:**
>
> You cannot duplicate rules defined in extension dependencies (the rules labeled Built-in), which includes the Default rule.

Add an If/Else Rule

An If/Else rule is a special type of nested rule where only the first child rule meeting the conditions is applied, and none of the remaining child rules are evaluated.

You use an If/Else nested rule when only one of the rules in a list of rules needs to be applied. Unlike regular business rules, where every rule is evaluated, the child rules in an If/Else rule are evaluated in order until one is applied; the unevaluated child rules are skipped. This means that rules won't be evaluated if they don't need to be.

For example, you might have a rule that should be applied when the worker category is Pending Worker, and some rules that should only be applied when the category is New Hire. Handling this type of situation, where only one rule should be applied, is very simple if you use nested If/Else rules. Let's take a look at how you could do this.

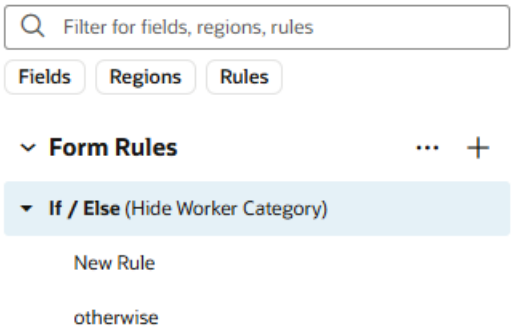
1. In the business rules editor, click + to open the Create Business Rule dialog box.

   You can create If/Else rules for form and collection rules.

2. Select **If/Else** in the dialog box.

3. Enter a label, id, and description for the rule. Click **Create**.

An If/Else rule containing two child rules is created by default. The default names for the two child rules are "New Rule" and "otherwise", but you can rename them to something more meaningful. You can add as many child rules as you want, but remember that only one of them will be applied.



4. Select New Rule.

   The first child rule is the first "If" part of the "If/Else" nested rule. In If/Else nested rules, rules are evaluated from the top to the bottom. New Rule is the first rule in the list, so it's the first to be evaluated.

   a. Click Edit, and then define the rule's conditions, and give the rule a meaningful name.

      Let's define this rule so that it's applied when the worker category is Pending Worker.

   b. Set the property overrides you want applied when the conditions are met.

5. Select the otherwise rule.

   The otherwise child rule is the "Else" part of "If/Else", and is only evaluated if none of the preceding rules were applied.

   a. Click Edit, and then define the rule's conditions, if you wish, and give the rule a meaningful name.

      If you don't set any conditions, the rule is always applied if it's evaluated. If a rule has no conditions you should make sure it's the last rule in the list.

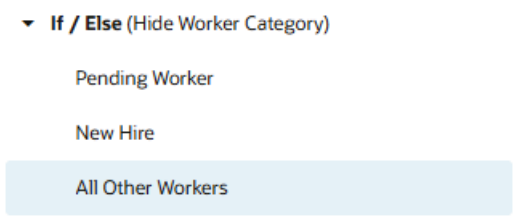
      Let's define this rule so that it's applied when the worker category is New Hire.

   b. Set the property overrides you want the rule to apply.

6. Add and configure additional child rules.

   You can add as many child rules as you want, and arrange the order they are listed.

   Let's add a child rule that is applied to all other worker categories, and move the rule below the New Hire rule.



Click **Edit** to define the conditions, and then set the property overrides you want the rule to apply.

At runtime, starting with the first rule, the rules are evaluated in order until a rule is applied. In this set of nested rules, the last rule is applied only if the worker category is not Pending Worker or New Hire.

Add a Switch Rule

A Switch rule is a special type of nested rule where instead of evaluating a list of rules, a switch parameter (switchOn) is evaluated, and the parameter's value determines which case is applied.

Let's say you want to apply a rule based on the role of the current user, and there are several different cases. For example, when the user's role is Payroll Manager, you want the Payroll Manager rule applied, when it's Administrator, the Administrator rule is applied, and so on. You can use the Switch nested rule to apply the appropriate rule based on the value of the user roles field, instead of evaluating a long list of rules at runtime.

Let's create a Switch rule that evaluates the role of a logged in user. You then create some different cases, for example, one case if the user role is Payroll Manager, and another case if the role is Administrator. At runtime, the user role is evaluated, and based on the user role value, the appropriate case is applied.

1. In the Business Rules tab, click + to open the Create Business Rule dialog box.

   You can create Switch rules for form and collection rules.

2. Select **Switch** in the dialog box.

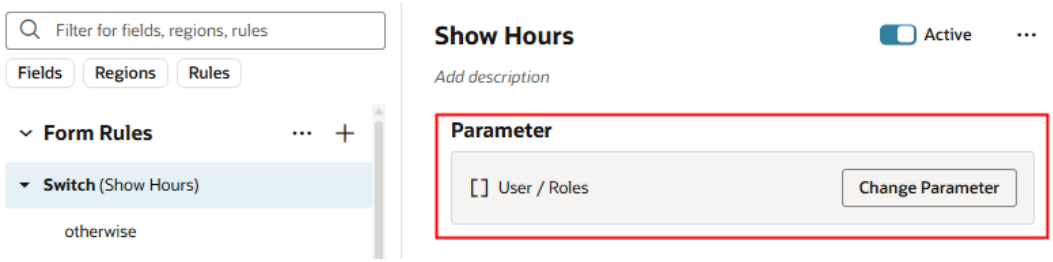
3. Enter a label, id, and description for the rule. Click **Create**.

4. Select the Switch parameter.

   a. Select the **Switch** parent rule

   b. In the rule's Parameter panel, click **Change Parameter**, and then select a parameter in the dropdown list. Click **Change**.

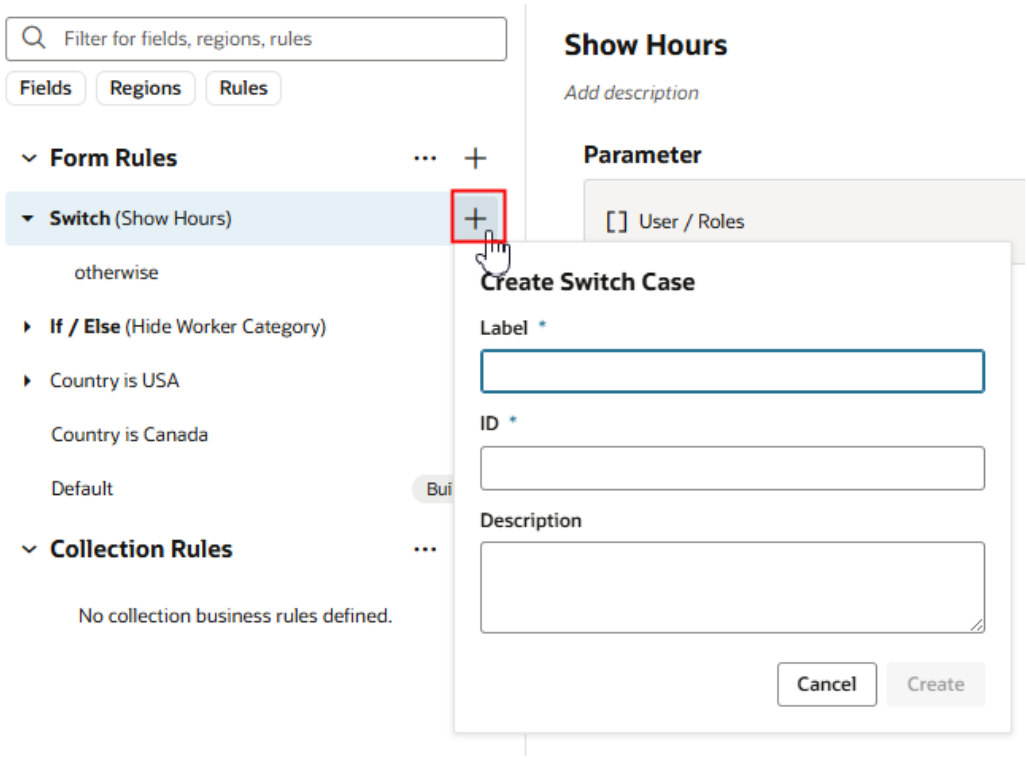
   The Parameter panel displays the parameter (the `[] User / Roles` field in this example).



5. Create cases.

   Let's create a case that is applied when the user role is Payroll Manager.

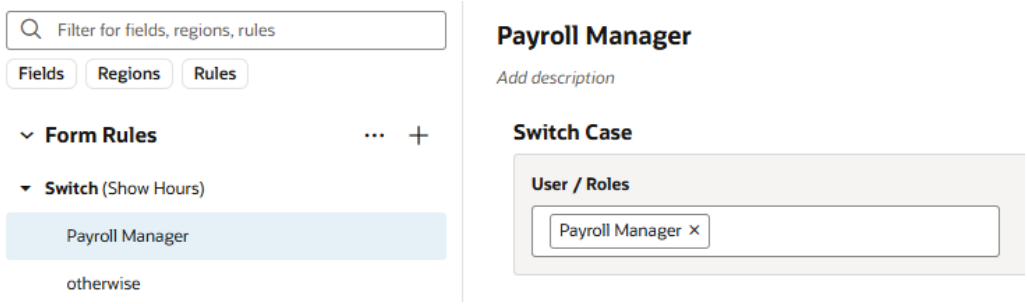
   a. Click + next to Switch to open the Create Switch Case dialog box.



b. Enter a label, id, and description for the case. Click **Create**.

c. Select the case, and then enter a parameter value in the Switch Case text field.

You can enter multiple parameters in the Switch Case field. For this case, enter Payroll Manager, so that the case is applied when the user role is Payroll Manager.



d. Set the overrides you want the case to apply.

6. Create any additional cases.

For example, create a case called Administrator, and then set the case's parameter value to Administrator. You then set the overrides that should be applied when the user role is Administrator.

7. Select the last rule in the list of nested rules (the default name is otherwise), and set the property overrides you want the rule to apply.

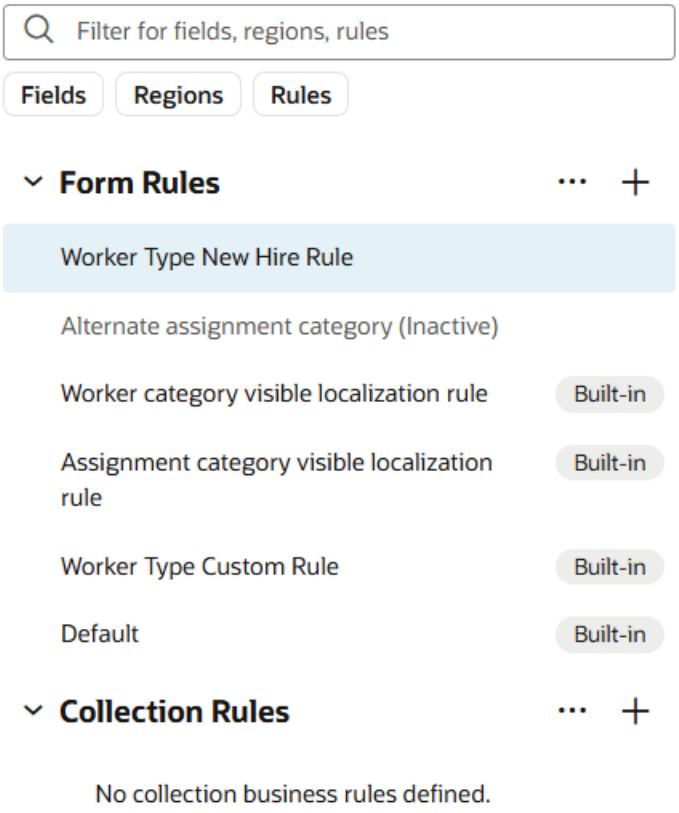
The last rule is applied if none of the cases have been applied. You cannot create conditions for the otherwise rule.

Filter Your Rules

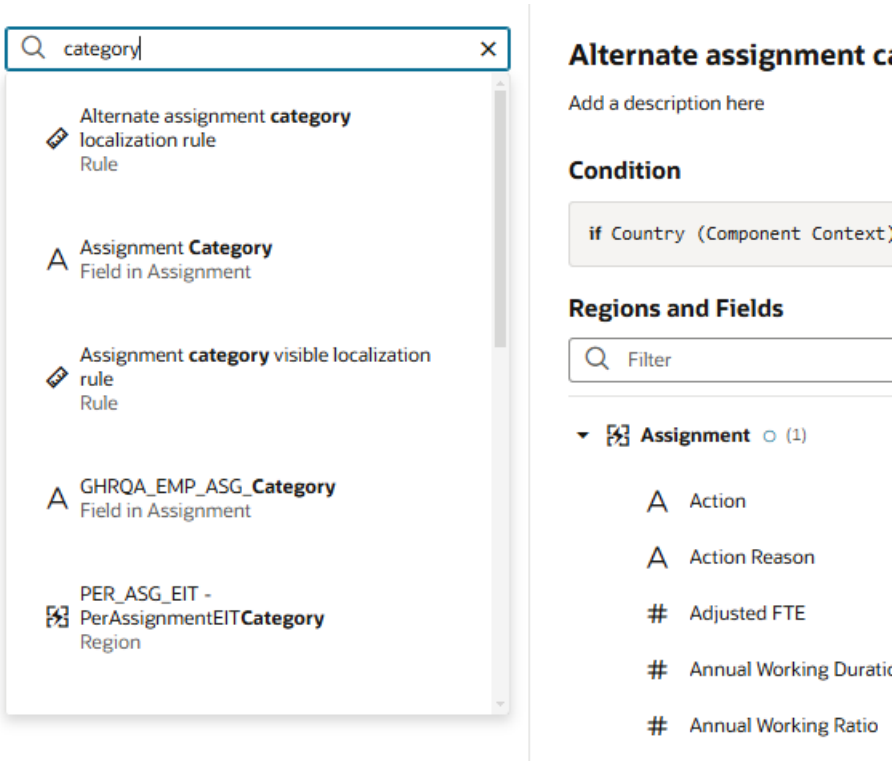
To help you manage your display and find things quickly, VB Studio offers two Filter fields: one at the rules level and one for regions and fields.

If you have hundreds of rules, it can be painful to scroll through laboriously to find the one you're interested in. Instead, use the Filter field to zero in on the rule you want.

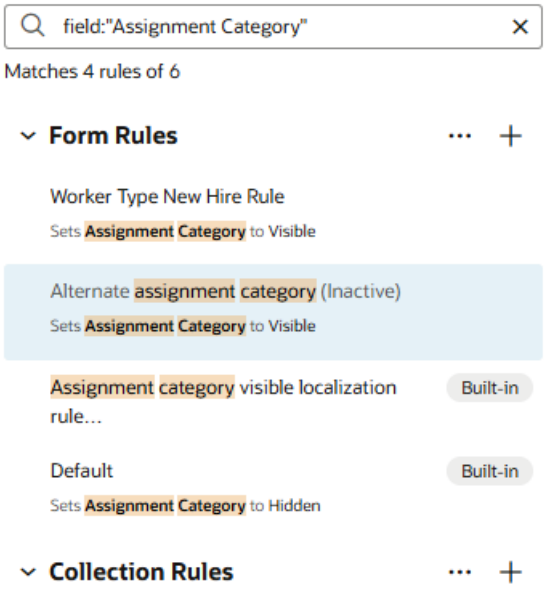
Suppose you have a display that looks something like this, only with many more rules:



Simply by typing "category" in the Filter field, you can see the rules, regions, and fields containing that word, thanks to the Filter field's auto-complete feature:



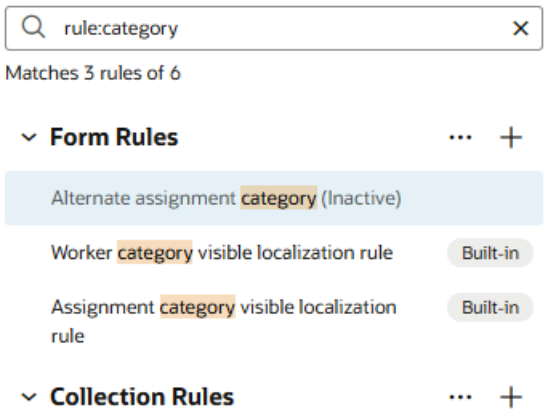
As you can see in the above image, an icon appears next to each search hit to indicate the type of entity containing the word: region, rule, or field. The list also contains a magnifying glass icon you can select to list all rules where the word occurs. In the image above, when you select the Assignment Category field, the list of rules is filtered to only include those that override a property of the Assignment Category field:



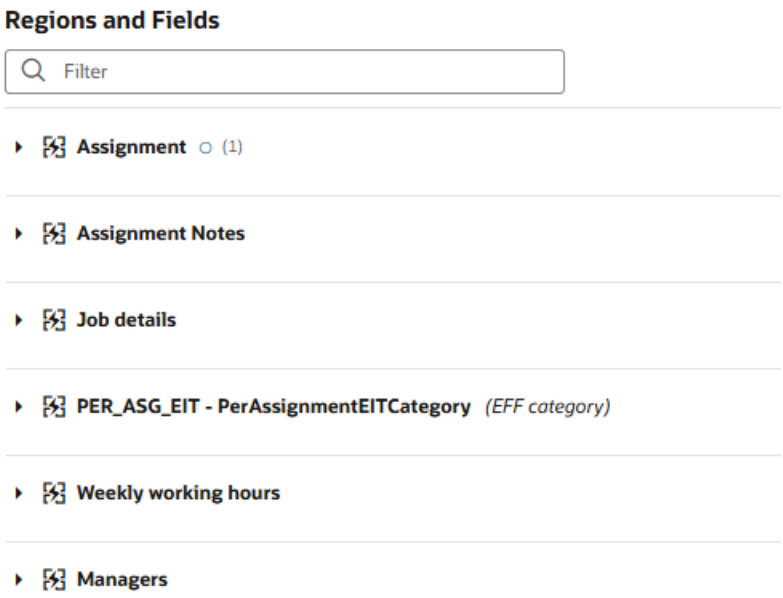
This makes it easy to tell at a glance where the Assignment Category field has been overridden, and to what property value.

The buttons beneath the Filter field (**Fields**, **Regions**, **Rules**) let you limit the scope of your search, making your filter operation that much more efficient. For example, when you click

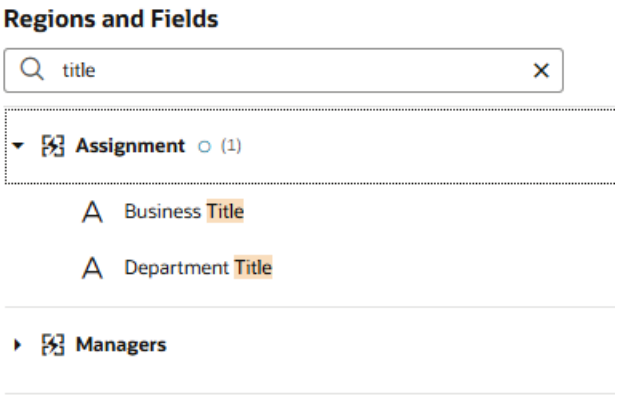
Rules, and then enter a term like "category" in the Filter field, the filtered list shows the rules that contain the term in the name:



The Filter field in the Regions and Fields section works similarly, but with a slight difference. If you don't have a region expanded in the display, but the filter criteria matches a field within that region, you'll see the region, but you won't see the field until you expand the region. For example, suppose you don't have any regions expanded, like this:



You're looking for fields with the term "title" so you enter that in the Filter field. The Assignment and Managers regions are displayed, but the other regions are hidden. It's not until you expand the Assignment region that you see the fields containing the term "title":



> **Note:**
>
> When using the region and field Filter field, it doesn't matter which rule is currently selected.

Set Conditions for a Rule

You determine when a rule is applied by defining a *condition*. For example, you might create a rule that is applied only when the user is in Canada and has the Manager role, or a rule that is applied when a form field is set to a specific value. In some pages, for example, when a page has a form for updating a record, you might need to add conditions to ensure that all the required details are entered in the form.

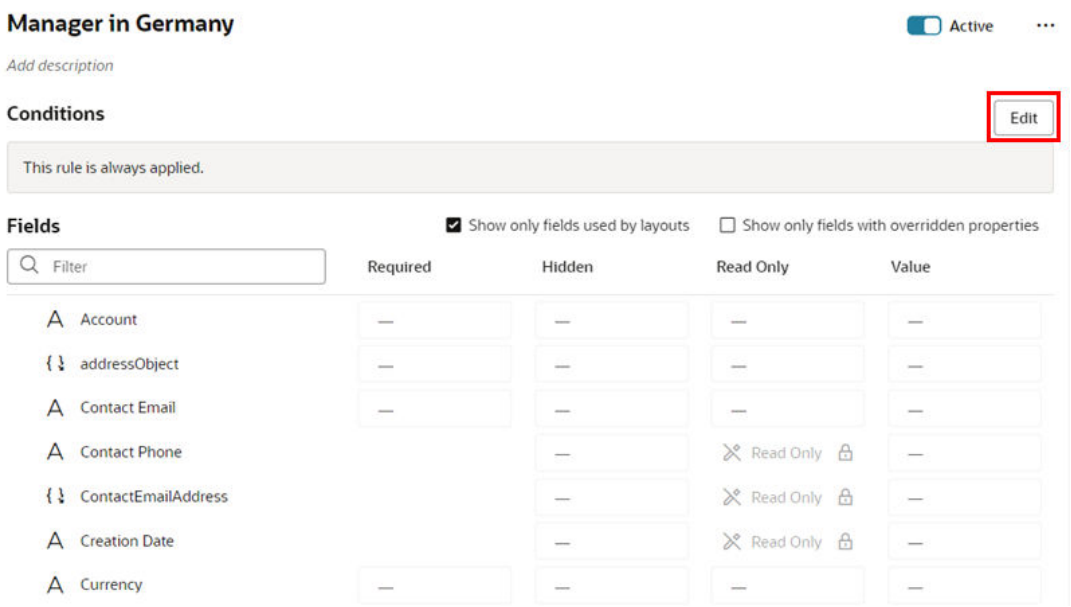
There are two ways to define the rule's conditions. The first way is to use the basic condition builder to create conditions by selecting criteria and values. This way should be enough to define most conditions. However, if you need to create more complex conditions, and you are comfortable working with expressions, you can click **Use Advanced Expression** to open the visual expression editor. For more about using the expression editor, see Build Advanced Expressions.

Rules define overrides that are applied to properties only when the rule's conditions are satisfied at runtime. For conditions that use criteria in the User context, like User Authenticated (`$user.isAuthenticated`) or Roles (`$user.roles`), the condition is met if the logged-in user satisfies the condition.

> **Note:**
>
> When using the Roles criterion (`$user.role`) in a condition, start entering the role name you need in the Value field. The Value drop-down will display up to 25 matching Oracle Cloud Applications job and abstract roles; keep typing until the desired role displays. (The drop-down will not list any duty roles. If you want to specify a duty role, you can manually type the duty role name in the Value field.)
>
> If the Roles drop-down list is empty, it might be because you haven't been assigned the correct role or privileges to see the roles. When extending a Human Capital Management (HCM) application, to see the roles in the list, you need to be granted the `PER_REST_SERVICE_ACCESS_USERS_AND_ROLES_LOVS_PRIV` privilege, or assigned a role granted that privilege (for example, `Employee` (`ORA_PER_EMPLOYEE_ABSTRACT`). Contact your instance administrator if you require this user privilege.

To create a condition for a rule:

1. Select the rule you want to edit, then click **Edit** to open the condition builder:



2. In the condition builder, define one or more conditions.

   When possible, Visual Builder starts you off by pre-populating the conditions with criteria used in the page, but you can use different criteria in your condition. The criteria, and their available values, depend on the object and what has been set for the page.

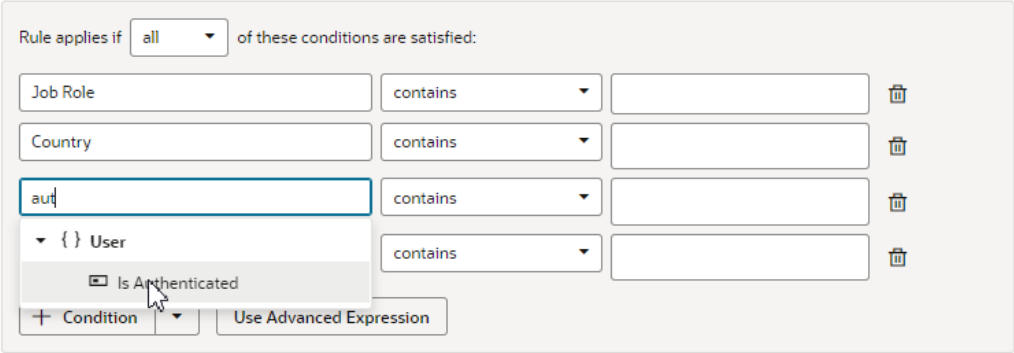
   To define a condition:

   a. Select a criterion from the dropdown menu.

      You can choose any of the listed criterion in your conditions, but you cannot change the list of available criteria to add your own.

