

Oracle® Cloud

Extending Oracle Cloud Applications in Visual Builder Studio Express Mode



Release 24.07

F94278-02

May 2024

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Cloud Extending Oracle Cloud Applications in Visual Builder Studio Express Mode, Release 24.07

F94278-02

Copyright © 2023, 2024, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	v
Documentation Accessibility	v
Diversity and Inclusion	v
Related Resources	v
Conventions	vi

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

What Is the Designer?	1-1
Select the Designer Theme	1-4
Access Visual Builder Studio	1-5
Other Views of the Designer	1-7

2 Control Your Display with Business Rules

What Are Business Rules?	2-1
Work with Business Rules	2-2
Filter Your Rules	2-2
Create an Extension Rule	2-5
Set Conditions for an Extension Rule	2-7
Create Condition Using a Field's Initial Value	2-10
Build Advanced Expressions	2-12
Use Nested Rules	2-17
Set Properties For Regions and Fields	2-19
Identify the Regions and Fields Impacted By a Rule	2-21
Set a Default Value for a Field	2-21
Set Properties at the Region Level	2-23
See Out of the Box Behavior	2-24
What Do the Blue Indicators Mean?	2-24
Locks, Blanks, and Dashes	2-25
Understand What Will Be Shown at Runtime	2-25
Interpret the Pop-Up Viewer	2-26
Display Messages When Conditions Are Met	2-28

3 Work With Page Properties

4 Work With Containers and Sections

Control the Sections Displayed on the Page	4-1
Configure the Content Displayed in a Section	4-3

5 Control Your Display With Rule Sets

Create an Extension Rule in a Rule Set	5-2
--	-----

6 Preview, Share, and Publish Your Changes

Preview and Share Your Changes	6-1
Publish Your Changes	6-2
Publish Your Extension to Other Instances	6-6
Resolve Conflicts	6-11
Use the Context Menu to Resolve Conflicts	6-14
Use the Conflict Editor to Resolve Conflicts	6-15

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](#)
- [Documentation Accessibility](#)
- [Related Resources](#)
- [Conventions](#)

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.
<i>italic</i>	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*.



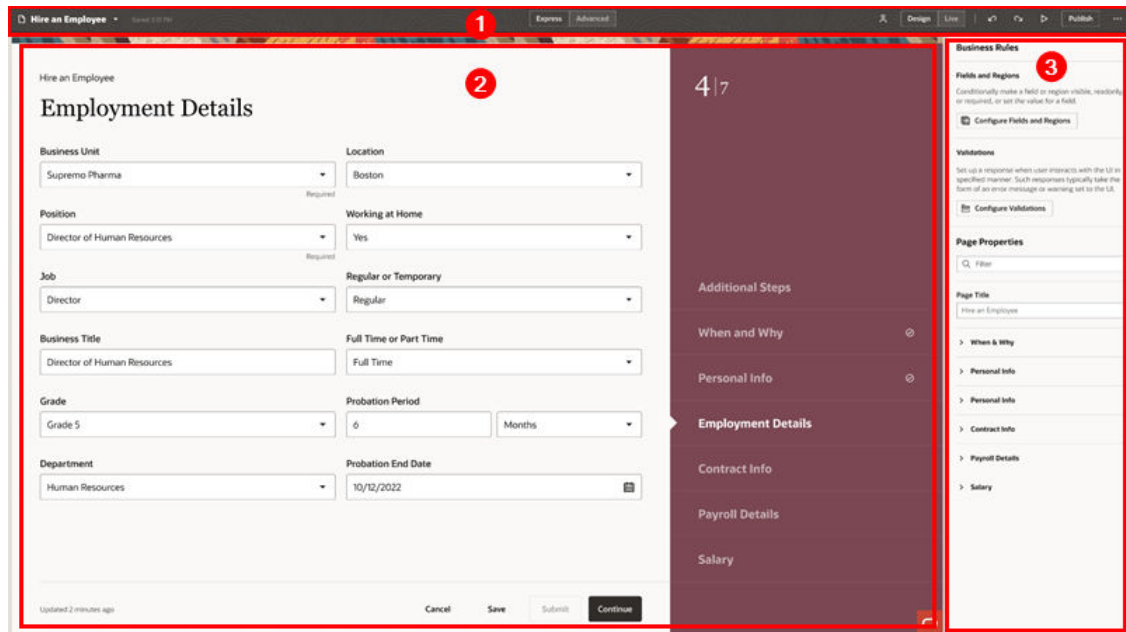
Note:

If you're an HCM user, you may also want to refer to [Overview of Redwood Application Extension](#) 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 is divided into three main areas:

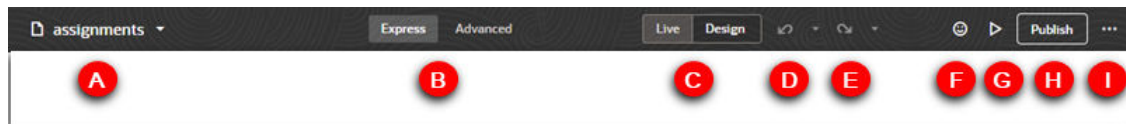


1. Header
2. Canvas/Editors
3. Properties pane

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

The Header

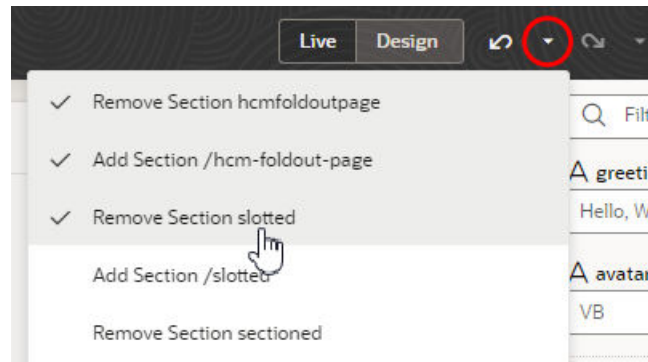
The header contains information about your current VB Studio session, and provides access to tools that help you share and publish your changes when you're ready.



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	Live / Design	Toggles between Live and Design modes. <ul style="list-style-type: none"> • Live mode: Displays the page as it will look and behave when it is published. Use Live mode to interact with the page and to confirm that your application is behaving as you expect. • Design mode: The mode you use most frequently, to select components on the page.

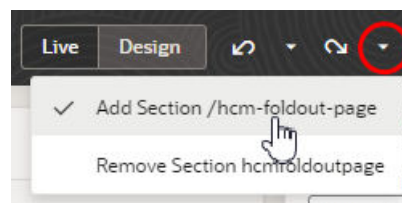
Label	Element	Description
D	Undo	Undo one or more of your changes. To undo your most recent change, click the Undo icon (hover your cursor over the icon to view the action that will be undone). To undo multiple changes, click the Undo drop-down list and select the actions you want to undo. For example, 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.

E	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 be 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.

Label	Element	Description
F	Feedback	Submit your feedback about VB Studio to Oracle.
G	Preview	Use the Preview action to see how your page looks and behaves in a browser.
H	Publish	After you're done testing your changes, use the Publish action to apply them to your development environment.
I	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.

VB Studio constantly saves your changes as you work, so there's no need to save them explicitly.

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 Visual Builder 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.

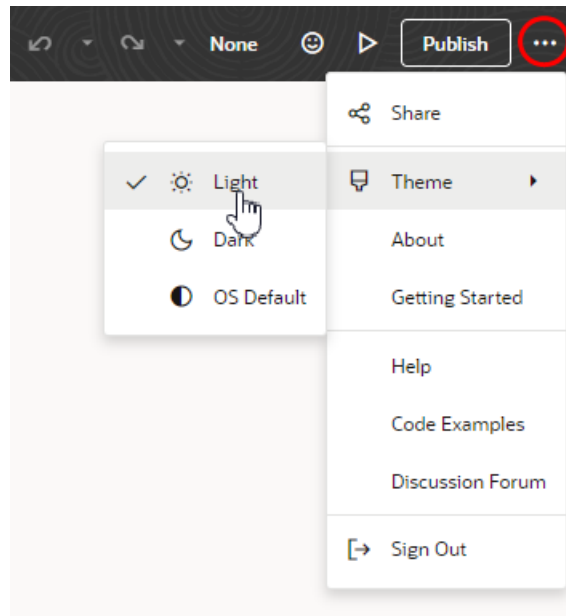
If an Oracle Cloud Application page has configurable containers or page properties, you can edit them from the Properties pane.