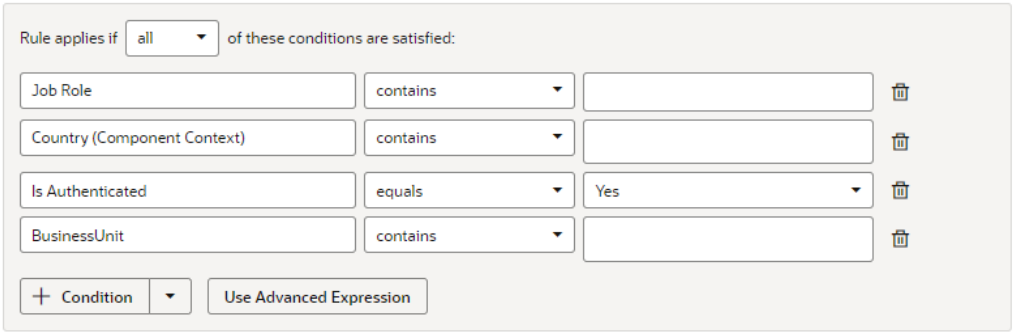
      Some fields in the list of criteria might be grouped under Object Context or Component Context. These are fields the extension developer has explicitly selected as useful when creating conditions.

If you know what criteria and values you're looking for, you can try typing in the field to filter the list:



b. Select an operator.

The options available in the operator dropdown menu are pre-defined based upon the criterion's type. For some criteria, there might be several available operators, including operators to "match" or "not match" a JavaScript regular expression set as a value. For other criteria, such as a boolean type like User Authenticated, the operator menu only has `equals`, and the only available values in the value menu are `yes` and `no`:



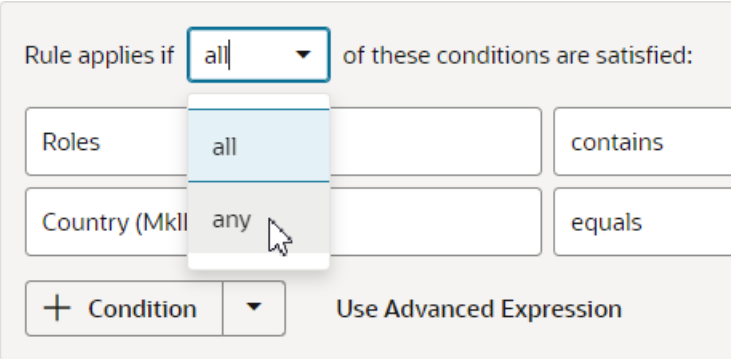
c. Specify the values.

Type a value or simple expression in the field, or select a value from the dropdown list. Depending on the criterion, you might be able to choose multiple values.

To compare the criterion's value to the value of another field, you can select another field with the same type in the dropdown list.

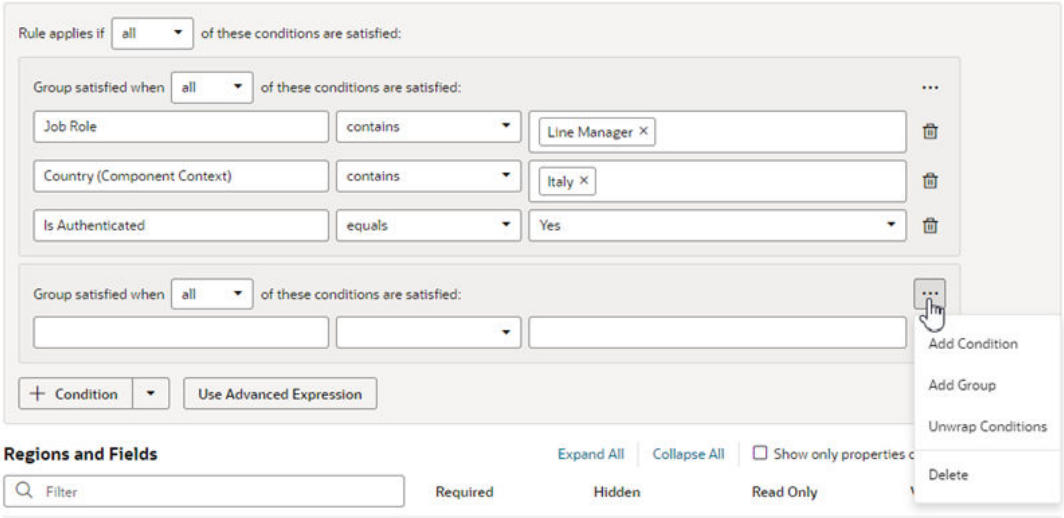
3. To add another condition, click **+ Condition**, and then define the new condition. To remove a condition, click  .

When you have more than one condition, the rule is applied only when ALL the conditions are true (by default). However, you can change this to ANY, which means only one of the conditions has to be true in order for the rule's overrides to be applied:



4. To create a group, click the arrow next to **+ Condition**, and then select **Add Group** in the dropdown menu. Grouping conditions lets you create more complex conditions for the business rule.

   When you click Add Group, your existing conditions and groups are combined into one group, and a new group containing a condition is added.



   Each group has a menu with options for managing the group:

   - **Add Condition**. Adds a new empty condition to the group.
   - **Add Group**. Creates a new group within the group.
   - **Unwrap Conditions**. Ungroups the conditions, and the empty group is deleted.
   - **Delete**. Deletes the group and all the conditions in the group.

   After creating a group, you can add and define conditions in the group, and set the logic (`any`, `all`) for the groups.

5. Click **Done** to close the condition builder.

   Conditions are always saved automatically, so you don't need to worry about explicitly saving your changes when editing a condition.

> **Note:**
>
> Keep in mind that different Oracle Cloud Applications actions (or pages) can interpret conditions differently, which is outside of the realm of VB Studio's knowledge or control. For example, suppose the Country condition is used for a given page in Human Capital Management (HCM). The Hire an Employee action may interpret this as the person who is being hired, while another action might interpret this as the person viewing the page. To understand how conditions are interpreted for a given Oracle Cloud App, consult that App's documentation.

To change the conditions for an existing rule (not a new one), click the rule in the list, then click **Edit** above the Conditions pane.

If you're creating a new rule, the next step is to set *properties* for the fields.

Create Condition Using a Field's Initial Value

When choosing a field in the condition builder, you might see fields listed under both **Field Values** and under **Initial Field Values**.

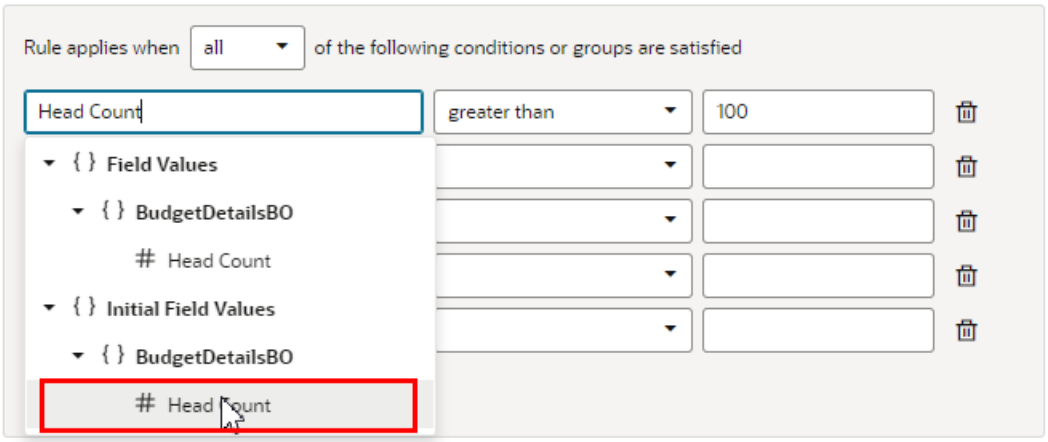
The Initial Field Value is set in the page's metadata by the page developer. The initial value set in the metadata isn't affected by changes made by rules or users. If the field is a number and no initial value is set, the value is 0; if the field is a string, the value is an empty string.

The Field Value, on the other hand, is the current value displayed in a page. This value may already have been modified by a rule or user. For example, say the value for the Head Count field is set to 50 in the page's JSON (the Initial Field Value). There might be some rule that sets the field's value (the Field Value) to 60. The Field Value is displayed in the Head Count field in the form. The user may then change the Head Count field to 70 in the form, so the Field Value for Head Count is now 70. The Head Count Initial Field Value, however, is still 50.

Let's look at how to add a rule that makes the Head Count field read only when the field's initial value is greater than 100.

1. In the business rules editor, create a new form rule.
2. Create a condition where the initial value of Head Count is greater than 100.

   In the criterion dropdown list, be sure to select Head Count under **Initial Field Values** (as opposed to under **Field Values**):



3. Set the Read Only property of the Head Count field to **Read Only**.

In this example, if the page developer has set the Head Count field's initial value to a value greater than 100, the Head Count field is read only when rendered in the page. If the value was set to less than 100, or no value was set, the Head Count field in the form is editable.

Build Advanced Expressions

When creating conditions for business rules, you may find your conditions are more complex than you can achieve using the basic condition builder. If this occurs, you can build your own custom expressions to suit your needs.

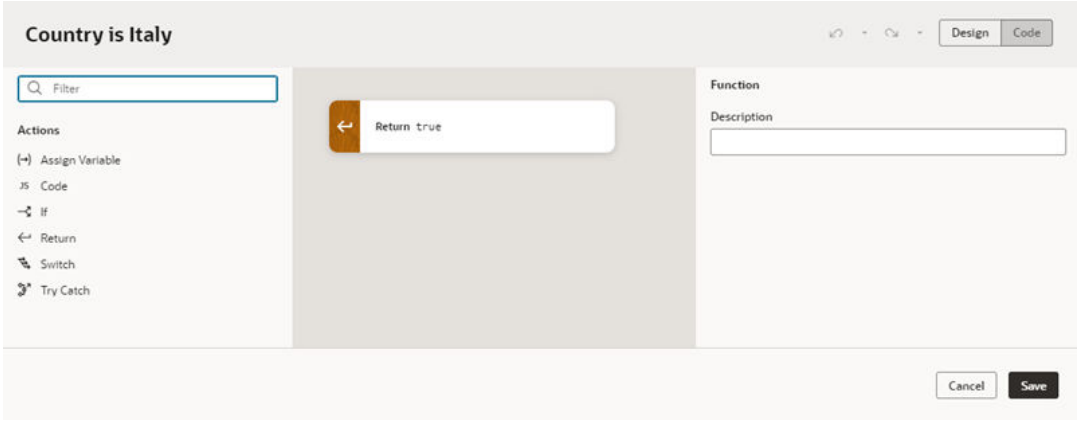
> **Note:**
>
> If you use advanced expressions in the condition builder, you will no longer be able to use the basic condition builder. If you have already defined some conditions in the basic condition builder, they will be displayed as actions in the Advanced Expression editor.

To create an advanced expression:

1. Click **Use Advanced Expression** in the condition builder.

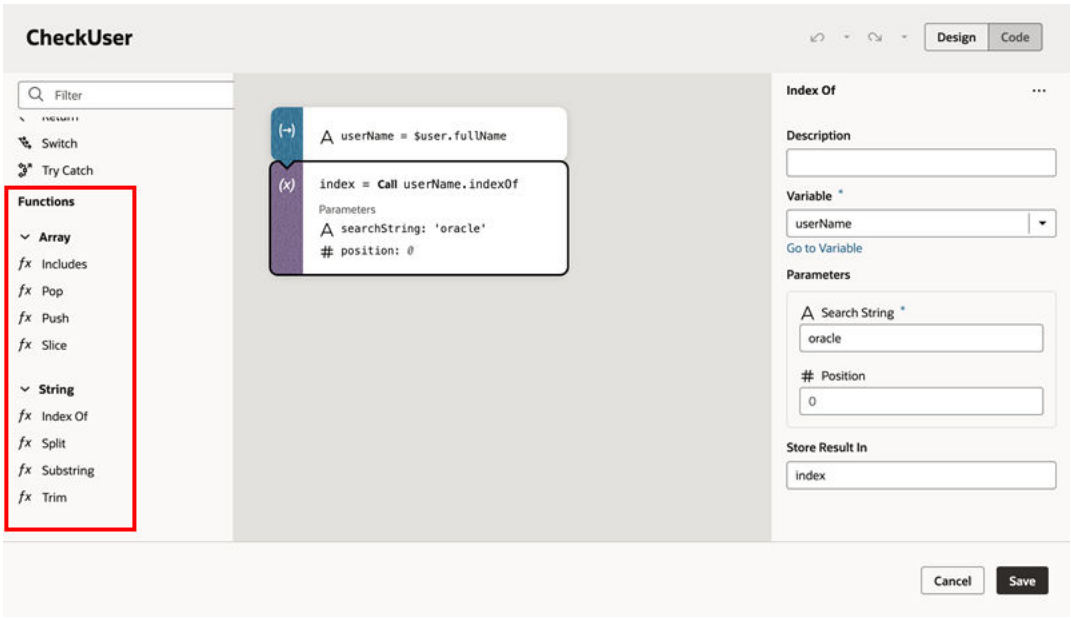
   The Advanced Expression editor has a Design view, which you use to visually create an expression, and a Code view, where you can type the expression. You toggle between the two views using the **Design** and **Code** buttons in the header.

   This image below shows Design view, which has an Actions palette on the left, a canvas in the middle, and a Properties pane on the right:

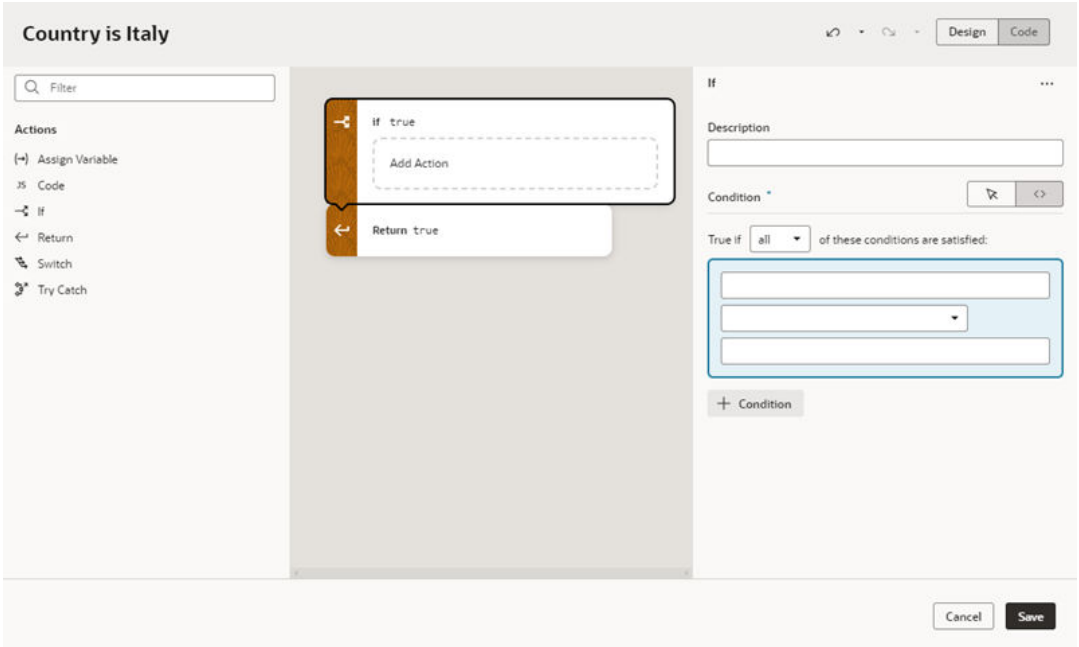


2. Drag an action from the palette and position it on the canvas.

   VB Studio includes some basic actions in the palette to help you build expressions. The palette might also contain additional custom functions that have been added to the extension. For example, if Oracle developers have added any global functions to the extension, like Add Numbers or Multiply Numbers, they are listed under Functions in the palette. Like the basic actions, you can drag the functions from the palette into the expression, and configure them in the Properties pane. (In Advanced mode, you can add your own global functions. See Add JavaScript Modules As Global Functions in *Extending Oracle Cloud Applications with Visual Builder Studio*.)



Let's take a look at what happens when we drag an If action onto the canvas:



When the If action element is selected on the canvas, the Properties pane contains a Description field and a condition builder.

3. Define the action in the Properties pane and on the canvas.

   a. Edit the action's properties in the Properties pane. The properties displayed in the Properties pane vary according to the action.

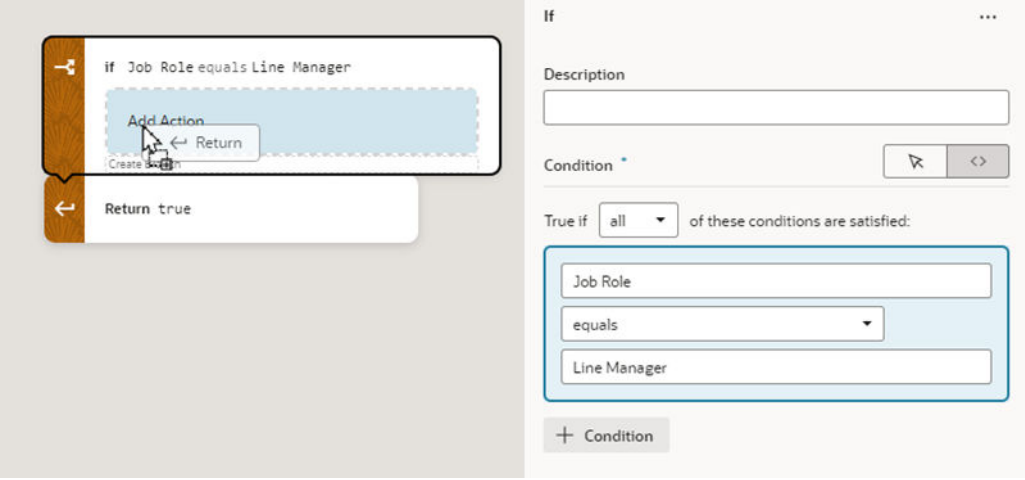
      For an If action, the Properties pane has a condition builder for selecting the condition's criterion, operator, and value. Click **+ Condition** in the Properties pane to add a condition. Right-clicking a condition opens an options menu for adding, moving,

and deleting the condition. You can also reposition a condition using drag-and-drop in the condition builder.

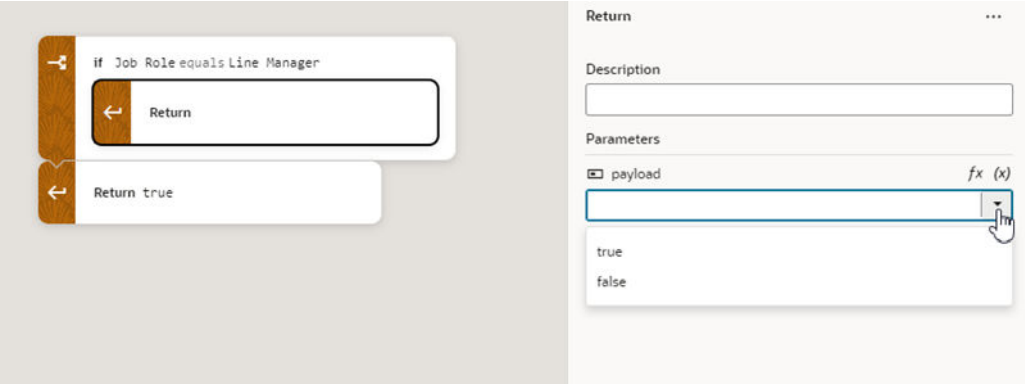
You can use the condition builder's toggle buttons to switch between the Design and Code views:

b. On the canvas, drag actions from the palette (for example, Return, Assign Variable, Try Catch) into the action's Add Action drop target.

For an If action, you need to define what happens when the condition is true. If the condition is true, and you want to return from the advanced expression, drag a Return action into the action's drop target:



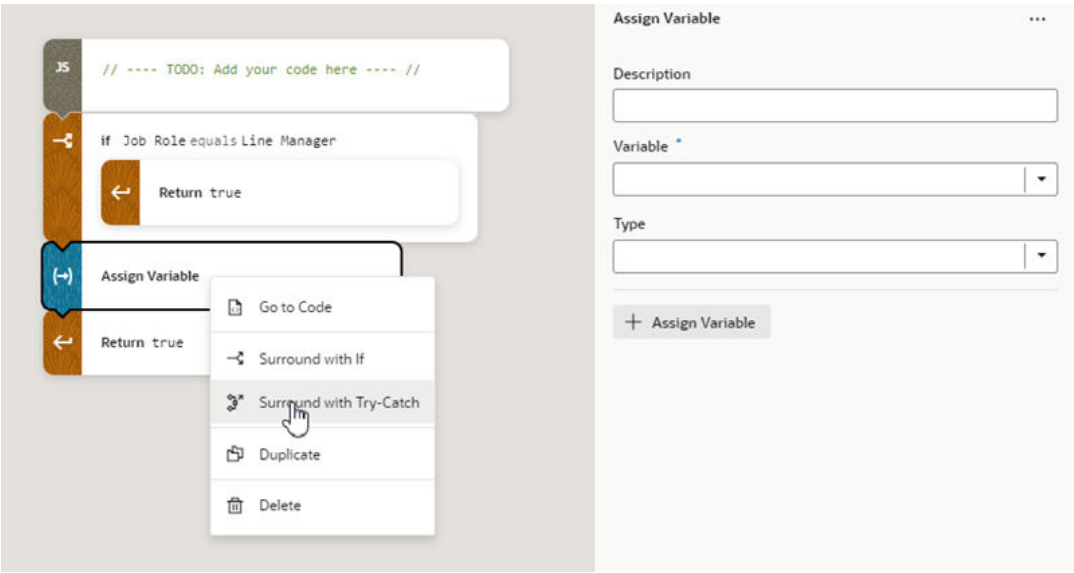
Now select the Return action on the canvas, and then select 'true' in the Payload dropdown list in the Properties pane:



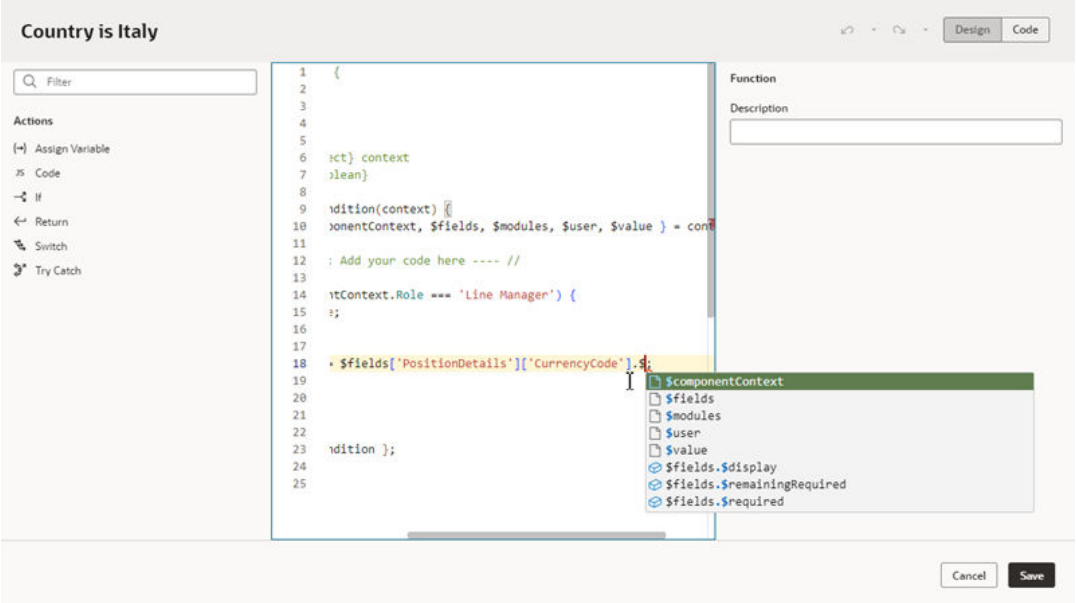
4. Repeat steps 2 and 3 to add more actions to the expression.

   As you add more actions, you can reorganize the order by grabbing the front of the action element on the canvas, and then moving it into a new position:

   Right-clicking an action on the canvas opens a popup menu with some convenient shortcuts, including deleting the action and switching to Code view:



At any point, if you're comfortable typing expressions you can click **Code** in the header to open the editor's Code view. In Code view, you can type your expression directly and take advantage of code completion in the editor:

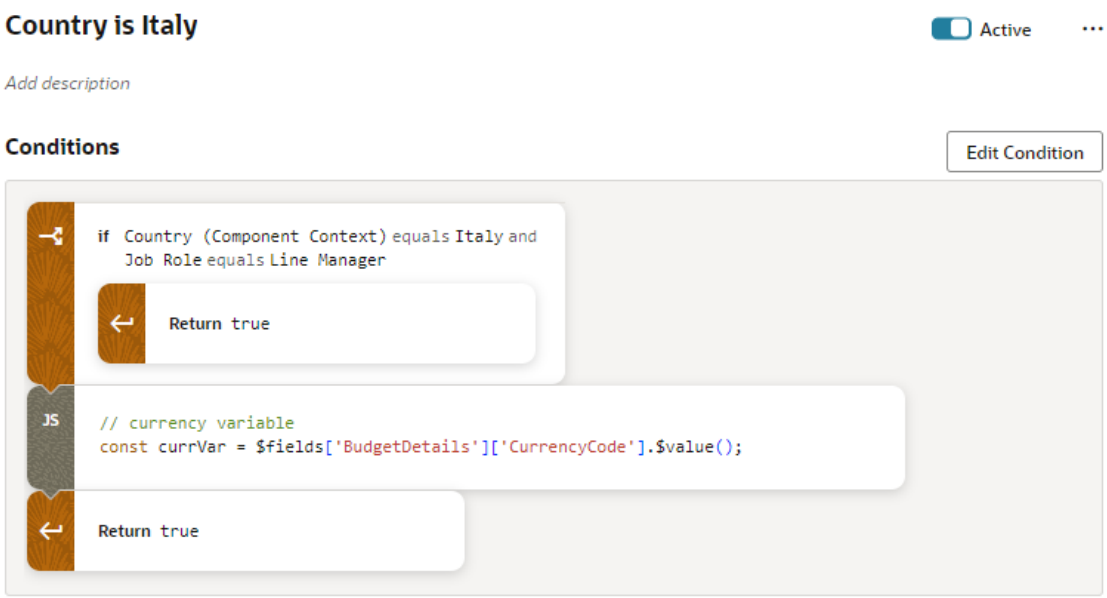


5. Confirm that the payloads for your Return actions are correct.

   When a Return action is selected on the canvas, you can choose a payload in the dropdown list in the Properties pane.

6. When you're finished building your expression, click **Save**.

When you look at your rule, the condition looks like this if it's created using the expression editor:



Override Field Properties in a Form

For each field on the page, you can set some *properties* to override the values set by lower-level rules, including the built-in rules provided by Oracle (as long as they are not locked).

These properties include:

- Required – Make required or optional
- Hidden – Visible or hidden
- Read Only – Editable or read only
- Value – Static or expression

In this context, a *region* is simply a dynamic form.

If more than one rule impacts a given field, it can be tricky to sort out what is finally displayed at runtime to each user group; Understand What Will Be Shown at Runtime can help you.

Remember that the Default rule is always active, which establishes the out-of-the-box behavior. All other built-in and extension rules are essentially *overriding* what is specified in the Default rule. If none of the other rules evaluate to true, then the only overrides applied are those defined in the Default rule.

To set a property on a field:

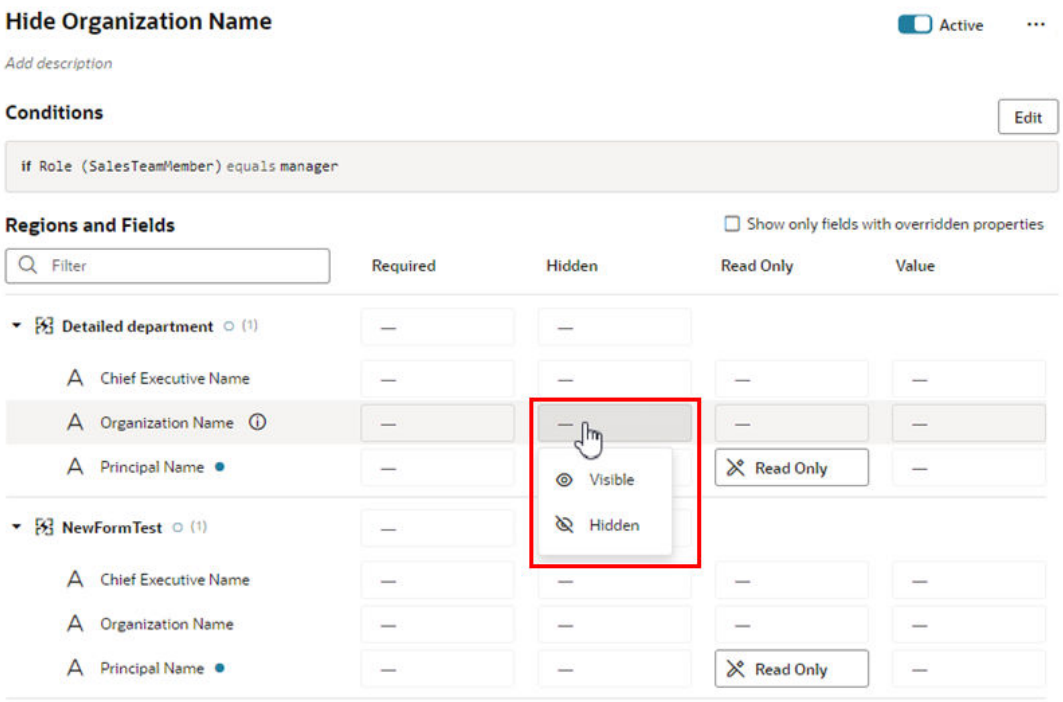
1. In the business rules editor, select the form rule you want to configure.
2. Locate the field you want to modify.

   You can use the Filter field to help locate fields in the editor.

   > **Note:**
   >
   > If you have trouble locating the field you want to modify, or the editor is loading slowly, confirm that there were no errors loading services in the editor. The editor only lists fields from services that have loaded successfully. You'll see "Loading Completed with Errors" in the editor's status bar if any errors occurred when loading the editor.
   >
   > If you see the error message in the status bar, contact your project administrator about resolving the error.

3. Click the dash—or an existing value—in the appropriate column, and select a new value for the property in the dropdown list.

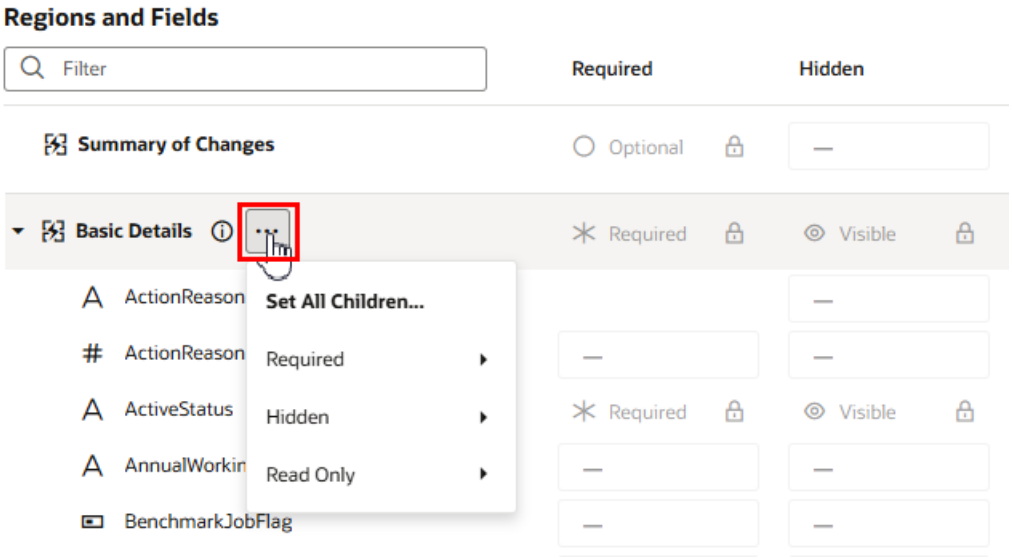
   For example, you can modify a field's Hidden property by selecting Visible or Hidden in the dropdown:



If you change your mind after setting a property, use the **Remove Override** option to restore the property to its original value.

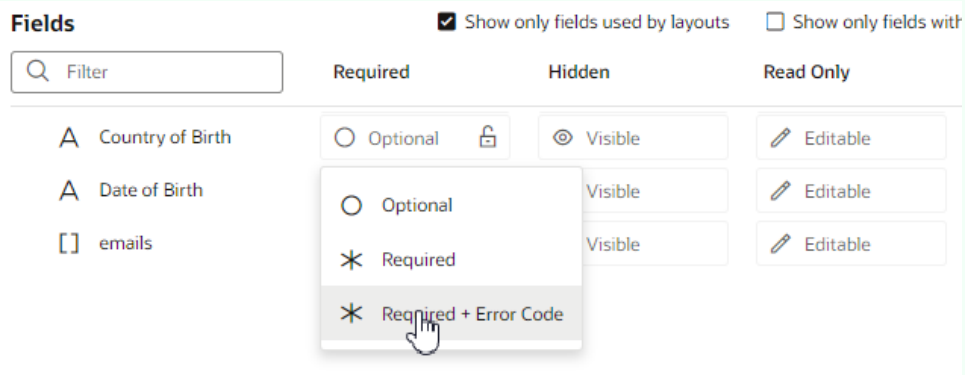
Descriptive Flexfields (DFF) and Extensible Flexfield (EFF) sections (or contexts), when shown, are treated like any other fields; that is, you can set the Required, Hidden, Read Only, and Value properties for them as needed.

If you see child fields when you expand a region or an object field, you can use the Set All Children popup menu to quickly override the properties of all the child fields. For example, you can use the popup menu to set the Hidden property for all the child fields in a region to Hidden instead of setting the property for each field individually. To override a property, click **...** next to the region to open the menu, and then select the property (Required, Hidden, or Read Only) and the override you want to apply:

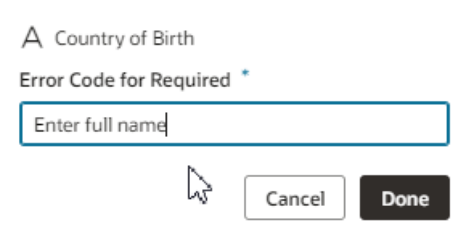


Tip:

When you set a field's property to Required or Read Only, you can configure the rule to display a tooltip on the field by selecting **Required + Error Code** or **Read Only + Error Code** in the dropdown:



You can then enter the tooltip text in the Error Code field:



The Error Code text is displayed in the tooltip if the rule is applied.

VB Studio automatically saves your work for you, so there's no need to do so explicitly.

Note:

A field marked as hidden can still be rendered as visible at runtime. For example, suppose when a page's rules are evaluated, a given field is both required, which means it must have a value, but also hidden. In addition, the field does not have a default value, which means that the user must supply a value explicitly. But how can the user supply a value if the field is hidden? To protect users from encountering this quandary, VB Studio will show the field even though it is marked as hidden, thus allowing users to enter a required value and move on from the page.

Set a Default Value for a Field

Use the *Value* property in a business rule to set a default value for a field.

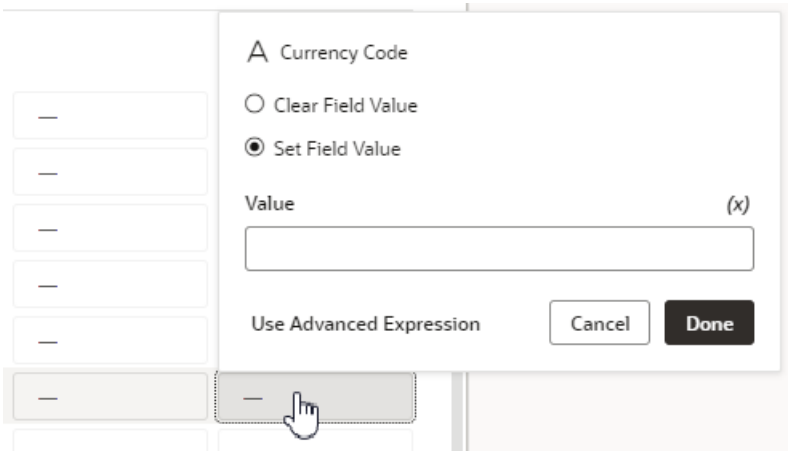
Suppose you have a form in which the user can enter a currency in the Currency Code field. If the user doesn't enter a value—that is, if the field is empty at runtime—you can populate the

field with a default value. You can accomplish this by using the Value property to set the field to, say, "euro", if the user is in Italy. When the user updates the form and clicks Save, the value "euro" is saved in the field, unless the user changes it to something else.

If the business rule sets the field to Read Only, of course, the user won't be able to change it. But if the field is editable, the user can change the value simply by updating the field.

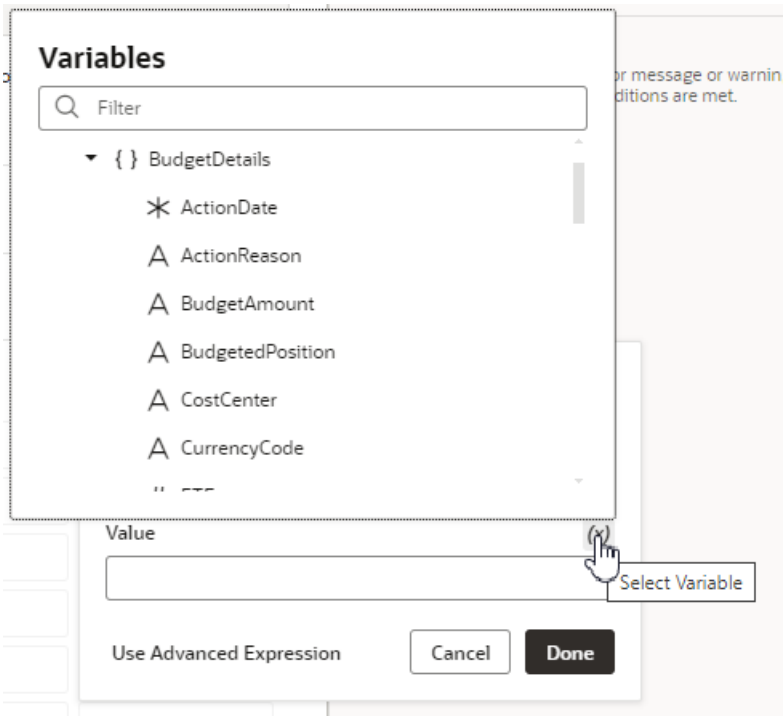
To set a value for a field:

1. In the Fields and Regions editor, find the field you want to modify.
2. Click the dash in the Value column to open a popup dialog for setting the value.



If there's already a value set for the field, click the value and select **Edit Value** in the popup menu. You can also select **Remove Override** in the menu to remove the value.

3. In the popup dialog, do one of the following:
   - Select **Clear Field Value** to remove any default value already set for the field by other rules. For example, you can use this option to remove the value set by the Default built-in rule. When the rule is applied, this option makes the field's value empty (the value is set to `null`).
   - Select **Set Field Value** to enter a default value for the field, which can be:
     - A static value (like "euro"), or
     - An expression which typically uses one or more different variables (like `$fields.BudgetDetails.CurrencyCode.$value()`) to calculate the actual value shown at runtime. When you use an expression, the value is recalculated if a variable referenced in the expression changes.
       To help you create your expression, you can click **(x)** and select variables from the list:



If you want to create a more complex expression, click **Use Advanced Expression** to open the expression builder. See Build Advanced Expressions for more on how to use the expression builder.

> **Note:**
>
> If you see a lock, this means the field has been locked by the Default built-in rule, and you can't override it. Values that should not be overridden might be locked by Oracle.

At runtime, if the business rule's conditions are met, the field will show the value set in the popup.

Set Properties at the Region Level

When setting properties at the region level, there are a few things to note:

- When you set the Required property at the region level, the region will be required for the page, but the fields within the region can be set individually (that is, the setting is not inherited). So, even if you hide all the fields within a region, the region heading will still appear on the page, albeit with no fields shown beneath it.
- If a field is marked Required by Oracle and then locked, the field's containing region will be considered mandatory at runtime, even though the region itself is not officially marked required. This is because a field cannot be shown on a page by itself, without its parent region.
- When you set the Hidden property at the region level, all of the fields within that region inherit that setting. So even if you make an explicit change for a field within the region, the region-level setting takes precedence.
- You cannot set the Read Only or Value properties at the region level.

Locks, Blanks, and Dashes

Let's take a look at what the lock icon, empty spaces, dashes, and grayed text mean in the context of a given rule.

The extent to which these attributes are displayed depends on the type of rule you're looking at. The default rule, for example, always shows the property values as set by Oracle. If nothing has been overridden, the values for the Required, Hidden, and Read-Only properties are always Optional, Visible, and Editable, respectively.

For all other rules:

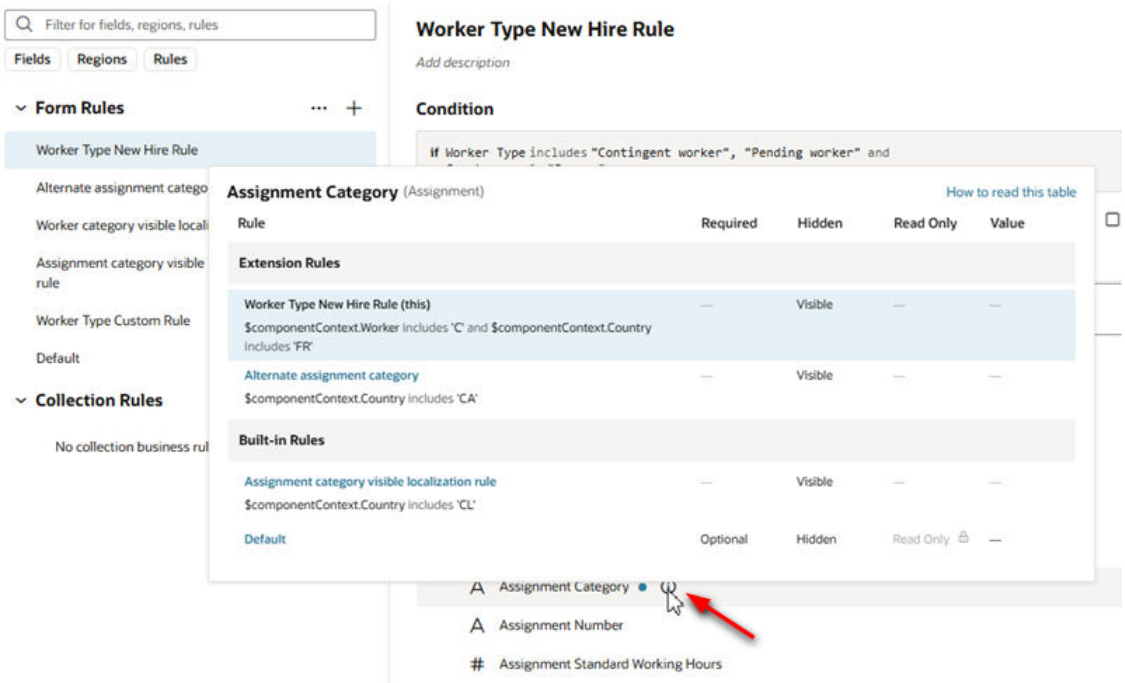
- A dash indicates that a value has not yet been set for the property by this rule. At runtime VB Studio evaluates all the rules from bottom to top, so a property setting for one rule can be overridden—and then overridden again—by rules that are higher up in the list.
- A lock icon means that a value has been locked by Oracle and cannot be overridden. (Any property value that appears in light gray is an Oracle-seeded value.)
- Blank fields represent system fields that are not available for modification by anyone.

Understand What Will Be Shown at Runtime

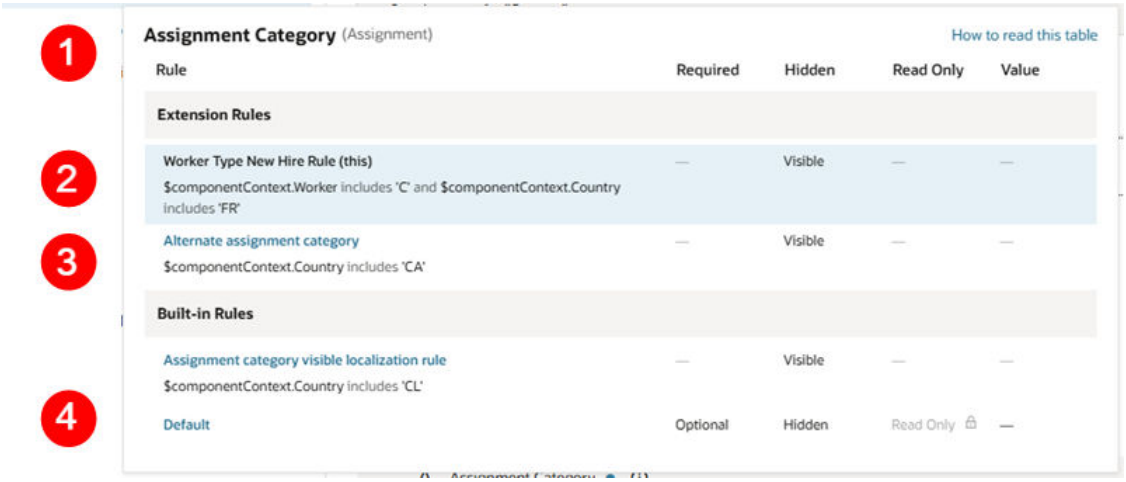
At runtime, VB Studio evaluates all the rules from bottom to top, starting with the Default rule, then moving through the built-in rules, and ending with the extension rules (if any).

As long as a rule's conditions are met at runtime, the rule's property values are applied to the display. However, since rules are evaluated one at a time (starting at the bottom), the topmost rule effectively takes precedence, as it can override what was previously set by lower rules.

To help you evaluate what will be shown for fields and regions at runtime, the editor provides a *pop-up viewer*. To display the pop-up viewer, hover over the field or region until you see an info label, then click it:



The pop-up viewer shows all the *active* rules that modify a region's properties, or a field's properties within the same region:



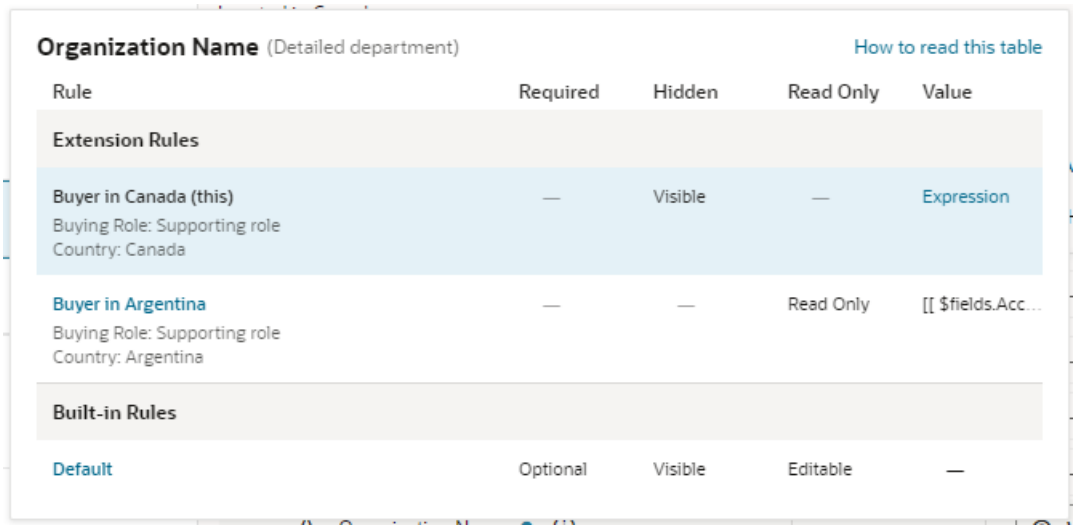
Let's take a closer look at the pop-up viewer:

1. Name of the field (Assignment Category) and the region (Assignment) that these rules pertain to.
2. Currently viewed rule, which is shown with a blue background. The conditions for each rule appears beneath the rule's name.
3. Another rule. If a rule name appears in blue, you can click it to see *all* the fields/regions modified by that rule (as opposed to just the selected field or region).
4. The Default rule, which specifies the out-of-the-box behavior. The Default rule is always active and cannot be modified. And since it doesn't have any conditions, it will always be considered "true".

Interpret the Pop-Up Viewer

Let's examine the rules for the **Organization Name** field to see how it would appear to two different users: first, a buyer in Canada, and then a buyer in Argentina.

The pop-up viewer shows that three rules have modified the **Organization Name** field in the **Detailed department** region:



Let's suppose the current user satisfies the conditions for the **Buyer in Canada** rule, that is, they have the Buying Role "Supporting role" and are located in Canada. What will he or she actually see on the page?

Let's begin by looking at the first rule that is evaluated, the **Default** rule, which is a built-in rule that is always applied. The properties for the **Organization Name** field are set to Optional, Visible, and Editable, and no value is set for the field. None of these properties are locked in the Default rule, so they can all be overridden by rules above it.

Moving up from the Default rule, the **Buyer in Argentina** rule is not enforced because it's conditions aren't met (the current user is in Canada.)

Moving up to the **Buyer in Canada** rule, our current user satisfies the rule's conditions, so this rule will be enforced, overriding the properties enforced by lower rules (in this case, just the Default rule.)