Select the Designer Theme

By default, the Designer uses a light theme to set the color palette for your work environment. You can personalize this theme to switch to a dark theme, or you can sync the theme with your OS settings.

To set the Designer's theme:

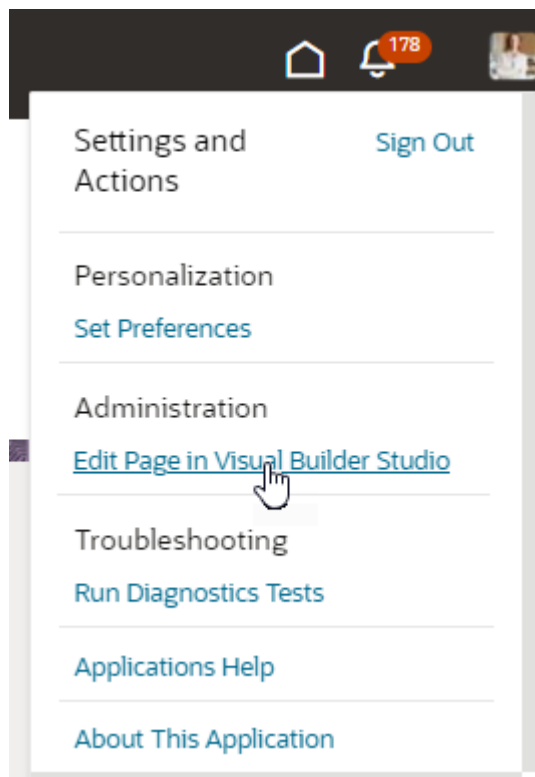
1. Click the Menu option in the upper-right corner.
2. Select **Theme**, then choose an option:



- 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**:



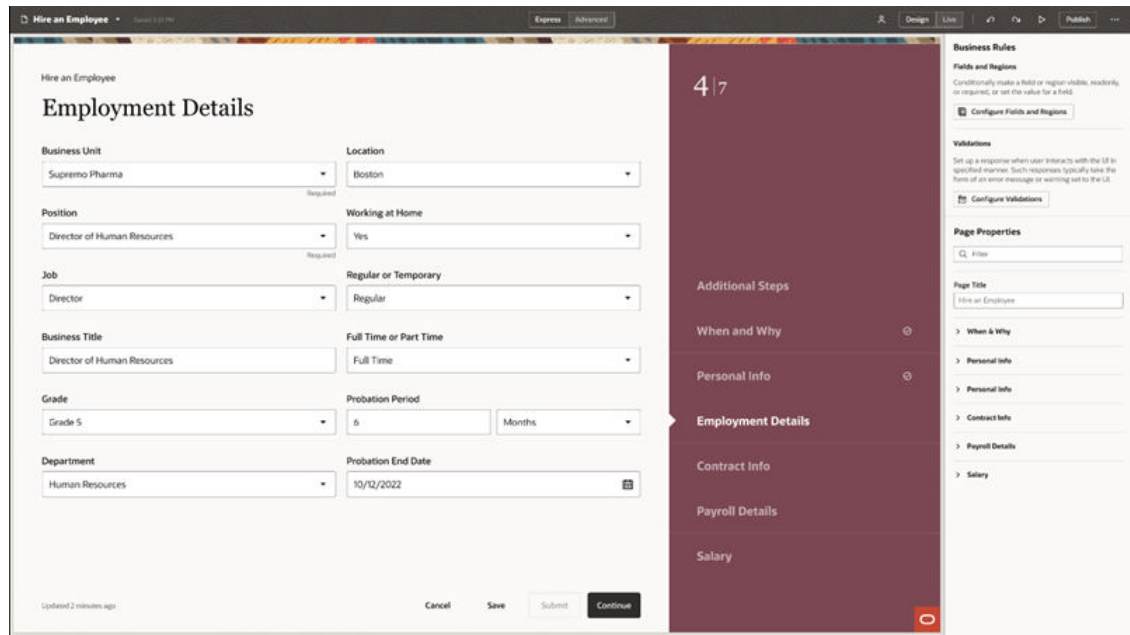
You will then land in the Designer, with all the tools you need at your disposal.