- The **Required** property does not have a value enforced by this rule, as indicated by the dash mark. Moving down the **Required** column, the next rule (Buyer in Argentina) doesn't enforce a value for this property because its conditions aren't met, so we look to the Default rule, which enforces a value of Optional. So, for the buyer with the "supporting role" located in Canada, the **Organization Name** field is optional because no rule overrides the value set in the Default rule; in other words, the user doesn't have to supply a value for this field before submitting the form.
- Now let's scan the **Hidden**, **Read Only** and **Value** columns in the rule. Two of these have values that are enforced by the rule, overriding values set in lower rules; that is, the Hidden property is set to Visible, and the Value property is set to an expression. The rule doesn't set a value for the Read Only property.

For buyers with a "supporting role" in Canada, then, the **Organization Name** field will be visible and its value set to an expression (as the field is not Read Only, the user has the power to change the value.)

Now let's see what happens at runtime if the user is a buyer with the "supporting role" but located in Argentina, rather than Canada. Once again we start with the Default rule, which sets the properties described above.

Then we move up to the next rule, **Buyer in Argentina**. This time, both conditions are satisfied, so let's look at what this rule does:

- Once again, this rule does not enforce a value for the **Required** property. Continuing down the column we reach the Default rule, which states that the **Organization Name** field is Optional for these users.
- Likewise, the **Hidden** property does not have a value, so we take the value from the Default rule, Visible.
- The **Read Only** property is set to Read Only, overriding any value set in lower rules (just the Default rule, in this case).
- The **Value** property is set to an expression (this expression is different from the one set in the Buyer in Canada rule.)

Moving up to the **Buyer in Canada** rule, this time the conditions for this rule are NOT satisfied, because the user is not in Canada, so this rule is not enforced.

In summary, buyers with the "supporting role" in Argentina can see the **Organization Name** field, but, unlike their counterparts in Canada, it is read only, and its value might be calculated differently.

What Do the Blue Indicators Mean?

Indicators in the **Regions and Fields** area help you see at a glance what has been changed.

In this example:

1. An empty blue circle next to a region (**When & Why Section**) indicates that while the region itself does not have a property overridden, at least one field in the region does.
2. The number in parentheses next to the region indicates the number of fields in this region that has overridden properties.
3. A blue dot next to a field or region indicates that there is at least one overridden property at this level.

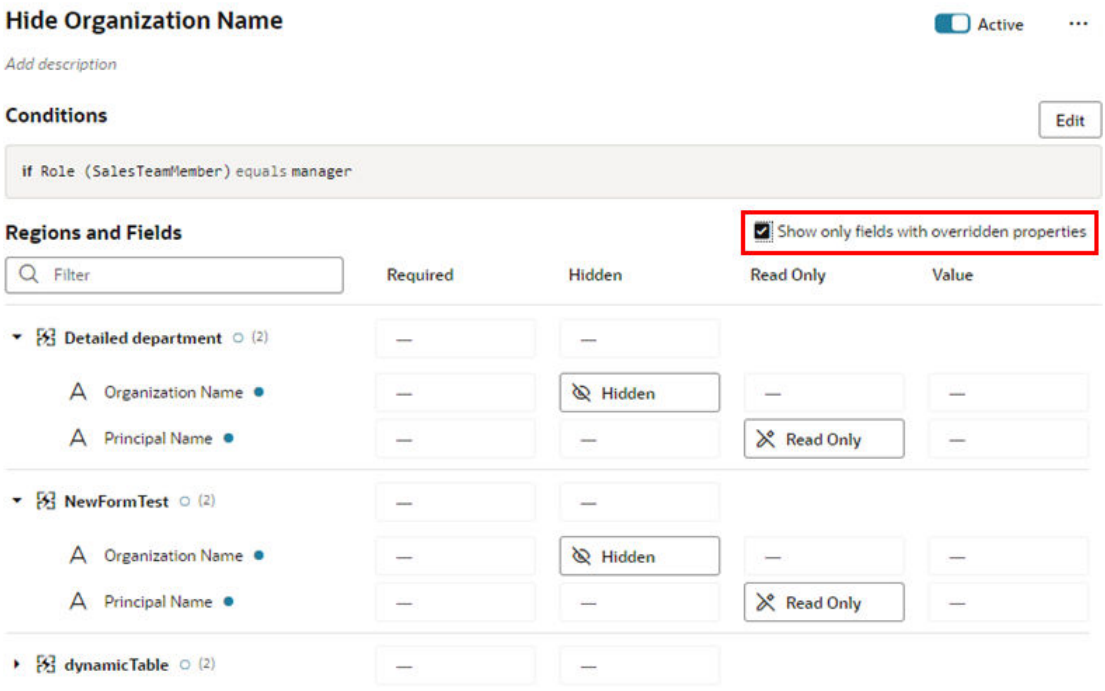
Identify the Regions and Fields Impacted By a Rule

Use **Show only fields with overridden properties** to understand which regions and fields are impacted by a rule, and how are they impacted.

To see the regions and fields that are affected by a rule:

- Select a rule, and then enable **Show only fields with overridden properties**.

The list is filtered to only display the regions and fields affected by the selected rule. This lets you focus just on the properties that have been overridden by the rule. In this example, the list only contains fields impacted by the **Hide Organization Name** rule.

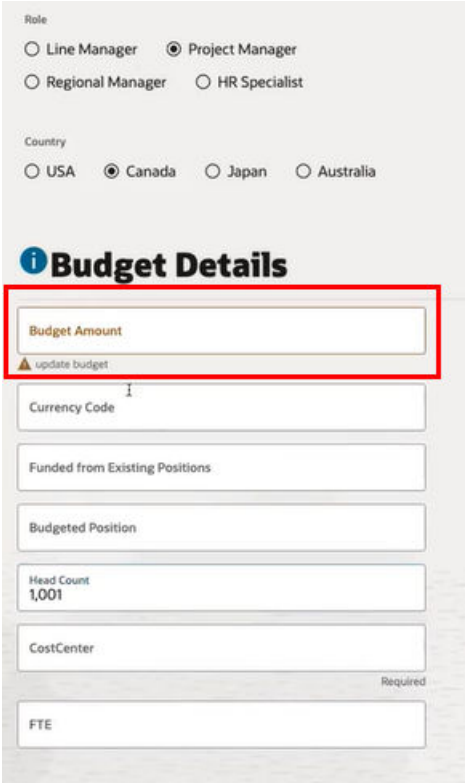


Display Messages in a Form

You use a validation rule to display a message when certain conditions are met. When you create a rule, you define the conditions for when the rule should be applied, and define the message that should be displayed.

Validation rules are set at the object level, rather than for a specific page. This means that when you define a validation rule for a form—for example, a form for editing the BudgetDetails business object—the rule is applied on every page where that form is used and the conditions are met.

Validation rules are particularly useful when you want to display some type of warning message based on data entered in a form. Suppose you want to display a message reminding the user to update the Budget Amount when the Head Count is more than 1000. You can create a rule that checks the value entered in the Head Count field, and display a message like this in the form:

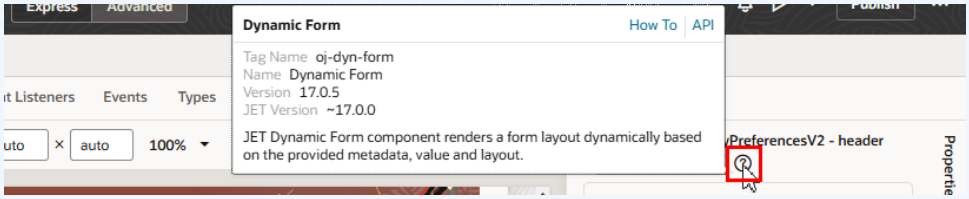

Role
Line Manager
Project Manager
Regional Manager
HR Specialist
Country
USA
Canada
Japan
Australia
Budget Details
Budget Amount
update budget
Currency Code
Funded from Existing Positions
Budgeted Position
Head Count
1,001
CostCenter
Required
FTE

Note:

Validation rules only display messages in forms rendered using the `oj-dyn-form` form component. If your application has any forms rendered using the older version of the form component (`oj-dynamic-form`), validation rules won't work correctly in those forms.

To check which version of the form component is used in a page, you need to view the page in Advanced mode:

1. Click Advanced in the header to open the page in Advanced mode.
2. Select the form component to open the component's details pane in the Properties pane.
3. Hover your cusor over the component's name to view the Info icon ( ), then move your cursor over the icon to open the component's Info popup.



In the example above, the popup shows this form is rendered using the `oj-dyn-form` form component.

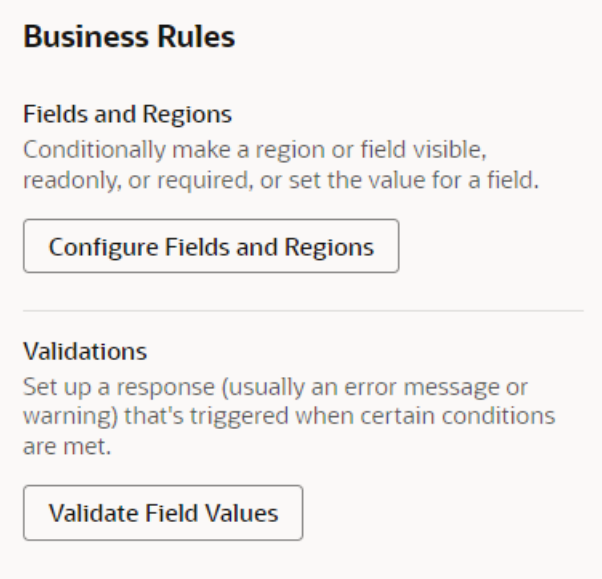
4. Click Express in the header to return to Express mode.

Create a Rule to Validate a Field

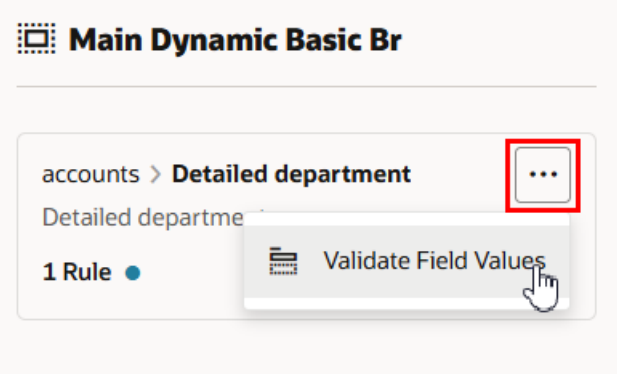
Let's take a look at how to create a rule that works like the example above, displaying a message in a form when the value for the Head Count field is over 1000.

To create the validation rule:

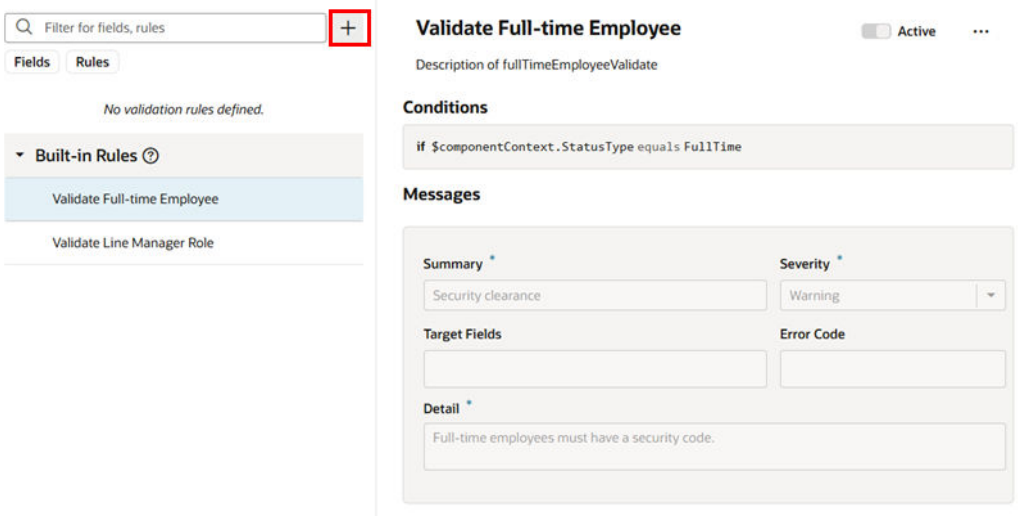
1. Open the page containing the form, and then click **Validate Field Values** under Business Rules in the Properties pane.



If you see a dynamic component's details pane like this in the Properties pane instead of the Business Rules pane, click **...**, and then select **Validate Field Values** in the menu:



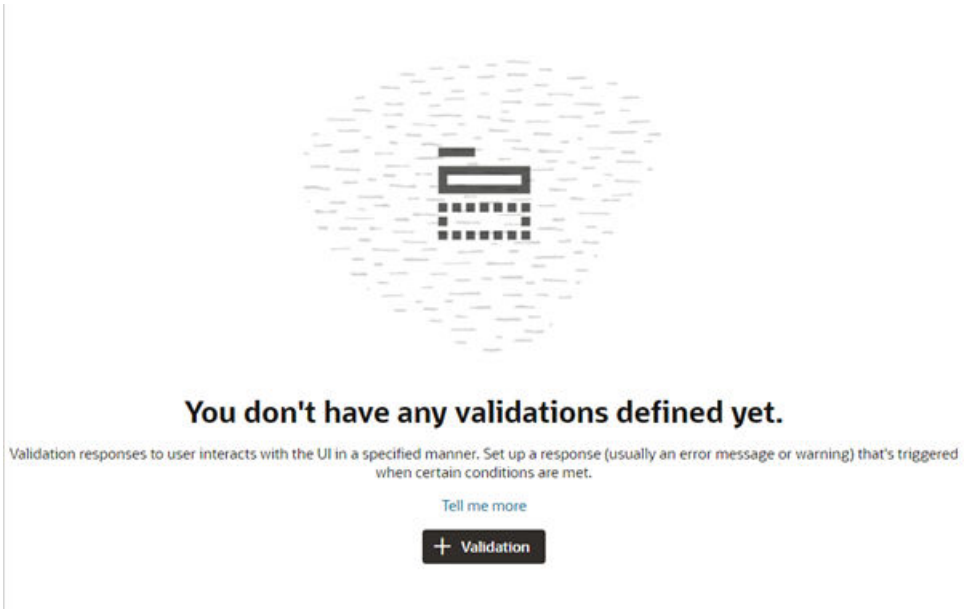
2. Create a new rule.
   - If some rules are already defined, click + to open the Create Validation dialog box.



Rather than creating a rule from scratch, you can duplicate an existing rule and use it as the basis for a new rule. To duplicate a rule, right-click the rule in the list, and then click **Duplicate**.

> **Note:**
>
> When duplicating rules defined in extension dependencies (the rules with the Built-in label):
>
> – You cannot duplicate a rule if it uses advanced expressions, for example, in the rule's condition or to override a field's value.
>
> – You cannot duplicate a rule if it uses any global functions that cannot be referenced in the extension. If the global functions are referenceable, when the rule is duplicated they are automatically imported, and the references to the function are updated in the rule.
>
> – When duplicating nested rules, you can only duplicate the top-level rule. When you duplicate the rule, the child rules are also duplicated.

- If there are no rules yet, click **+ Validation**.



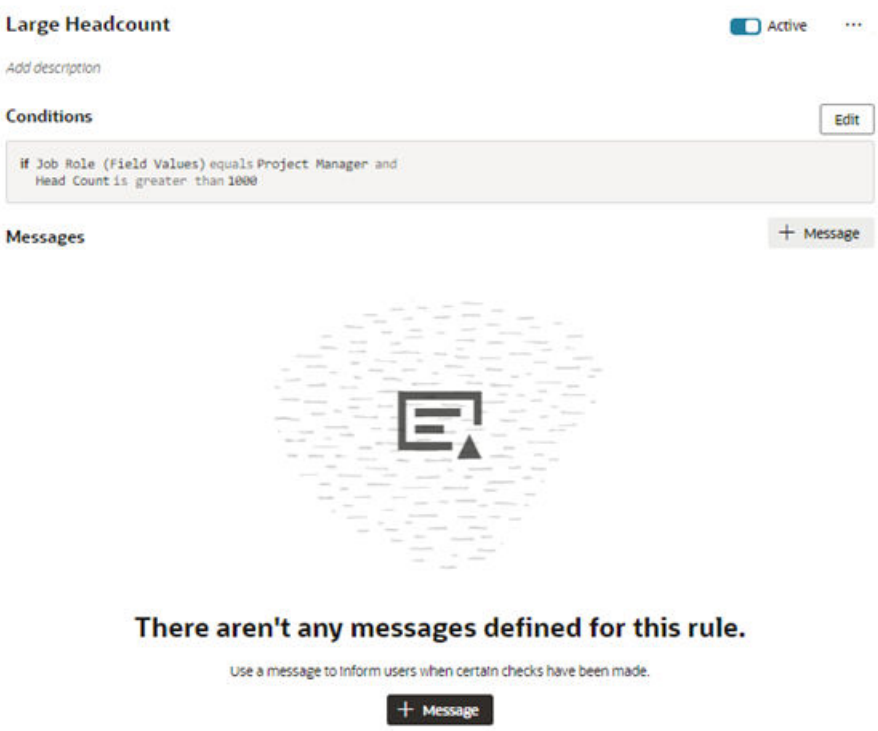
3. Select the **Regular** Rule Type in the dialog box, and then enter a label, id, and description. Click **Create**.

   In the Create Validation dialog box, you can select a validation rule using the Template dropdown list (if any are defined in the app), or choose a rule type (Regular, If/else, or Switch). For more, see Add an If/Else Rule and Add a Switch Rule.

4. Click **Edit**, then specify the rule's conditions. Click **Done** when you're finished.

   You create conditions for validation rules using the standard condition builder, or you can click **Use Advanced Expressions** to use the advanced expression builder if you want to create more complex conditions.

   For this rule, we want to create two conditions: the Job Role must be 'Project Manager', and the value for Head Count must be greater than 1000. The rule will be applied when both these conditions are met:



5. Specify the message details.

   When the conditions are met, we want the warning message "Update budget" displayed under the Budget Amount field in the form.

   a. Click **+ Message**.

   b. Type the message texts in the **Summary** and **Details** fields.

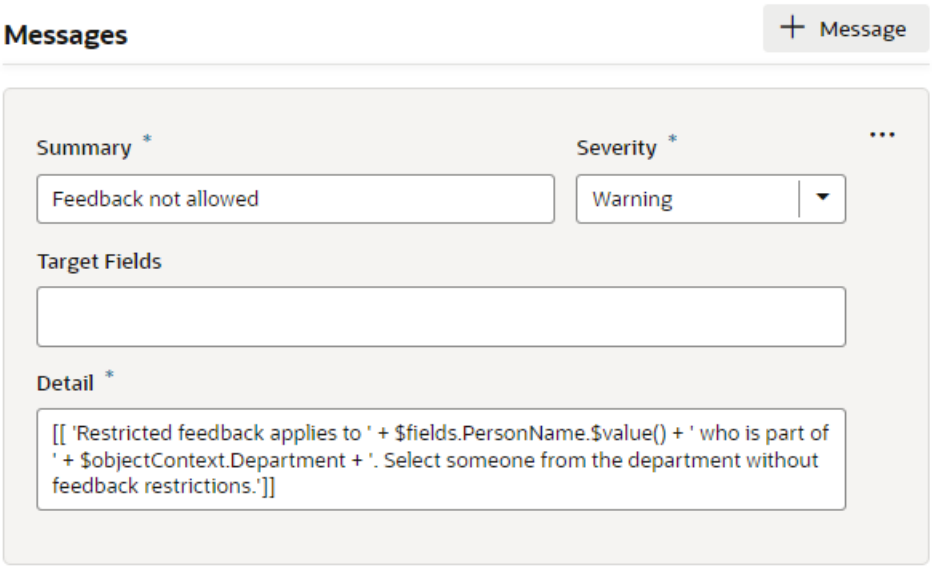
      You use these two fields to enter the text for the message.

      - **Summary**. The Summary text is displayed in the title of a message dialog. The text in the Summary field is not displayed on the page if you are using an inline warning, but it's still a required field.
      - **Detail**. The Detail text is the actual text message. This text is displayed inline in the form if you are displaying an inline warning, and in a message dialog if it's not displayed inline.

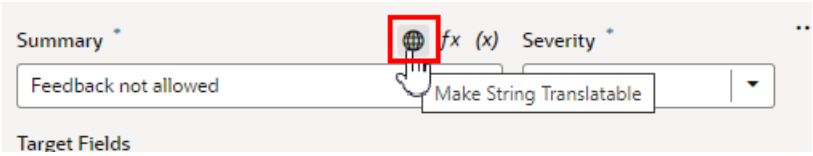
      Your message text can be a simple string, like "This number is invalid", or you can write more targeted messages by passing field values and context parameters into the message. For example, you could add something like this in the Detail field:

      ```
      [[ 'Restricted feedback applies to ' + $fields.PersonName.$value() + ' who is part of ' + $objectContext.Department + '. Select someone from the department without feedback restrictions.']]
      ```

      To create a message that includes different types, such as variables or objects, you need to surround the message with `[[ ]]`, so that it's treated as an Object and evaluated, and not treated as plain text. To construct the message, any static text has to be enclosed in quotes, and you need to use a `+` operator to concatenate the static text, variables, and other entities:



To generate translations of the Summary and Detail texts, you can hover over the text field and then click to open the Translatable String popup:



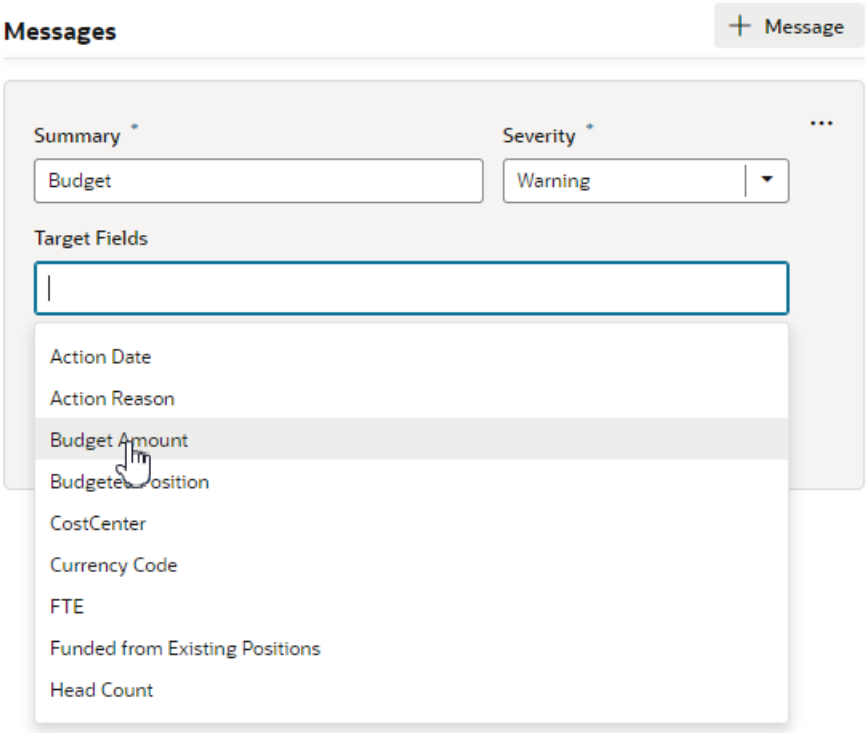
It's not possible to add parameters in the Translatable String popup, but you can click fx to open the Expression Editor, and then manually edit the translation object if you want to use parameters.

c. Select the Warning in the **Severity** dropdown menu.

   The Severity menu contains the following options:

   - **Error**. Choose this when there is some data that the user *must* correct before they can submit the form. This is the highest severity level.
   - **Warning**. Choose this to call attention to a field, for example, if you want to let a user know to check data that was entered. A warning message won't prevent the user from interacting with the page.
   - **Info**. Choose this for messages that are only informative.
   - **Confirmation**. Choose this for messages that confirm an operation or task was completed. This is the lowest severity level.

d. Select the field where you want the inline message displayed in the **Target Fields** dropdown menu.

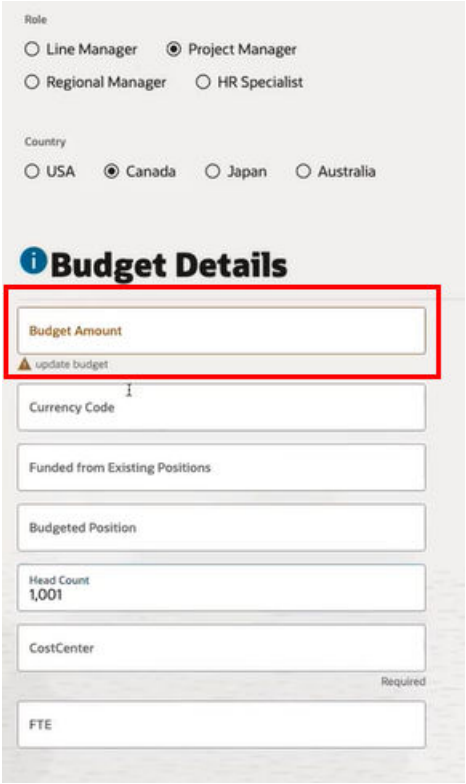


If you do not select a target field, and the form contains only one editable field, the message is automatically applied to it.

> **Note:**
>
> A message is not displayed inline when:
>
> - You do not select a target field, and the form has multiple fields.
> - You select more than one target field.
>
> If you want messages displayed in a dialog box instead of inline, a message component in the page needs to be manually configured to handle the messages.

To add a different message to the rule, say a message with different text displayed under a different field, click **+ Message** to create a new message and specify its details.

To check if your rule is working, view the page in Live view and test the form by entering values to trigger the rule.

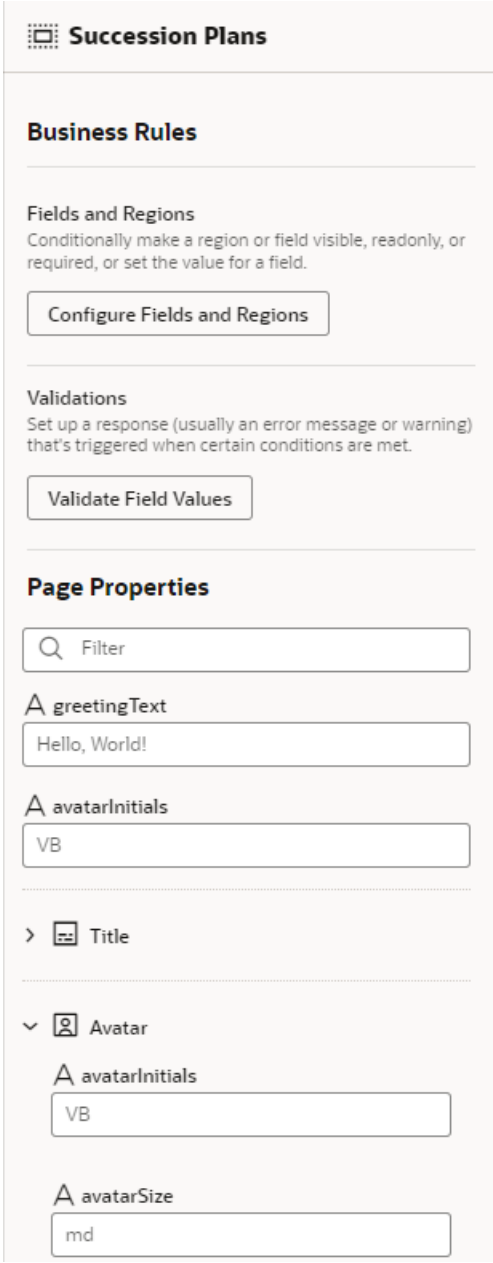

Role
Line Manager
Project Manager
Regional Manager
HR Specialist
Country
USA
Canada
Japan
Australia
Budget Details
Budget Amount
update budget
Currency Code
Funded from Existing Positions
Budgeted Position
Head Count
1,001
CostCenter
Required
FTE

3

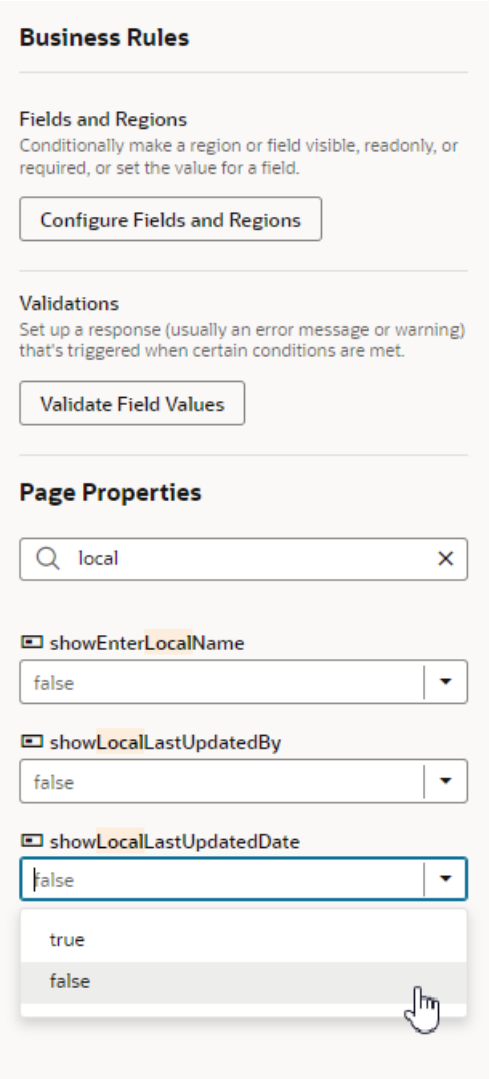
Work With Page Properties

Page properties store values or expressions that are used in the page, and can also be used to control what is displayed in a page. When you open a page that has editable page properties, you can edit them directly in the Properties pane.

The editable page properties are listed in the Page Properties panel in the Properties pane. In the example below, the first page property is greetingText, which you can change by typing a new text in the field. Some page properties might be organized for you into sections in the Page Properties panel. In this example, the page properties related to the avatar have been grouped in the Avatar section:



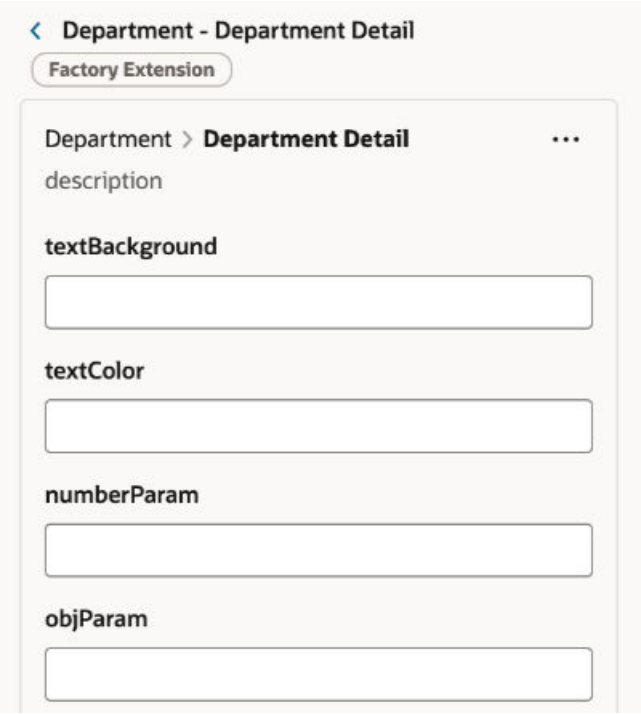
If the list hasn't been organized for you, the page properties are displayed as a simple list. The list can be quite long, so you can use the Filter field to help you locate specific page properties. Here the list has been filtered to only show the page properties containing "local":



Page properties can be used to control what is displayed in the page. In the image above, you can see how `showLocalLastUpdatedDate`, used to control the `LocalLastUpdatedDate` field, can be set to 'true' or 'false'.

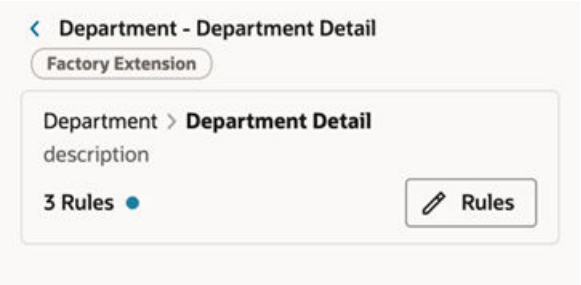
> **Note:**
>
> Oracle ensures that each page property is thoroughly documented (at least in the Human Capital Management Cloud Application), so be sure to use these descriptions as a resource as you work with page properties. It's a good idea to emulate this practice when creating your own page properties as well.

If a dynamic component in the page uses a fragment, you can edit the fragment's input parameters in the Page Properties pane. When a dynamic form or table has only one display rule (the default rule), or a rule that is always applied, you can edit fragment input parameters directly in the Properties pane when the component is selected:



If you'd like to set different input parameter values, based on conditions, you can create new rules in the rules editor, and specify the input parameters. To open the rules editor, open the options menu in the Properties pane (…) and click **Add Rule**.

If the component has more than one display rule that could be applied, the details pane has a link to open the component's rule set editor where you can edit the fragment's input parameters:



Set Property Values Using Rules

In some pages, you can use rules to set values for extendable page properties. In the rules editor for page properties, you create rules and conditions for setting values, much like you do when using rules to override property values for fields in forms and tables.

For example, your page might use a Guided Journey Page Level Code property, and the dependency developer has made the property extendable. You can create rules to set the property's value based on conditions, such as the user's role. The value is applied only when the rule's conditions are met.

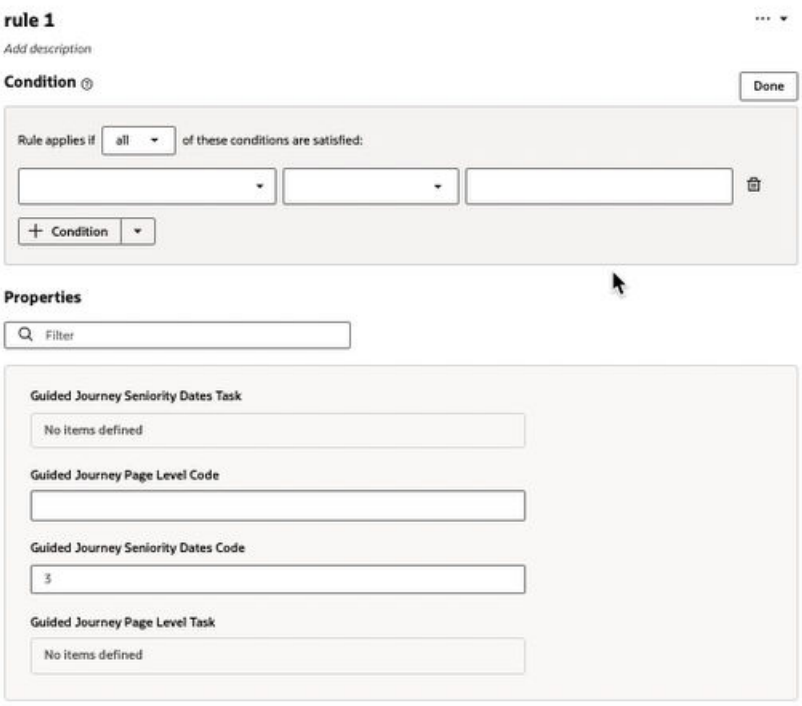
The following steps illustrate how to use the page property editor to set a property's value.

1. Open the page you want to modify.
2. In the Business Rules pane, under Page Properties, click **Configure Page Properties** to open the Page Properties editor.
   When you open the Page Properties editor, click **Start Preparing** if the editor displays a message "We need to prepare your page properties".
3. In the editor, click **+ Rule**, and then enter a name and description in the Create Page Properties Rule dialog box. Click **Create**.
   Your new rule is added at the top of the list of rules in the editor.
4. Move the rule into the position you want it evaluated.
   To change a rule's position, right-click the rule and use the Move up and Move down items in the popup menu.

   Like business rules, the order in which rules appear in the list is important. Rules in the Page Properties editor are evaluated at runtime from top to bottom. So if a property value is applied by the topmost rule in the list, that value might be replaced by a value applied by a rule below it. However, the rules you create always have precedence over the rules written by the dependency developer, so your rules are evaluated after the built-in rules.
5. Select the new rule in the list of rules, then click **Edit** to open the condition builder.
6. Create the rule's condition.
   You create conditions in the page property editors just like you set rule conditions for fields in forms and tables.

   > **Note:**
   >
   > You cannot use the visual expression builder to create conditions for page property rules.

7. In the editor's list of Properties, set values for the page properties that you want to be applied when the condition is met.

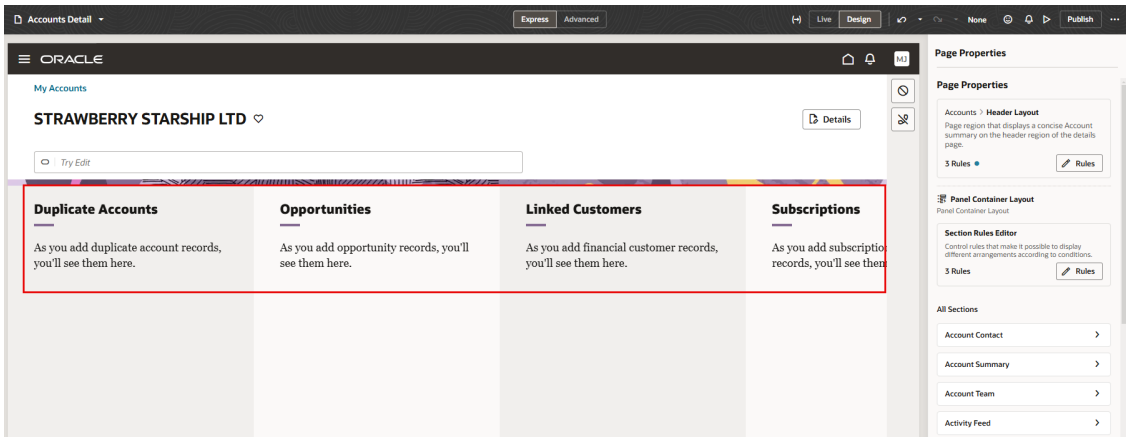

rule 1
Add description
Condition
Done
Rule applies if all of these conditions are satisfied:
Condition
Properties
Filter
Guided Journey Seniority Dates Task
No items defined
Guided Journey Page Level Code
Guided Journey Seniority Dates Code
3
Guided Journey Page Level Task
No items defined

4

Work With Containers and Sections

Some pages use containers to display various types of content. Containers are pre-defined areas in a page that let you display content grouped into logical regions called *sections*. These sections are displayed within containers, which you can use to rearrange the content already placed there. You can also add and remove sections within containers.

Here's an example of a page that has a foldout-style container with four sections next to each other, each displaying different content:

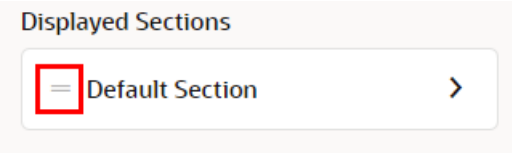


Now let's look at how to configure the sections displayed in a container.

Control the Sections Displayed on the Page

You can change the sections in a container directly in the Properties pane, or use the container's rules editor if you want to use conditions to define which sections are displayed.

If you don't need to define any rules or conditions in your container, you can modify the container's built-in rule directly in the Properties pane, to add, remove, and change the order of sections. This means that the sections listed under Displayed Sections in the Properties pane are always displayed. You can modify the container in the Properties pane if you see sections with handles listed under Displayed Sections in the Properties pane:



If you see All Sections instead of Displayed Sections in the Properties pane, this means the container already uses some rules in addition to the built-in rule, and you need to define the container's sections in the rules editor.

To change the sections displayed in a container:

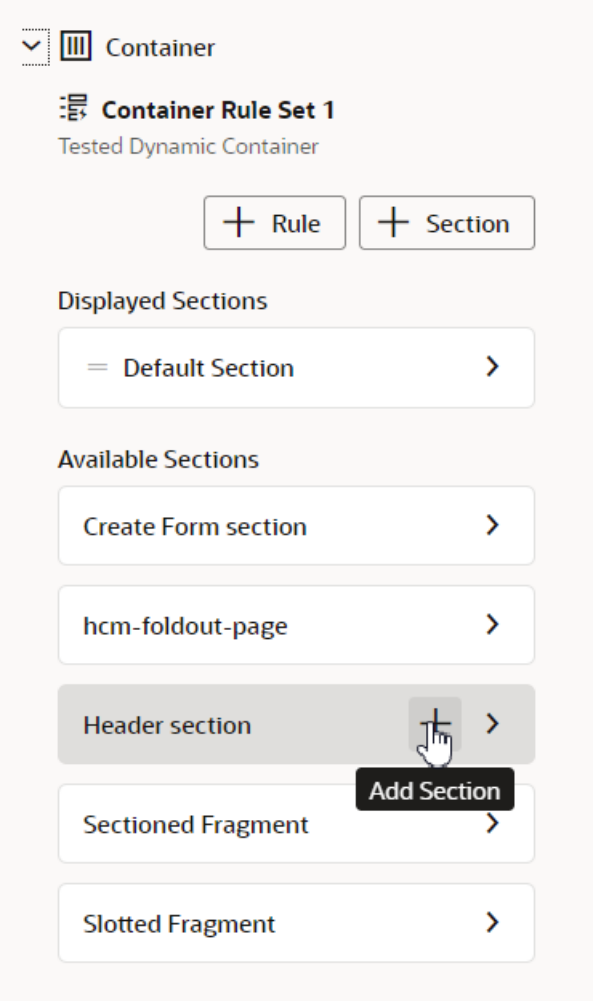
1. Locate the container in the Properties pane.

   The Properties pane lists all the sections that you can display in the container under Available Sections. These sections are usually defined in dependencies, but also includes sections defined in your extension.

2. Define the container's sections without creating rules and conditions:

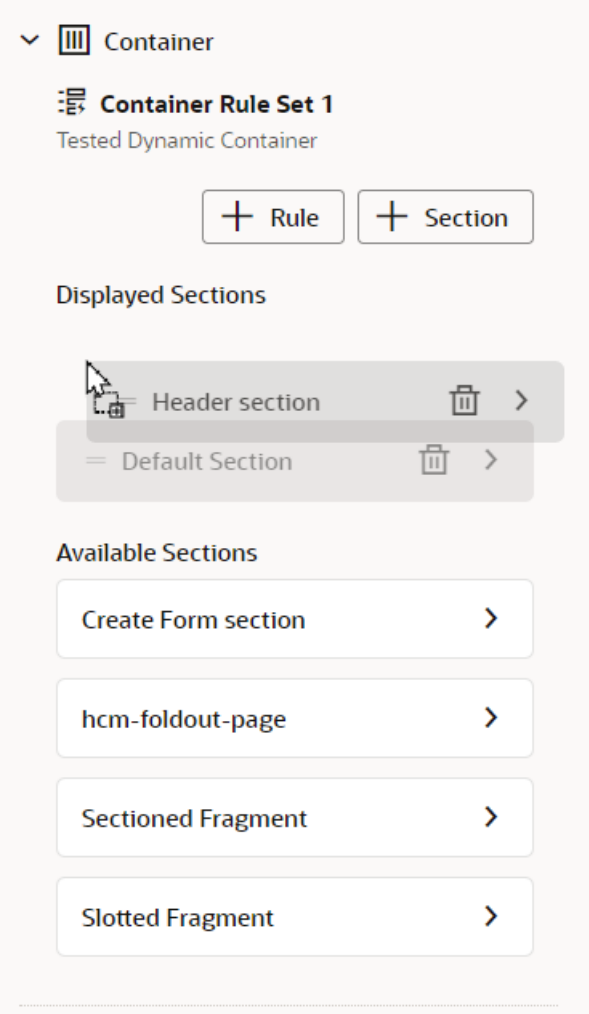
   If the container already uses rules, skip to step 3 below to define the sections using the rules editor.

   a. Add a section by hovering over the section in the list of available sections and clicking +.



   The section is added to Displayed Sections.

   b. Re-order the sections under Displayed Sections by grabbing a section's handle and moving it into the position you want.



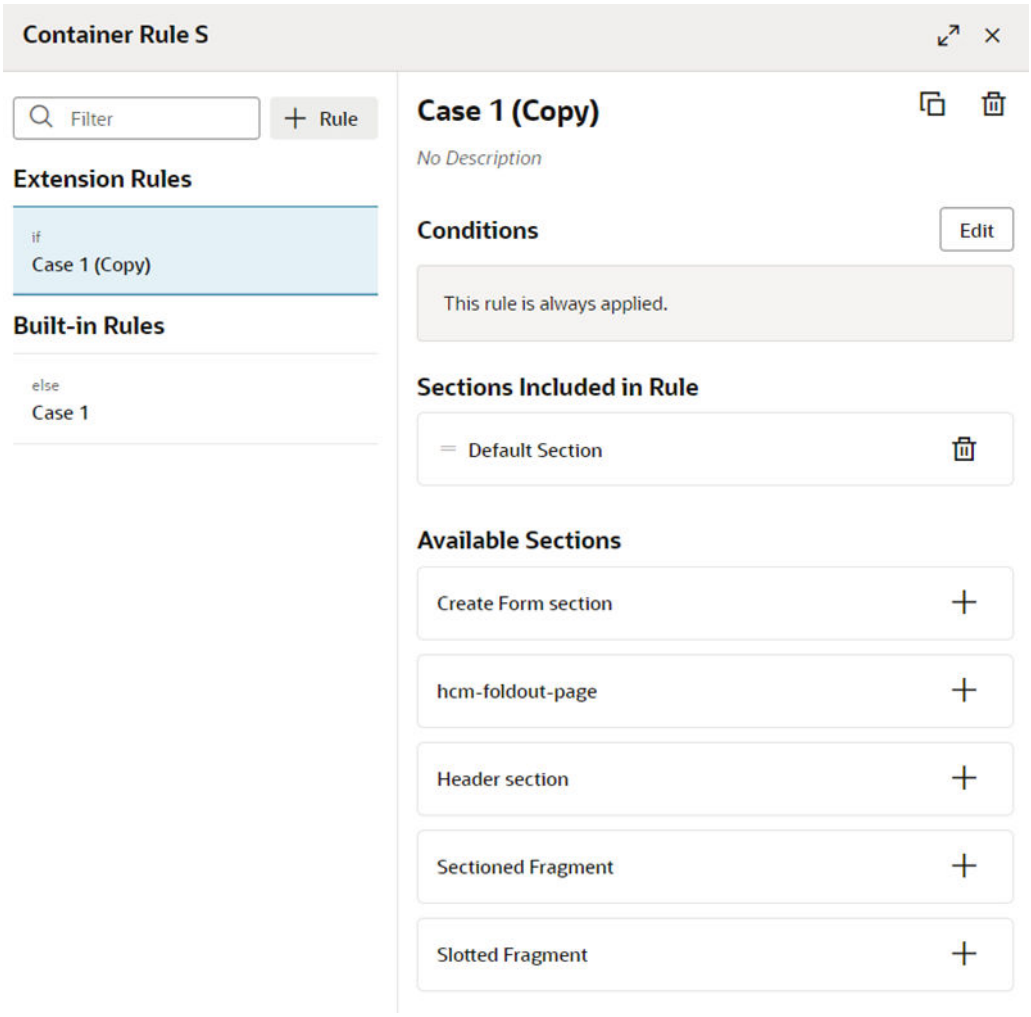
c. Remove a section from Displayed Sections by hovering over the section and clicking .

3. Define a container's sections using rules and conditions:

a. In the Properties pane, click Rule to create a rule, or **Rules Editor** to open the editor.

The rules editor displays a list of rules for determining the sections displayed in the container. Rules listed under Built-in Rules are defined in the extension dependency, and you cannot edit them.

When you create a rule, it is added to the list of Extension Rules:



To duplicate a rule, right-click the rule in the list and select **Duplicate** in the popup options menu.

To delete a rule, right-click the rule and select **Delete** in the options menu.

b. Change the order your extension rules are listed by selecting the rule, and then dragging it into the position you want.

   Rules are evaluated in order, from the top down. If a rule's conditions are met, the rule is applied, and none of the rules below it are evaluated. When you create a rule, it is added to the top of the Extension Rules list, so you need to make sure your rules are in the correct position.

c. Define the rule's conditions in the condition builder by selecting the rule, and then clicking **Edit**.

   The condition builder is the same one used to create conditions in business rules. If you're not familiar with using the condition builder, see Set Conditions for a Rule.

d. Choose the sections you want displayed in the container when the rule's conditions are met.

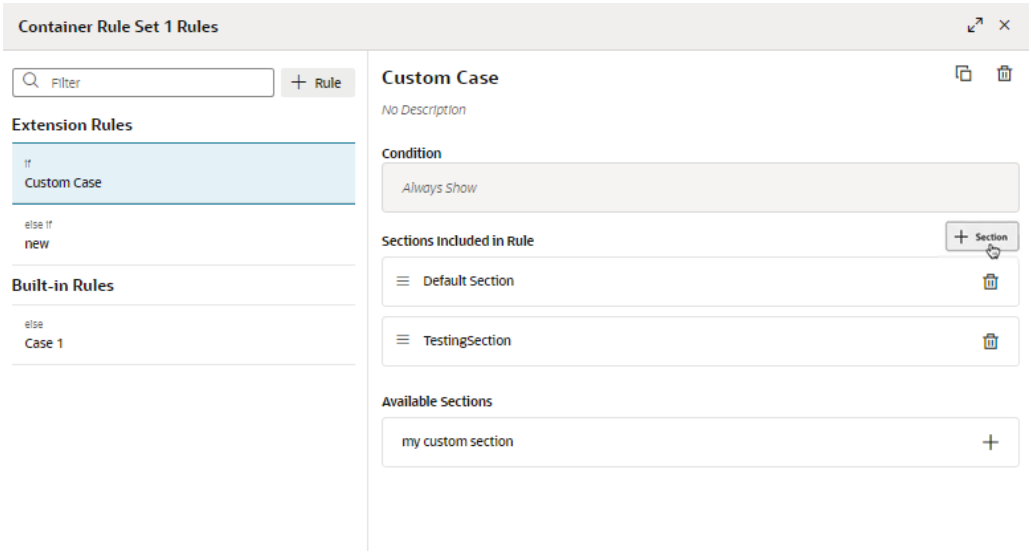
   - To add a section to the list of displayed sections, click the section's + button in the list of Available Sections.
   - To remove a section, click 🗑 .