 **Note:**

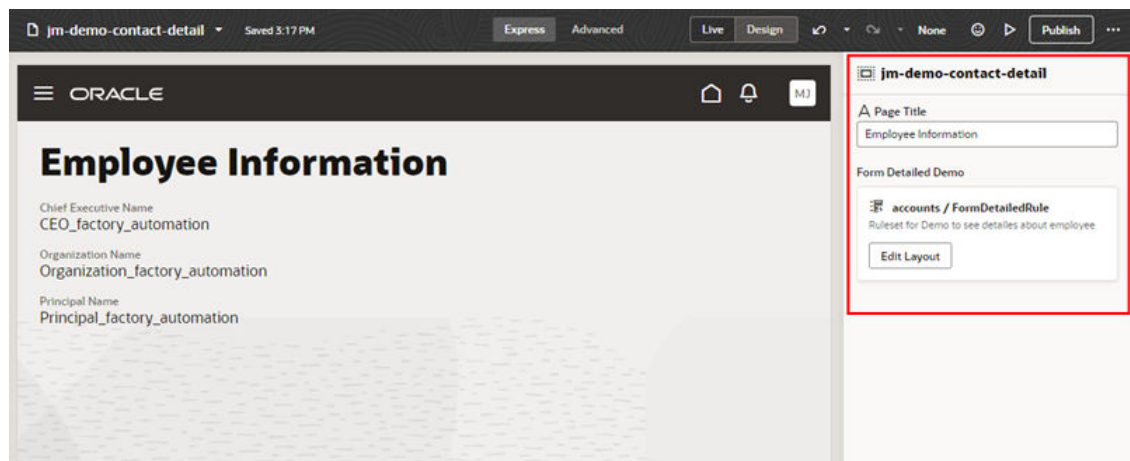
If you don't see the **Edit Page in Visual Builder Studio** option in your Oracle Cloud Application, it could be because:

- 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.

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.

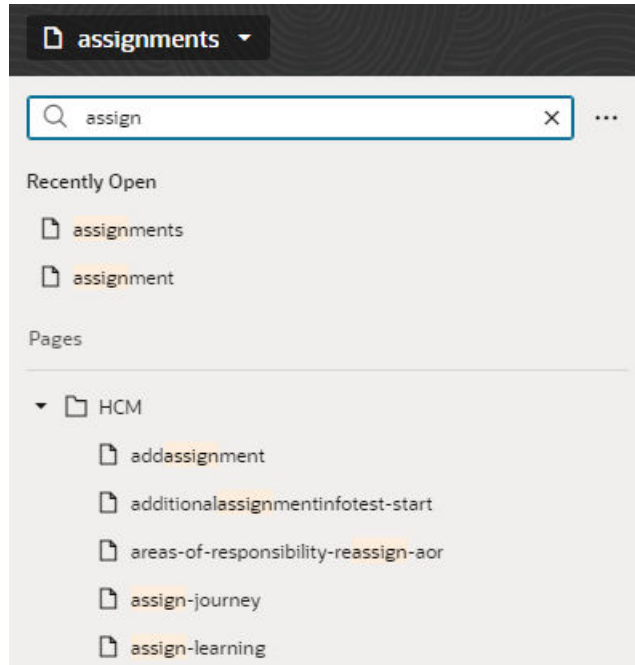
For example, you may see a notice in the Properties pane that says your page is not extensible, which means that neither the business rules editors nor the rule sets editor are appropriate for this page. If you see this message, it's best to return to the browser tab with

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. 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:



We can't display this page.

There's nothing wrong with the page, but there are some background issues we can't resolve at the moment.

[Tell me more](#)

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.

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 on a page are displayed at runtime, based on conditions such as the user's role and the employee's business unit or legal employer.

Business rules allow you to easily control what's displayed on the page as a whole, instead of configuring page elements 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 or regions on a page, provided that the specified conditions in a rule are met at runtime. In this context, a *region* is simply a dynamic form.