To change the order the sections appear in the container, grab the section's handle next to the section name and move it into position.

4. Create new sections in the container.

   You can create sections if a 'template' for new sections is defined for the page in the dependency. New sections are based on this template. You can't create sections if no template is defined.

   a. Click **+ Section**.

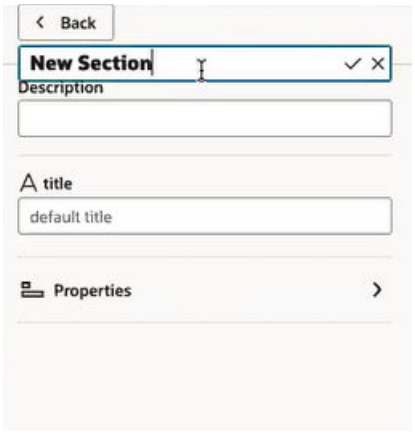


   When you create a section, it is added to the list of Available Sections in the rules editor, and to the All Sections list in the Properties pane.

   b. Click the new section in the All Sections list in the Properties pane.

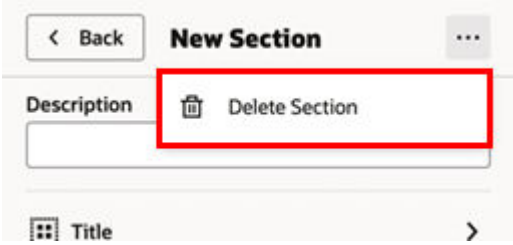
   c. In the Properties pane, edit the section's name, description, and properties.

   To edit the section name, hover over the name and click :



   Sections might have other editable properties. These properties are based on the template used for creating sections in the container.

   You can delete a section you created by opening the section's details pane in the Properties pane, and then selecting **Delete Section** in the options menu.



d. Click **Back**.

5. Close the rules editor.

Configure the Content Displayed in a Section

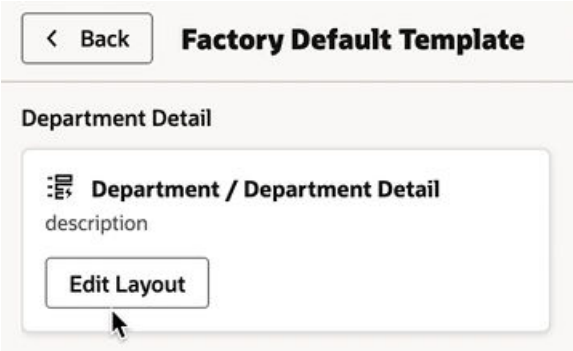
Some sections have content that you can modify. For example, a section might contain an editable fragment, or a form that you can edit to hide or add fields.

To configure the content displayed in a section:

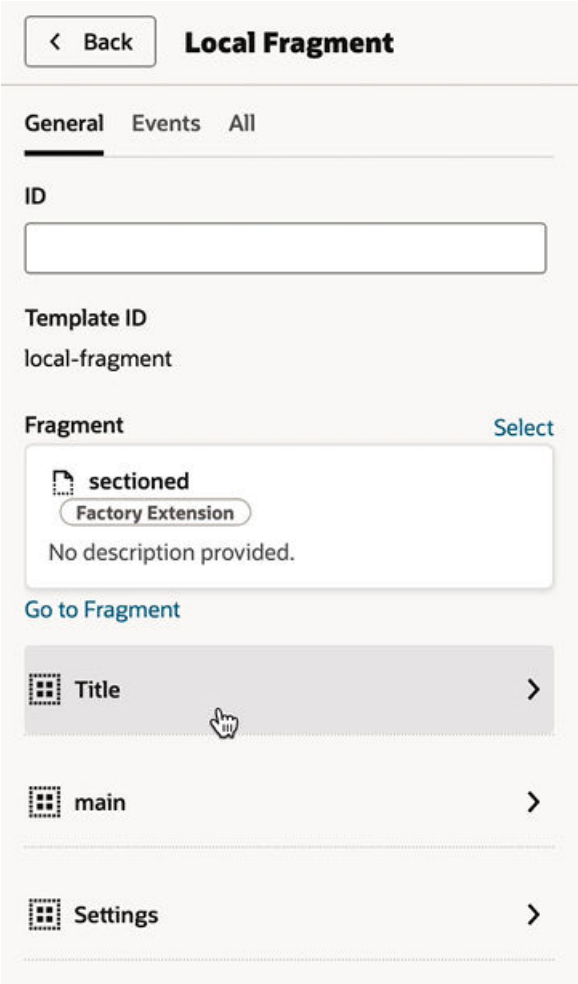
1. In the Properties pane, select the section you want to modify.

   The Properties pane displays the section elements that you can configure.

   In this image, the Factory Default Template section contains a rule set that you can modify. When the section contains an editable rule set, click Edit Layout to open the Rule Sets editor where you can modify what's displayed in the section:



   In this section containing a fragment, you can edit the fragment's editable properties in the Properties pane:

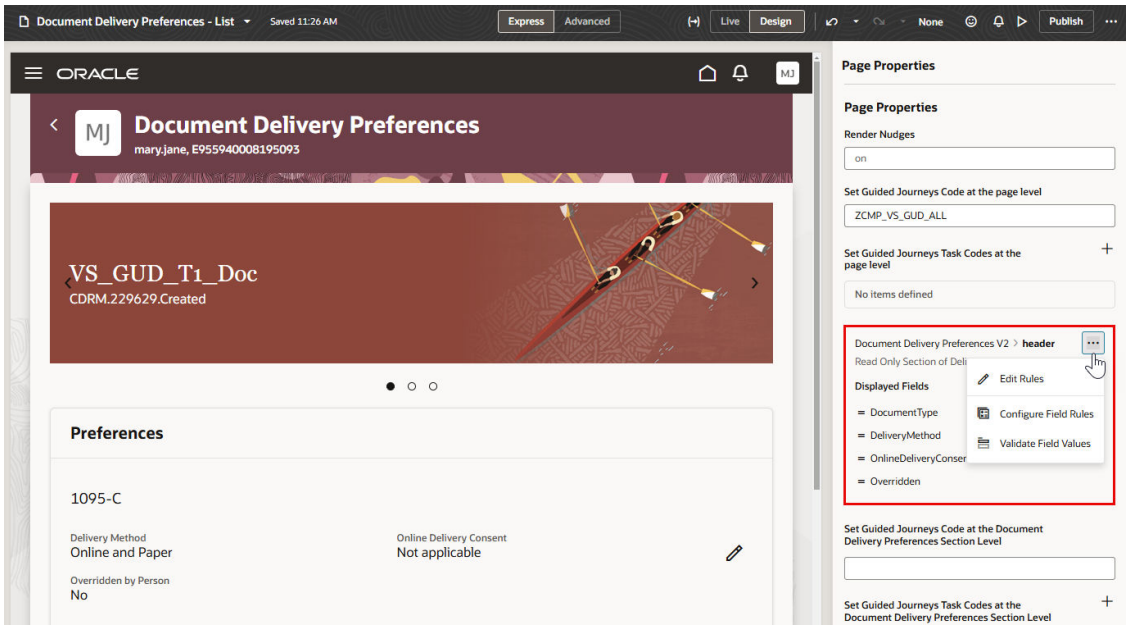


2. Click **Back** to return to the page's Properties pane.

5

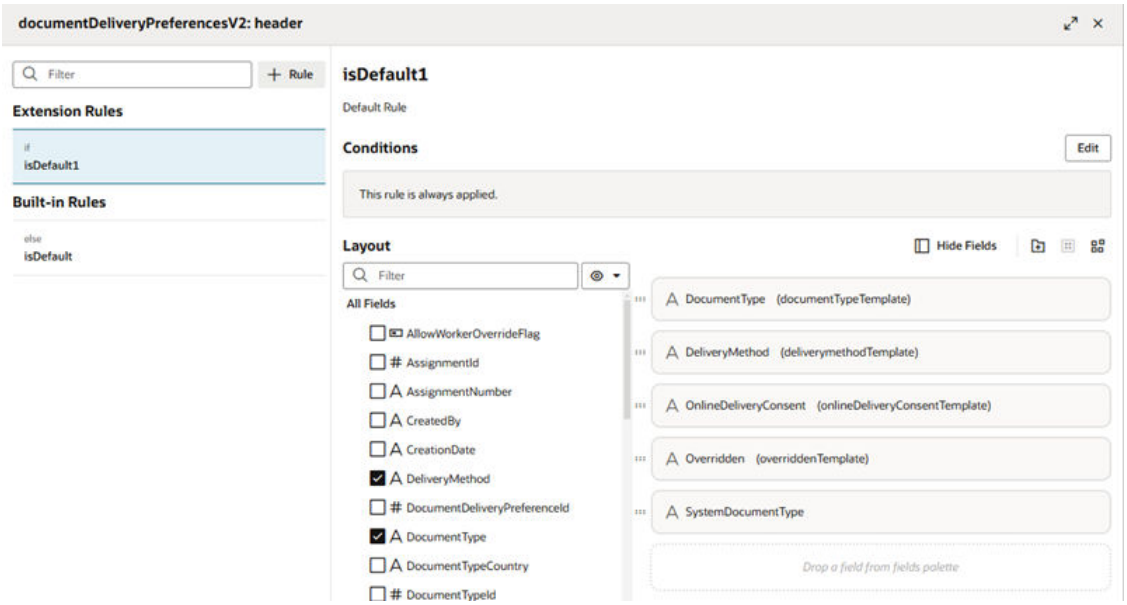
Control Your Display With Rule Sets

Depending on how your Oracle Cloud Application page was built by Oracle, you may land on a view of the Designer where the Properties pane has a list of layouts for the page's dynamic tables or forms instead of a Business Rules pane, as shown here:



Dynamic table and form components display fields stored in a data source, such as a business object. The fields or columns displayed by a dynamic component, as well as the order and how they are rendered in the page, are defined using layouts.

Each dynamic component has a *rule set*, which is a set of rules that determine which layout is used at runtime. You use the rule set editor to edit the rules and layouts defined in the rule set. This image shows what a rule set looks like when you open it in the editor:



In Express mode, it's likely that your tasks will be limited to creating rules and the layouts that are applied when the rule's conditions are satisfied. This chapter describes how you can configure rule sets in Express mode.

If you need to perform more sophisticated tasks and configurations, you can switch to Advanced mode, which provides more features for configuring extensions. For more on what you can do in Advanced mode, see Customize Dynamic Tables and Forms in *Extending Oracle Cloud Applications with Visual Builder Studio*.

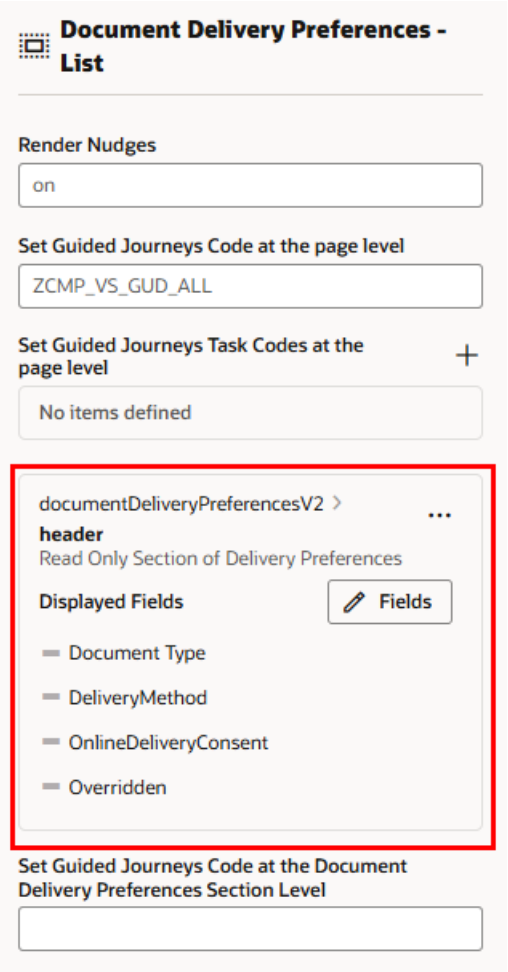
Configuring the Fields Displayed in the Component

Many dynamic forms and tables have rule sets with only one default rule. When this is the case, you can change the fields displayed in the component directly in the Properties pane, without opening the rule set editor.

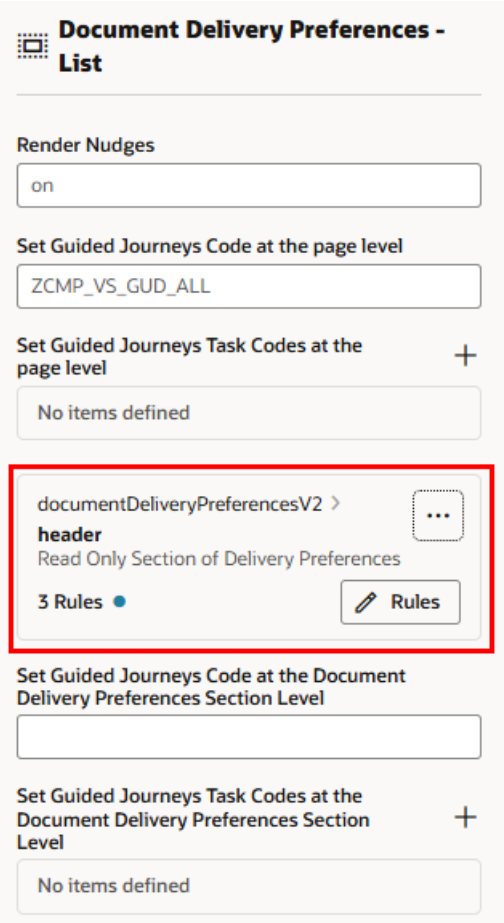
To edit the layout in the Properties pane:

1. Open the page you want to modify in the Designer.
2. Locate the component you want to edit in the Properties pane, or select the component on the page to open the component's details pane in the Properties pane.

   If the component's details pane looks something like this, the component's rule set has one rule, which is always applied, and you can edit the layout to add, remove, and change the order of the fields directly in the Properties pane:

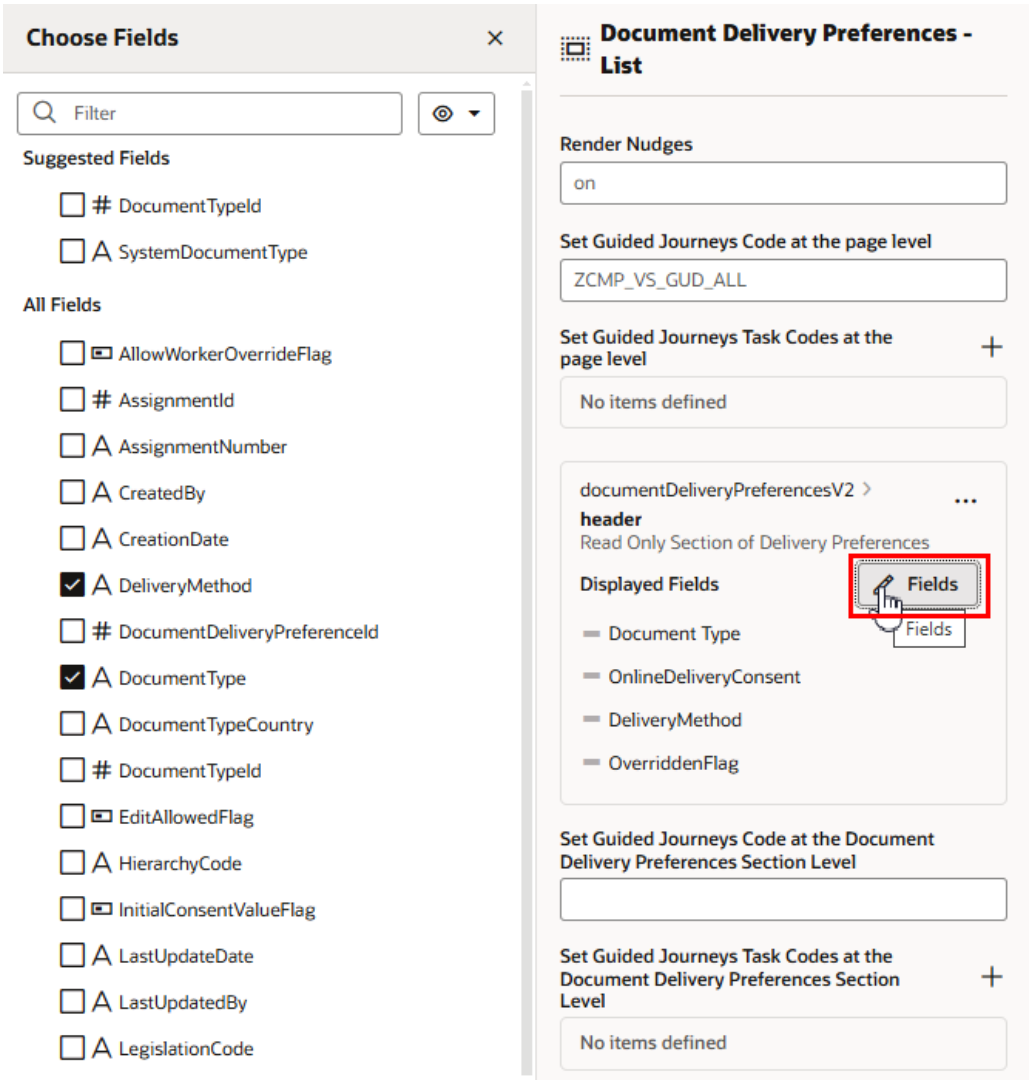


However, if the details pane looks like this, you need to use the rule set editor to configure the fields:



In this case, you click **Rules** to open the rule set editor. For more, see Configure Rules in the Rule Set Editor.

3. Edit the fields displayed in the component.
   - To change the field order, grab the handle next to the field and moving it into position.
   - To delete a field, click the Trash icon next to the field name.
   - To add and remove fields, click **Fields**, and then select and deselect fields in the Choose Fields popup.



The changes you make in the Properties pane are always displayed in the component because the rule controlling the fields has no conditions. If you want to use rules with conditions, or you want to group fields, you need to create the rules in the rule set editor. For more, see Configure Rules in the Rule Set Editor.

Configure Rules in the Rule Set Editor

If the rules in your component's rule set use conditions, or you want to create rules that use conditions, you need to use the rule set editor.

The rule set editor lists rules defined in the extension dependency under Built-In Rules. You cannot edit the rules under Built-in Rules, but you can copy them to use as the basis for your own rules. The rules that you create and can edit are listed under Extension Rules.

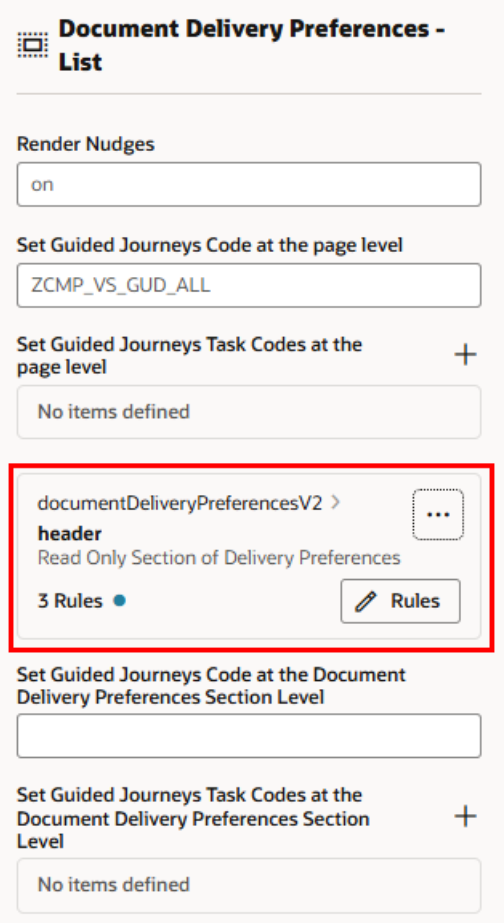
Like business rules, the order in which rules appear in the Extension Rules list is important. However, unlike business rule, rules in rule sets are evaluated at runtime *from top to bottom*. The first rule where all the conditions are met determines the layout used. None of the rules below it in the list are tested. Keep this in mind as you're working on the rules.

Let's look at how to create a rule to hide a field when the user is not authenticated.

1. Open the page you want to modify in the Designer.
2. Open the rule set editor.

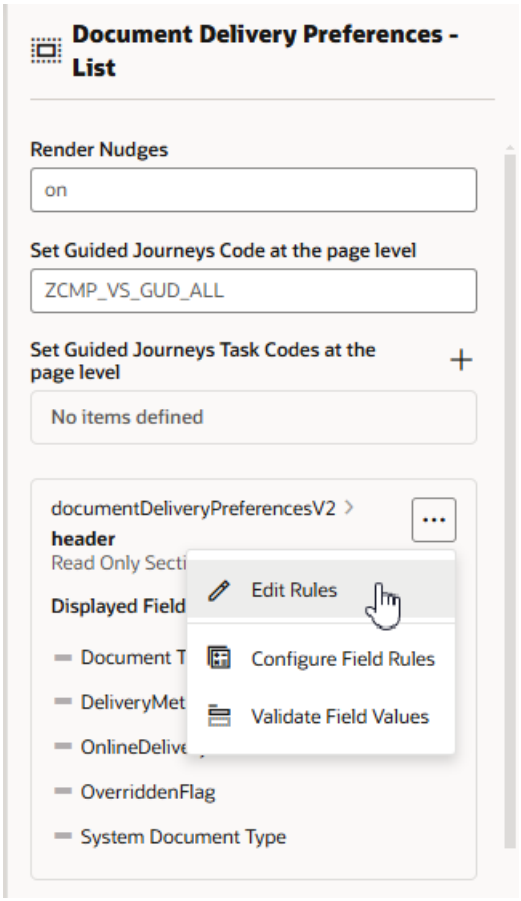
   Locate the component you want to edit in the Properties pane, or select the component on the page to open the component's details pane in the Properties pane.

   If the component's rule set has any rules with conditions, you might see something like this in the Properties pane. Click **Rules** to open the rule set editor:



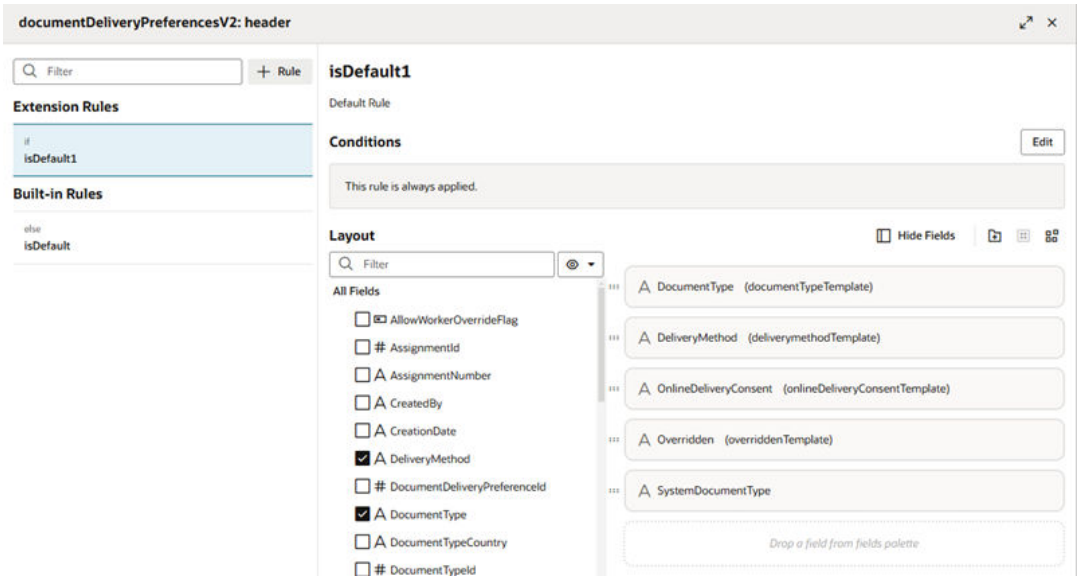
   In the example above, the rule set has three rules, including the default rule.

   If you see something like this in the Properties pane, select **Edit Rules** in the options menu to open the rule set editor:



3. Create a new rule in the rule set editor.

   Let's look at the rule set for a form layout as it appears in the editor:



   The left pane of the editor lists the rule set's rules. A rule set always has one built-in default rule (`isDefault`) that is always applied. (In this example, the list also contains a rule `isDefault1` that is created automatically if you edit the layout in the Properties pane.)

To the right, the selected rule's conditions are displayed above the list of available fields and the fields displayed with the rule:

To create a rule:

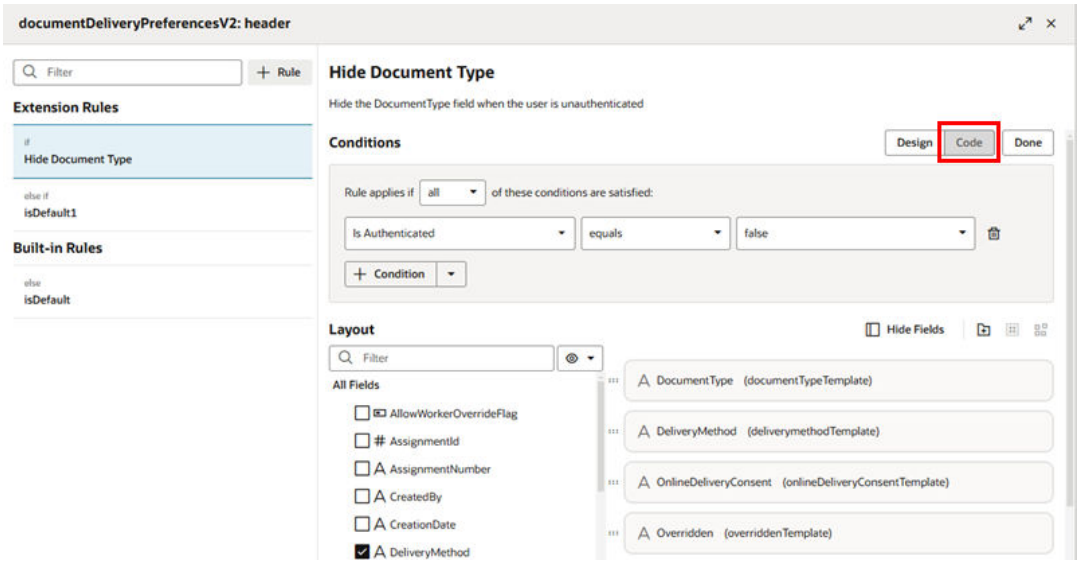
- Click + Rule in the left pane to create a new extension rule, and then provide a name and description in the popup dialog.
- Right-click a rule in the list of rules, and then select **Duplicate** in the popup options menu. Enter a name and description for the new rule in the dialog box.

4. After you create the rule, position it in the Extension Rules list by grabbing its handle and dragging it into the position you want.

   Rules are evaluated from the *top down*, so make sure the rules are in the order you want them evaluated.

5. Select the rule in the Extension Rules list, then click **Edit** to open the condition builder and define its conditions.

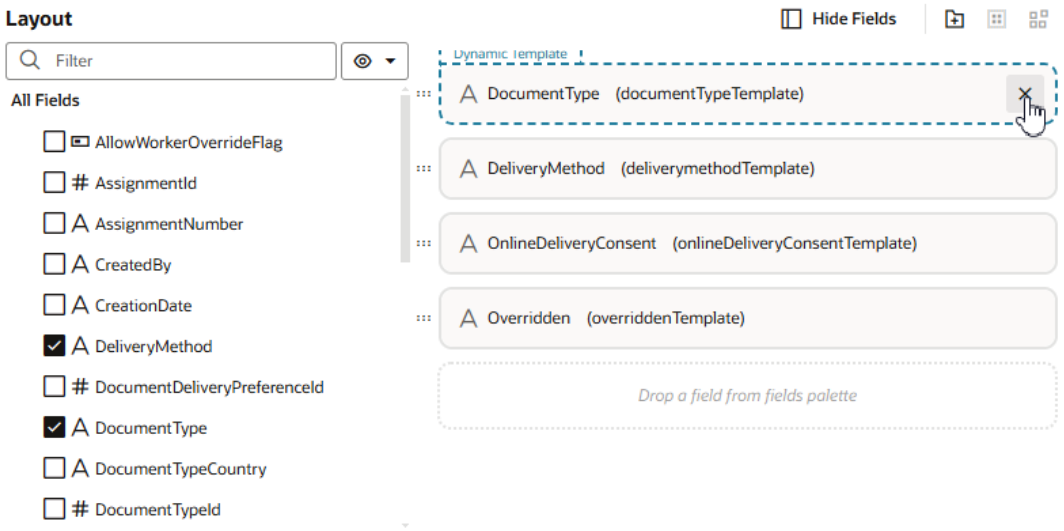
   The condition builder is the same one used to create conditions for extension rules in business rules. You can click **Code** to write the condition in the code editor. If you're not familiar with using the condition builder, see Set Conditions for a Rule.



Click **Done** to close the condition builder. Your changes are automatically saved.

6. In the Layout pane, select the fields you want displayed in the component when the rule is active.

   Add a field or object to a layout by selecting the checkbox next to the field, or by dragging it from the list into the layout. To delete a field, you can deselect its checkbox in the Fields palette, or click X in the layout:



The layout editor contains a Fields palette listing all the fields that can be displayed in the layout. If the palette is hidden, click **Show Fields**. You can click **Hide Fields** in the toolbar to hide the Fields palette.

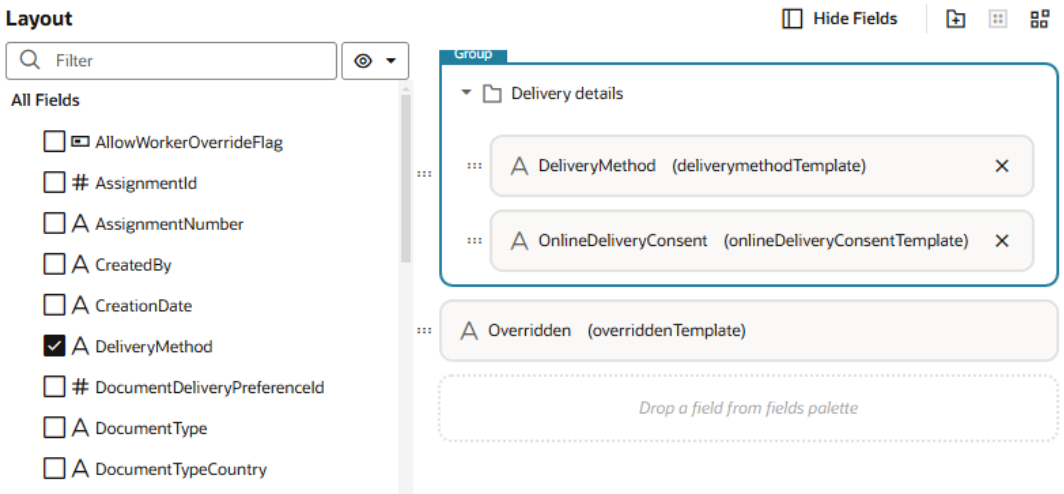
The fields you can display in a rule's layout are determined by the fields available from the data resource used by the component. This data resource is defined in the extension dependency. You can choose any of the fields listed in the Fields palette—and the order in which they should appear—but you can't include fields from other data resources.

To help you locate the fields you might want to add, the Fields palette might contain a Suggested Fields section at the top of the palette. This section lists the fields that have been identified as the most relevant or most important when building your layout.

You can also filter the list of fields by entering a string in the Filter field at the top of the Fields palette.

7. (Optional) Group fields by selecting all the fields that you want to include in the group, either by holding down the CMD key (on macOS) or the Ctrl key (on Windows), and then clicking in the toolbar.

The selected fields are grouped under a folder in the list. You can type a name for the new folder in the layout editor:



You use groups to divide a dynamic form into different sections. Each subsection has a heading based on the group name.

To ungroup the fields in a group, select the group and then click in the toolbar.

8. Organize the order that fields and groups are displayed in the component by dragging its handle and dragging it into position in the layout.

> **Note:**
>
> When the rule's conditions are met, you can see the changes you make to the page in the Designer. (You need a browser window wide enough so that the page is visible on the left, next to the editor.)

9. Click **X** at the top of editor to close it.

6

Preview, Share, and Publish Your Changes

After you've made your changes in the Designer, you have a few options for testing your work before you actually push it to your production instance.

Preview and Share Your Changes

Here are some techniques for testing your page before publishing it:

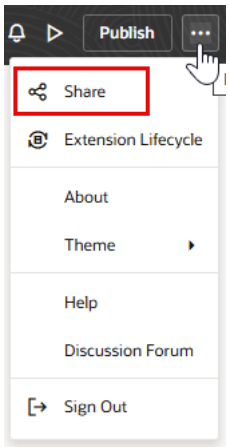
- To preview your page, click **Preview** in the Designer's header:

  This opens your page in another browser tab. You may be asked to re-enter your credentials for the Oracle Cloud Applications instance before the tab opens.

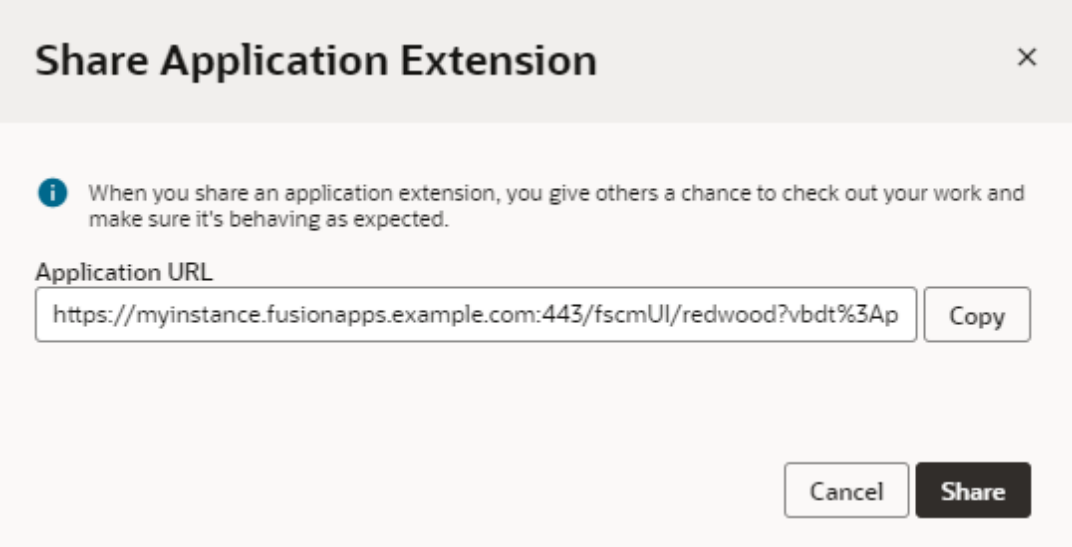
  As the name implies, Preview gives you a chance to fully test your app before sharing it with others. In Preview, the pages in your app will look and behave as they will when published. You can test the navigation between pages in the app, as well as links to external pages.

  To quickly check how a page behaves, you can use Live view in the Designer. However, Live view is not a substitute for Preview, and some behavior in Live view, such as how pages are rendered, or using links to navigate from the current page, might not work as expected.

- To allow others to test your work, click **Share** from the Menu in the Designer's header:



  You should now see the Share Application Extension dialog, which will have a URL with your changes that you can give to your team members:



Click **Copy**, then share the URL with the team members who want to test your changes via an email or direct message (DM). Make sure you click **Share** *before* you click **Copy**, as otherwise the URL will not be valid.

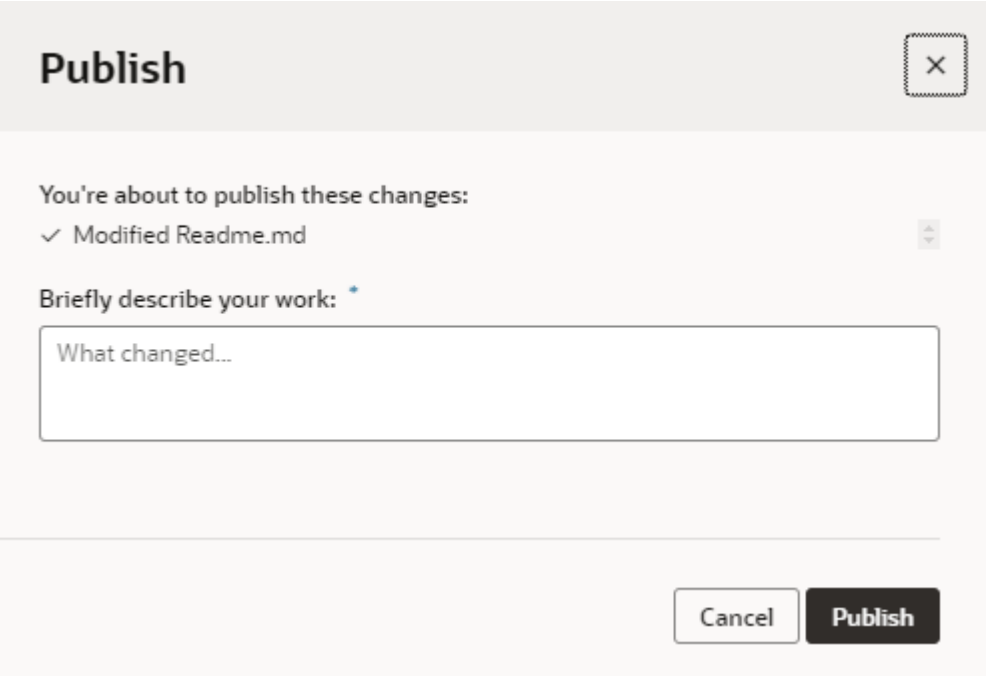
Publish Your Changes

When you and your team members have finished testing your work, it's time to kick off the publishing process so your changes can be applied to your Oracle Cloud Applications instance within your development environment.

To publish your work (which in VB Studio is referred to as an "extension"):

1. Click **Publish** in the Designer's header:

2. If you have changes that haven't been saved to the underlying repository, enter a brief comment describing your changes when prompted, then click **Publish**:

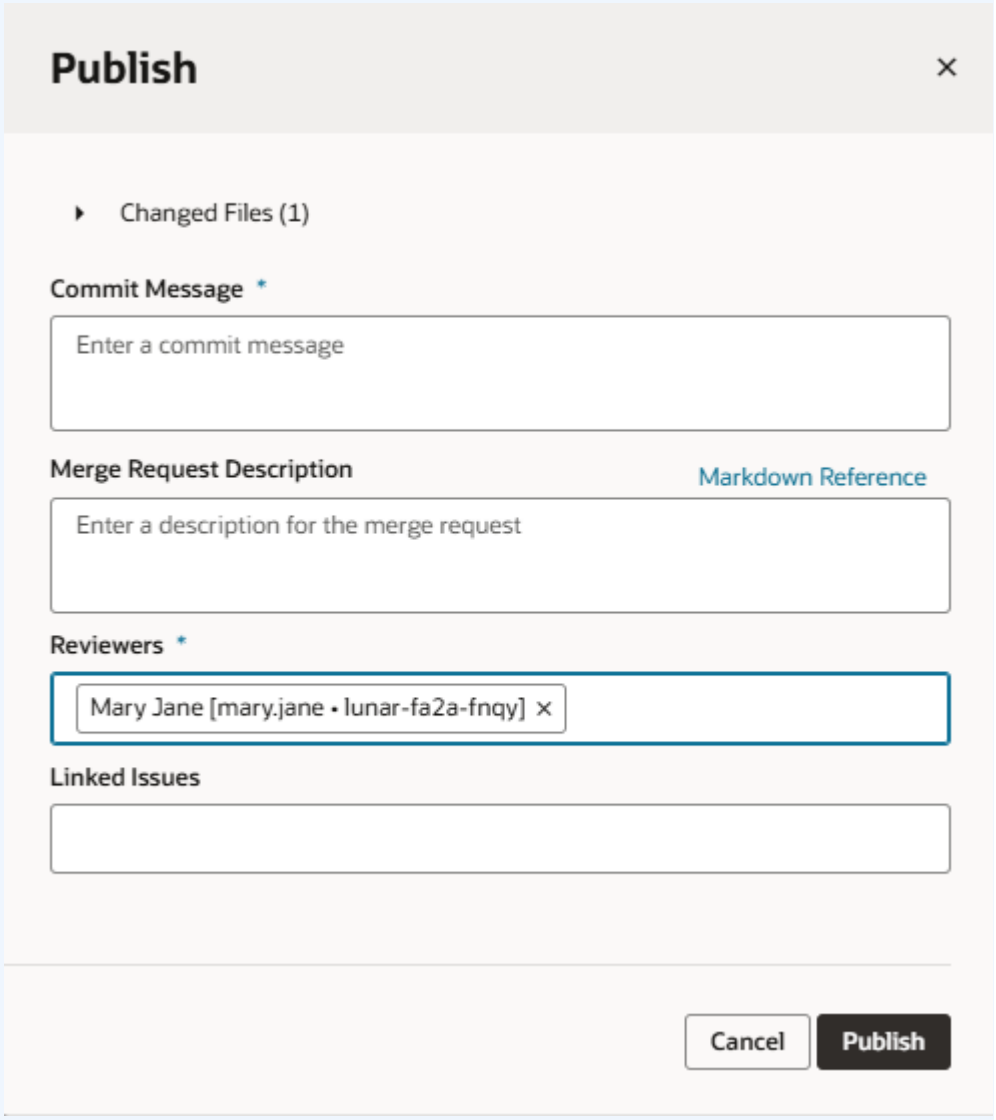


Tip:

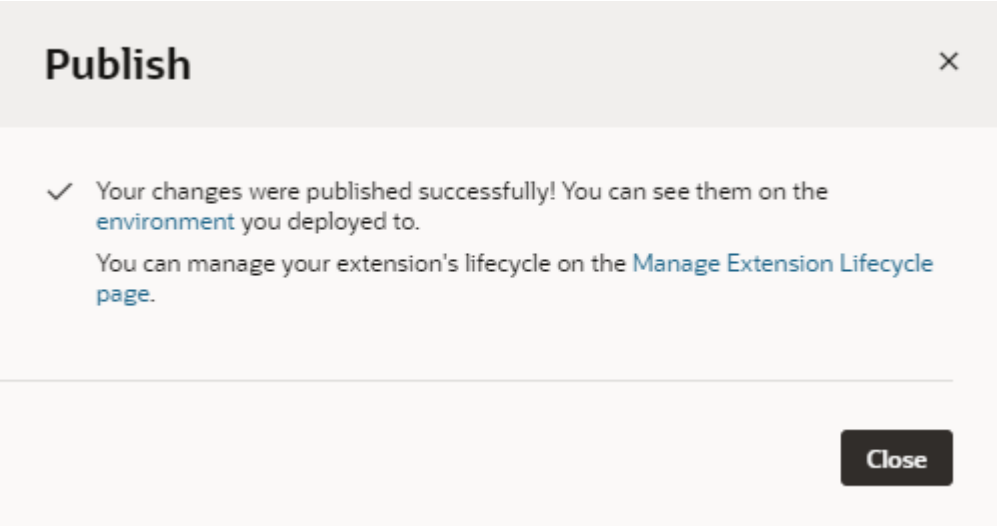
It's considered best practice to include a link to your changes in your description. This makes it easy for others to review your changes, either during the Publish process or later, if it's ever necessary to debug something that may have gone wrong. You can generate a URL to your changes by clicking **Share** from the Menu in the Designer's header (represented by three horizontal dots).

Note:

Your view of the Publish dialog may be different based on some advanced settings. If your administrator has set things up so that all changes are first reviewed and approved by a developer (who will actually look at the code representing your changes), you'll need to enter a commit message to briefly describe your work. You can also enter (or update) a merge request description describing all the changes you've made, using the template (Markdown, Textile, or Confluence) set up for your project. (If you don't know what these are, you can skip this field.) Then, add at least one reviewer if a name isn't already specified, add the linked issues if you know them, and click **Publish**:



At this point, you'll be notified that your changes were successfully published. Click the **environment** link to see your deployed changes. (Make sure you open the link or copy its location to your clipboard before you click **Close**. You won't have access to it after the Publish dialog is closed.)



(If you'd like to publish your changes to an Oracle Cloud Applications instance other than the one you're working in, click the link to the **Manage Extension Lifecycle page**. See Manage Your Published Extensions.)

Note:

If you don't see a link to your environment, it's likely that your extension uses continuous integration and delivery (CI/CD) pipelines for deploying your changes. In this case, you'll see notifications in the bottom right corner of the Designer indicating that the publish process has started. (If you miss these notifications or want to track progress, look for them under **Notifications** in the header.)

If prompted however, you must complete a few extra steps that allow your changes to be deployed to the Oracle Cloud Applications instance you're working with:

- If you're prompted for missing credentials, simply enter your user name and password for your Oracle Cloud Applications instance and click **Add Credentials and Continue**.
- If you're prompted for authorization, you'll need to ask your admin to go to the Deploy job in the CI/CD pipeline used to publish your changes and provide login credentials for your Oracle Cloud Applications instance. Once that is done, click **Publish** to try and publish your changes again.

When publishing is complete (indicated by the Deploy build successfully completing), you can access the Oracle Cloud Applications instance to see your changes.

Publishing via CI/CD pipelines gives you the flexibility for advanced lifecycle operations, such as adding extra steps to the out-of-the-box configuration or using shell scripts against a target environment. If this isn't required, switch to the Advanced mode and disable the CI/CD pipeline.

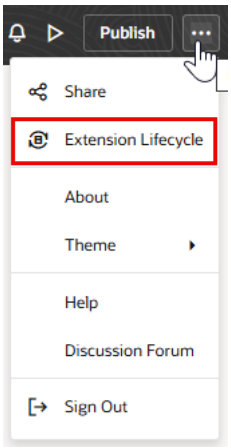
IMPORTANT: To see the changes you just made, your users will have to sign out of the Oracle Cloud Applications instance, then sign back in again to be certain they're seeing the latest.

Manage Your Published Extensions

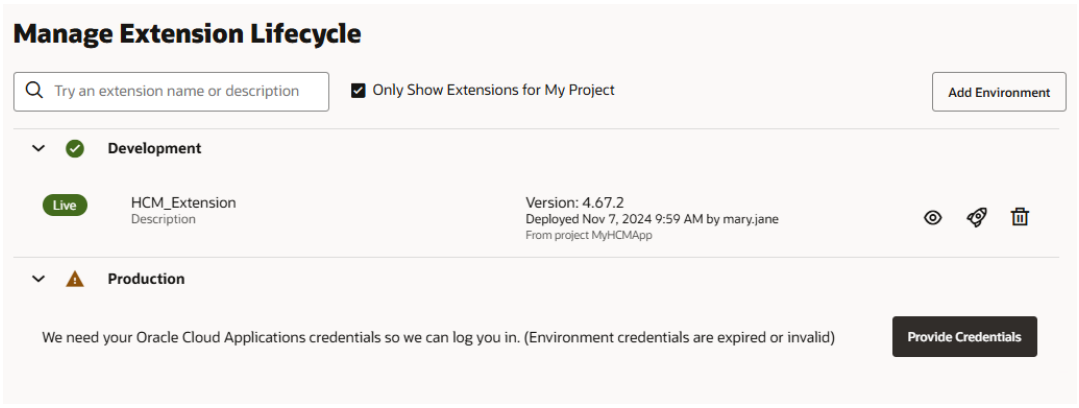
By default, the **Publish** button deploys your extension *only* to the Oracle Cloud Applications instance you're currently working in, which is typically an instance used for testing. To publish to a different instance, say to your production instance, use the Manage Extension Lifecycle page. From this page, you can deploy an extension to other instances and also delete them when no longer needed.

Before you can deploy an extension to another instance from this page, make sure the extension is already published to your default Oracle Cloud Applications instance. You must also have a valid user name and password for the target Oracle Cloud Applications instance you want to deploy to.

1. Click **Extension Lifecycle** from the menu in the Designer's header:



You should now see the Manage Extension Lifecycle page, which lists all the environments that are available for you to deploy to.



In VB Studio, an *environment* defines an Oracle Cloud Applications instance you want to work with. By default, VB Studio creates an environment for you called "Development", which is where the **Publish** button deploys your extensions.

> **Note:**
>
> If you see a **Provide Credentials** button next to an environment, you must first sign in before you can deploy your changes to it.

Expand each environment to view a list of the extensions already deployed. Each entry includes the extension name plus additional details, such as extension version, deployment date, who deployed the extension, and project name.

Use the **Only Show Extensions for My Project** checkbox to toggle between viewing extensions deployed only from the current project, or from all projects.

2. To publish an extension to another Oracle Cloud Applications instance:

   a. Find the extension that you want to deploy and click  .

   b. Choose the environment that points to the instance you want to deploy to.

   > **Note:**
   >
   > Your project administrator should have already created any additional environments you might need to deploy to. If you don't see an expected Oracle Cloud Applications instance or if that instance is unavailable, check with your project administrator or add a new environment right from this page (see step 3).

   c. Click **Deploy**.

   IMPORTANT: To see the newly published extension, your users will have to sign out of their Oracle Cloud Applications instance, then sign back in again to be certain they're seeing the latest.

3. If you'd like to deploy to an Oracle Cloud Applications instance but it's not listed on this page, you can add it.

   a. Click **Add Environment**.

   b. In the Add Environment panel, enter basic environment details, such as the environment's base URL, environment name, and user name and password for the target Oracle Cloud Applications instance.

   c. Click **Add**.

4. To delete an extension from an Oracle Cloud Applications instance, click  .

   Deleting an extension restores your application to its original state.

Resolve Conflicts

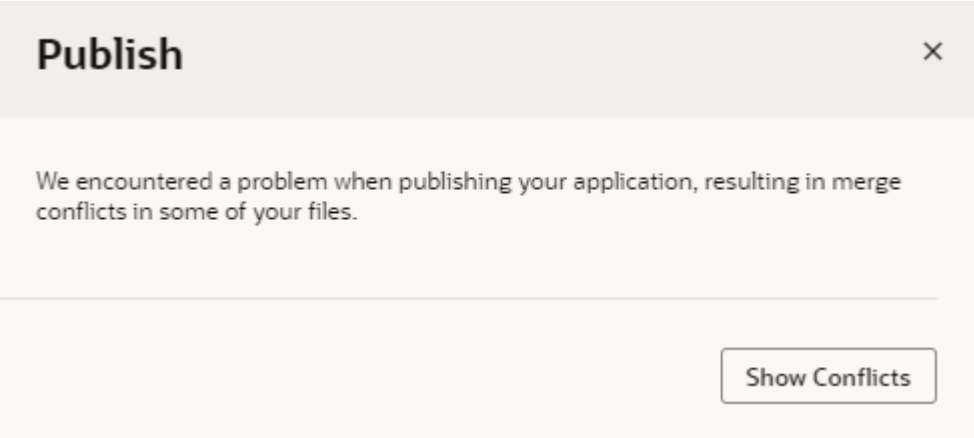
Sometimes when you publish changes, you might run into conflicts. Conflicts usually occur when you and your teammate make changes that overlap or conflict with each other. For example, both of you might have changed the same line in a file in different ways, or one of you might have deleted a file while the other modified it. In such cases, VB Studio cannot tell

which change should take precedence—it's up to you or your teammate to make that decision and resolve the conflict.

Before you begin resolving conflicts, familiarize yourself with some concepts underlying source control in VB Studio. At the heart of your work in VB Studio is *Git*, which stores source files in a *repository* and manages all your changes through *branches*.

When you first update pages, you're starting with a set of files as they exist in a Git repository's default branch (`main`), which is the *remote repository* containing the source from which your pages are built. As you begin to make changes, your work is saved to a local branch, in a *local repository*—but these changes are not visible to others. To make them available to others, you *publish* your changes, which through a series of Git operations, such as *commit*, *fetch*, and *merge*, pushes your changes from your local branch to the `main` branch in the remote repository.

Now let's say you've changed lines 2 and 3 in `Readme.md` but are yet to publish those changes. If someone else modified the same or subset of these lines in `Readme.md` and published those updates, then when you publish your changes, you'll be warned of conflicts between the file's remote version and your local version:



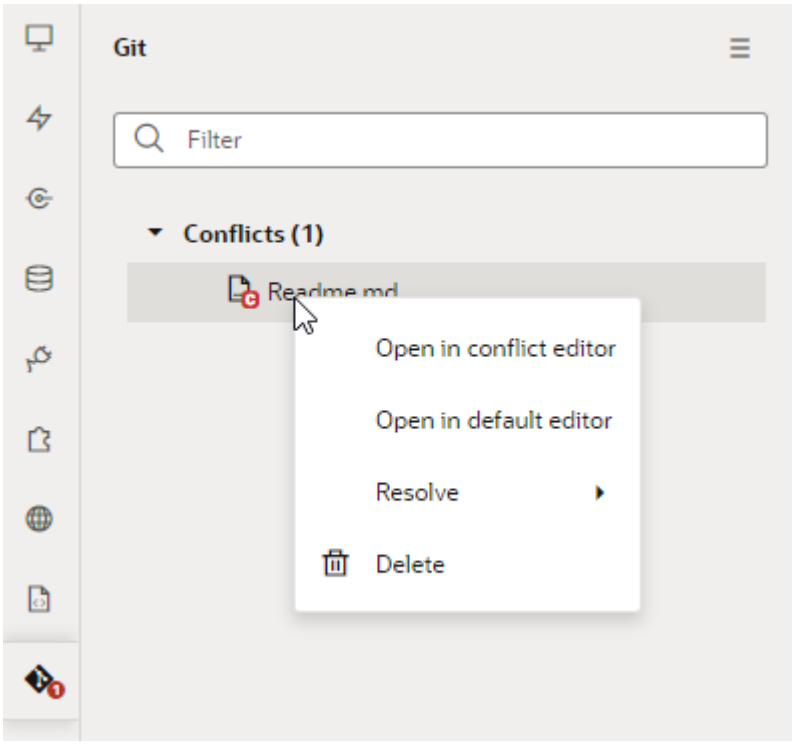
Here's what to do when this happens:

1. Click **Show Conflicts** to view the files identified as containing conflicts.

   Note that at this point, VB Studio switches to *Advanced* mode, which provides extensive Git options that you can explore further if you want, but it is not required. The actions described in this section are likely sufficient to help you resolve the conflicts and proceed with publishing.

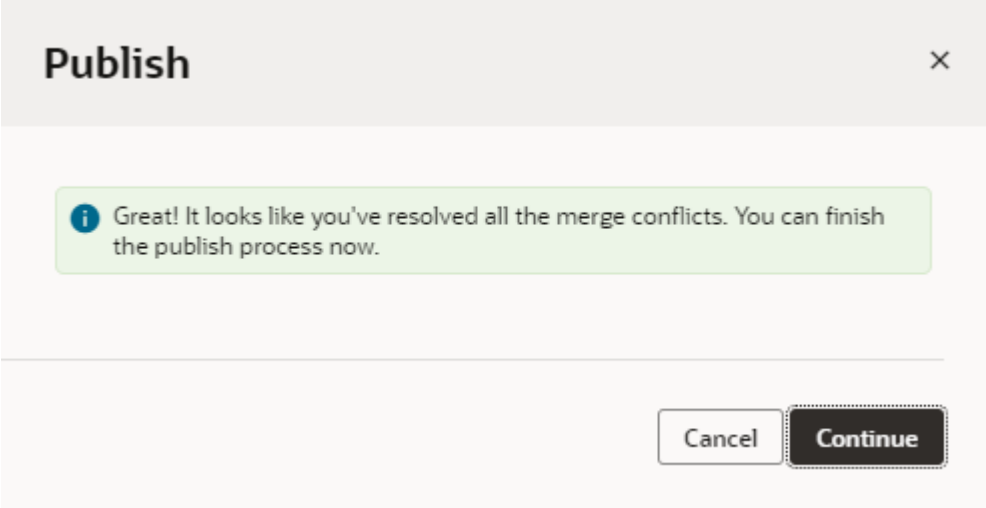
2. Right-click the file listed under Conflicts in the Git Panel and take action.

   The options shown depend on the type of conflict in the file. Here's an example of what you see when people make different changes to the same line of the same file (`Readme.md`):



Action	Steps
Open in conflict editor	Select this option to open the file in the conflict editor; you can also just click the file to open it in the conflict editor. The conflict editor has controls to help you navigate between conflicts and provides options to resolve them. See Use the Conflict Editor to Resolve Conflicts.
Open in default editor	Select this option to open the file in the default editor, which is the designated file editor in VB Studio (for example, the Page Designer for an .HTML file or the JavaScript editor for a .JS file). This option is useful for non-text files (such as Excel worksheets and schema files) or other artifacts that aren't supported in the conflict editor. You can then open these files in the default editor and manually resolve the conflicts.
Resolve	Select this option to quickly resolve the file's conflicts, instead of going through each conflict in the conflict editor. This option is useful when the file has only a few conflicts that you can easily resolve by selecting either your version or the other version. See Use the Context Menu to Resolve Conflicts. **Note:** Resolve is not an option for non-text files. If you run into conflicts for binary files (say, an image), you'll need to delete the file from the Git repo, then add it again *after* you've resolved all other conflicts and committed them to the remote repo.
Delete	Select this option to delete the file from the Git repo.

3. If you have multiple files with conflicts, right-click each file in the Git Panel and take action.
4. Once you've resolved all the conflicts, click **Publish** in the header. If the conflicts are all resolved, click **Continue** to complete publishing your changes.

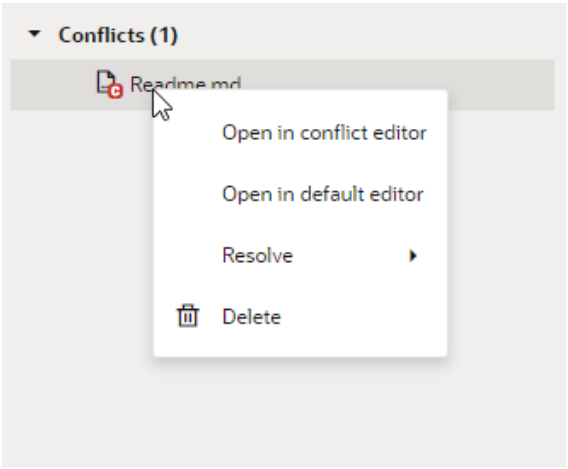


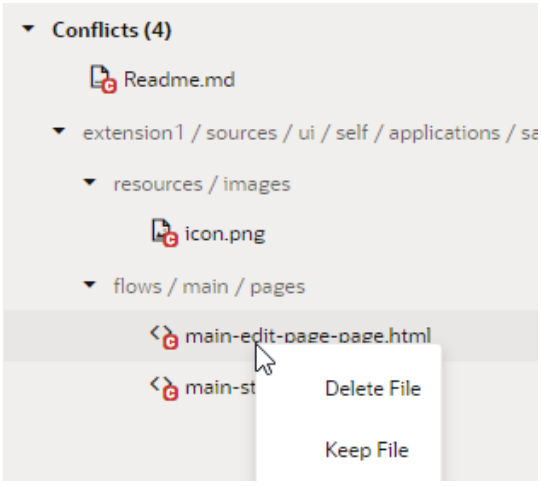
Use the Context Menu to Resolve Conflicts

When a file contains conflicts, you can use its context menu in the Git Panel to quickly resolve conflicts. You do this when the file has just a few simple conflicts that you can quickly resolve by keeping either your changes or somebody else's.

> **Note:**
>
> If you want to keep changes from both branches (or a combination of them), or if the file has a large number of conflicts, it's better to open the file in the conflict editor and resolve each conflict separately.

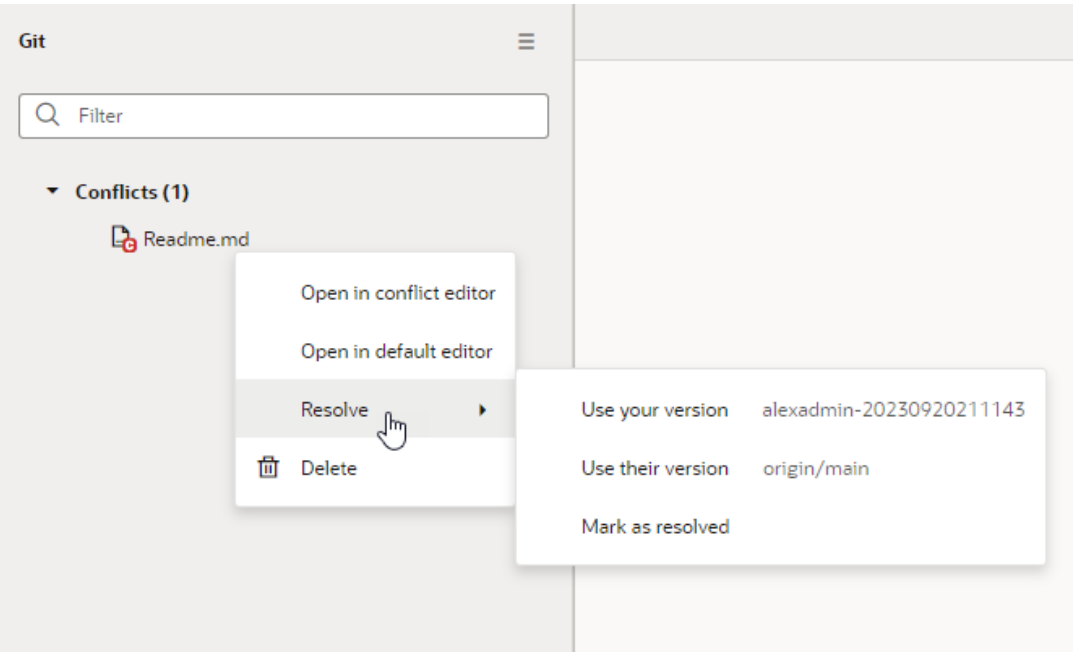
The options available to you in the context menu depend on the type of conflict in the file. For example, what you see when a file is deleted in one branch and modified in another (as shown here on the right) is not the same as what you see for a file modified by different people on different branches (shown on the left):





For demo purposes, we'll use the example of a file that's been modified by you and someone else.

1. Right-click the file with conflicts in the Git Panel and select **Resolve**.



2. Select the option you want to use:
   - Select **Use your version** to keep your changes in the local branch, also listed for identification purposes.
   - Select **Use their version** to keep changes in the remote `main` branch.
   - If needed, select **Mark as resolved** to mark the file as resolved. When you select a file version (**Use your version** or **Use your version**), the file is automatically marked as resolved. Select **Mark as resolved** only if you manually made changes to a file either in the default editor or the conflict editor and did not mark the file as resolved (using **Resolve and Close**) in the editor.

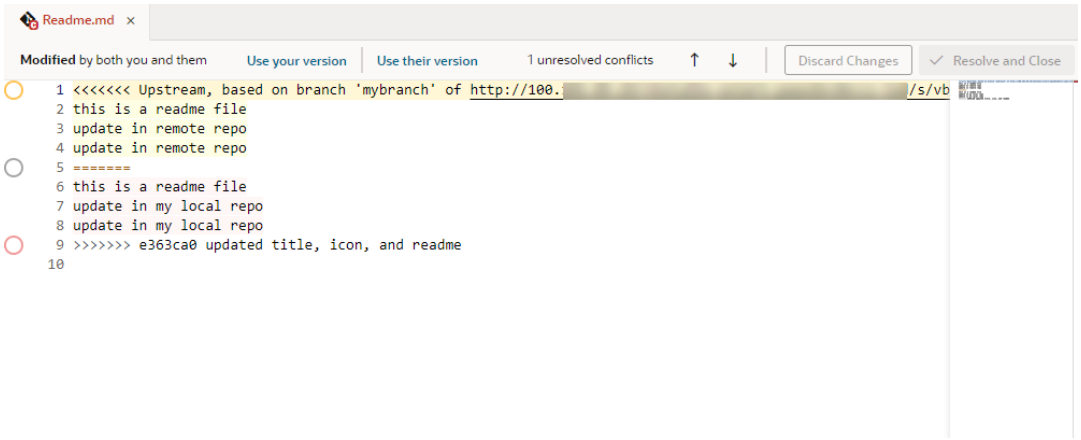
Use the Conflict Editor to Resolve Conflicts

When you open a file with conflicts in the conflict editor, you can use the available controls to navigate between conflicts and resolve them.

1. Right-click the file with conflicts in the Git Panel, then select **Open in conflict editor**. You can also double-click the file, especially if the type of conflict doesn't give the option to open the file in the conflict editor.

   When the file opens in the conflict editor, conflicts between your local version of the file and the remote version are highlighted. For example, here's an example `readme.md` file (with

a single unresolved conflict) that was modified by you and someone else:



2. Decide how you want to resolve the conflict. You can use the options in the toolbar or the markers that appear next to a conflict in the editor.

   The options available to you depend on the type of conflict you are resolving. For example, if you deleted a file in your local branch and someone modified the same file in the remote branch, you might see **Delete File (Your Version)** and **Keep File (Their Version)**. (It's worthwhile to note that the content of deleted files shows in read-only mode and cannot be edited until the conflict is resolved. Note also that non-text file content won't show in the conflict editor.)

   For demo purposes, we'll use the example of a file that's been modified by you and someone else. Here are the options available to you for this use case:

Action	Step
To keep changes in the remote branch	• Select **Use their version** in the toolbar, or • Click in the canvas and select **Their Version**.
To keep your changes in the local branch	• Select **Use your version** in the toolbar, or • Click in the canvas and select **Your Version**.
To keep changes from both branches	Click in the canvas and select **Use Both Changes**.

> **Note:**
>
> You can also ignore these options and update the file as you would any text file, to keep the lines you want and delete the rest. You do this when you want to incorporate changes from both branches, for example, to keep some of your changes and some of theirs.
> The lines between `<<<<<<<` and `>>>>>>>` represent a conflicting change, with changes from each ref separated by `=======`. Remember to delete these markers as well as Git-related comments when resolving a conflict.

3. If the file has more than one conflict, click ↓ in the toolbar to go to the next conflict and take steps to resolve it.

4. Optional: Click **Discard Changes** and confirm when prompted to revert *all* the updates you've made to resolve conflicts in the file.

   Once you discard your changes, resolve the conflicts again to proceed.

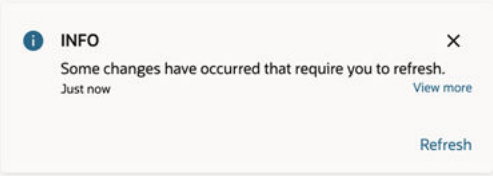
5. When you see the **No unresolved conflicts** message in the toolbar, click **Resolve and Close** to mark the file as resolved.

Refresh Your Source Files

To avoid code conflicts in the first place, it's important to keep your source files up-to-date with everybody's changes. Simply refresh your workspace, when notified, to bring in changes that your teammates have published.

> **Note:**
>
> For optimal performance, this functionality takes effect only in workspaces with fewer than 1000 files.

1. When others have published changes that apply to your work, you'll see a notification that your workspace needs to be refreshed:



> **Tip:**
>
> The notification to refresh your workspace is an informational message that clears after five seconds. If you missed the notification or want to take action later, look for the message under **Notifications** in the header. It's helpful to periodically review unread notifications, so you don't miss important information. VB Studio checks for changes in your repository every 10 minutes.

2. Click **Refresh** in the notification.

   Under the covers, VB Studio does either a Git *merge* or *pull* to refresh your workspace with the latest updates. Click **Close** in the Refresh Workspace dialog.
   If you run into conflicts, click **Show Conflicts** to go to the Git Panel in Advanced mode and take steps to resolve the conflict. See Resolve Conflicts.

Roll Back a Published Extension

Sometimes after publishing your changes to an Oracle Cloud Applications instance, you might need to roll them back. When you "roll back" an extension, what you're actually doing is redeploying a previous extension version. To do this, you'll need to locate the previous version

in VB Studio, and then move it to your extension's remote repository so that you can redeploy it.

It's always best to redeploy the previous version's actual build artifact that was originally packaged and deployed. If you have access to this build artifact in Advanced mode and you'd like to redeploy it, see this blog.

But, if build artifacts from previous extension versions aren't available, here's what you can do instead:

1. Make sure you're working in VB Studio's Advanced mode.
2. In the Designer, switch your branch to the previous extension version.

   > **Tip:**
   >
   > Look for the branches named <user name>-<timestamp>; these are the branches that are created each time you publish an extension (if you didn't rename them).

   When switching, select the **Create new branch based on parent branch** checkbox.
3. Push the new branch to your remote repository.
4. From the VB Studio left navigator, click **Builds**, then configure your extension's packaging job with a new "Branch" parameter.

   Set the parameter's default value to `main` so that, when you use the **Publish** button in VB Studio, everything still functions as usual.
5. Run the packaging job manually, entering your newly created branch (instead of `main`) in the parameter you just added.

   The packaging job will run and create a build artifact using the specified branch. Once the job finishes, it will automatically trigger the deploy job as usual.

For step-by-step instructions, see this blog.

7 Troubleshooting

These topics cover some common issues and how to address them.

Respond to the Housekeeping Popup

If you see a popup that says "We need to do a bit of housekeeping!", you'll need to hop over to Advanced mode to take care of it.

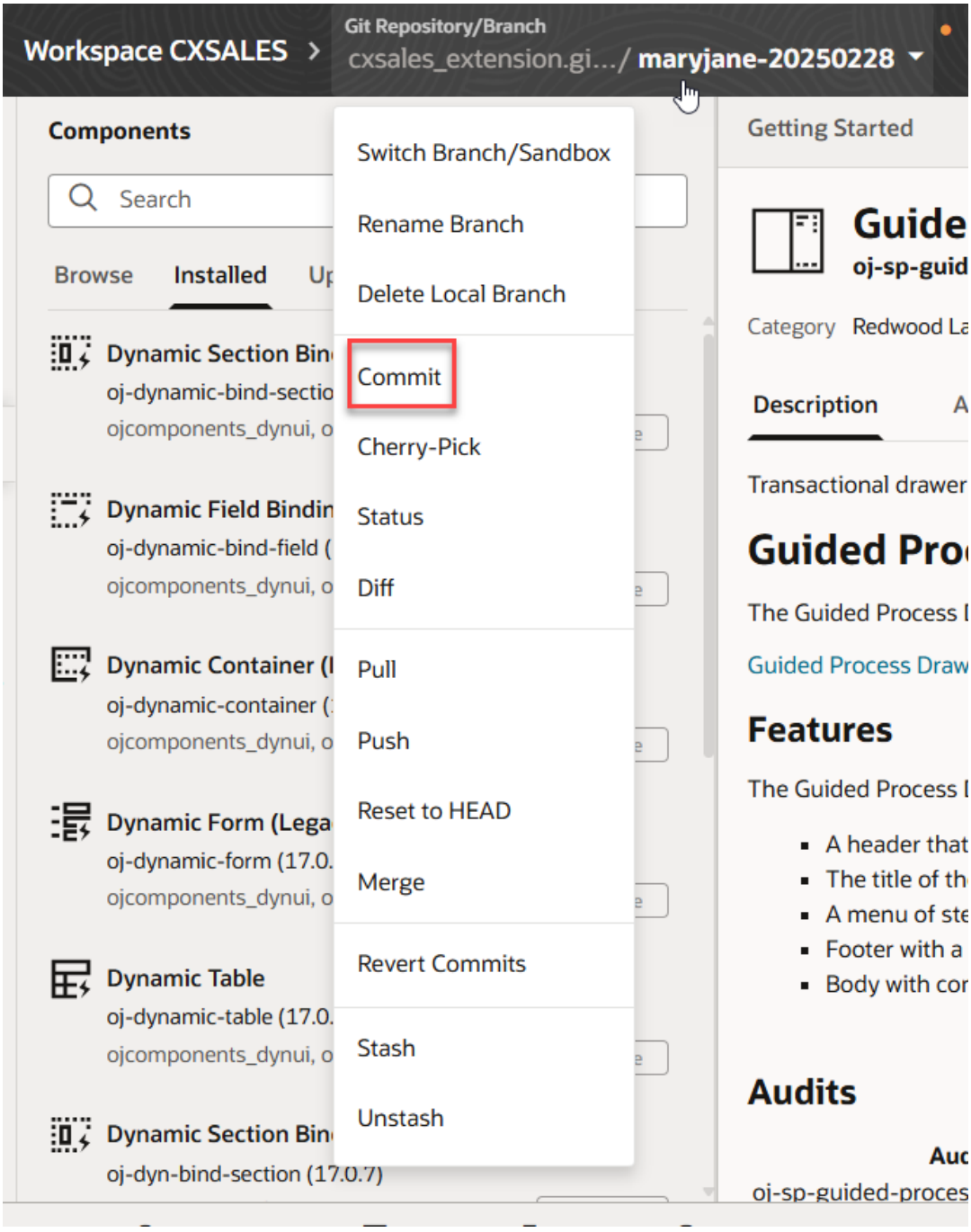
1. In the popup, click **Open Settings**.

   You'll be immediately switched to Advanced mode. At this point, there's a few different situations that may require your attention:

   - If you see the Application Migration dialog, that means there's a new version of VB Studio's underlying libraries ready for you to uptake. Click **Upgrade**.
   - Separately, you might also see (under Runtime Dependency) a message that there are new migrators available; click **Migrate** to get the latest.

   If either the **Upgrade** or **Migrate** buttons are grayed out because you have some outstanding changes, proceed to the next step. Otherwise, you're done here, and can click **Express** in the header to get back to work.

2. To commit any pending changes you may have, hover over this area in the header, then click **Commit**:



This should allow you to upgrade or migrate, as needed. When you're done, click **Express** in the header to return to your Oracle Cloud Application.