Note:

If you're looking for information on business rules as they pertain to business objects, see [Create Rules for Business Objects](#).

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 two dynamic forms (Detailed department and NewForm). These are listed in the Regions and Fields section in the image below. Both components use fields from the same data object, and you can quickly see every field used in each component. For example, the Detailed department region consolidates all the fields that are displayed in the form, regardless of the layouts defined in the form. The same applies to the NewForm region. 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 component using that data object.

The screenshot displays the Business Rules configuration interface. On the left, there is a search bar and tabs for 'Fields', 'Regions', and 'Rules'. Under 'Extension Rules', three rules are listed: 'Hide Business Unit' (Inactive), 'Buyer in Canada', and 'Buyer in Argentina'. Under 'Built-in Rules', the 'Default' rule is selected. The main panel shows the configuration for the 'Default' rule. It has no conditions. The 'Regions and Fields' section is expanded, showing a table with columns for 'Required' and 'Hidden'. Two regions are listed: 'Detailed department' and 'NewForm'. Each region has three fields: 'Chief Executive Name', 'Organization Name', and 'Principal Name'. All fields are set to 'Optional' and 'Visible'.

There are two types of rules: *extension rules*, which are created by you, the extension developer; and *built-in rules*, which are created by Oracle as part of your extension dependencies. In this example, the Default rule is a built-in rule defined in an extension dependency.

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.

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:

+ Rule

Fields
Regions
Rules

▼ Extension Rules ? ...

Country is Germany

Country is Brazil

Head Count Approval

▼ Built-in Rules ?

▶ Project Manager in Australia

Country Is Japan

Country is Canada

Job Role is Regional Manager

Job Role is Line Manager

Default

Head Count Approval

Head Count is greater than 100

Conditions

Head Count (Initial Value) is greater than 100

Regions and Fields

▼ When & Why Section

- A Action Reason
- ▣ customAAField
- A customWWField
- A Degree Conferred
- ✳ Hire Date
- # yearsOfExperience

▼ BudgetDetailsSection ○ (2)

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

×

A **Budget Amount**
Field in BudgetDetailsSection

▣ **BudgetDetailsSection**
Region

A **Budgeted Position**
Field in BudgetDetailsSection

Q **budget**

▶ Project Manager in Australia

Country Is Japan

Country is Canada

Head Count Approval

Head Count is greater than 100

Conditions

Head Count (Initial Value) is greater than 100

Regions and Fields

▼ When & Why Section

- A Action Reason
- ▣ customAAField
- A customWWField

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 (not shown), or field. Select the magnifying glass icon to list all rules where the word occurs. In the image above, when you select the Budget Amount field, the list of rules is filtered to only include those that override a property of the Budget Amount field:

field:"Budget Amount" X

Matches 4 rules of 20

▼ **Extension Rules** ? ...

Head Count Approval
Sets **Budget Amount** to Required

▼ **Built-in Rules** ?

Job Role is Line Manager
Sets **Budget Amount** to Optional

Default
Sets **Budget Amount** to Required and Visible and Editable

This makes it easy to tell at a glance where the Budget Amount 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 **Regions**, and then enter a term like "when" in the Filter field, the filtered list shows the rules that override a property of the region When & Why Section:

region:when X

Matches 2 rules of 20

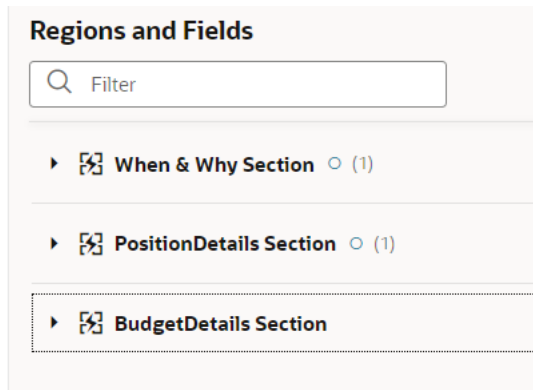
▼ **Extension Rules** ? ...

Country is Brazil
Sets region **When & Why Section** to Hidden

▼ **Built-in Rules** ?

Country is Canada
Sets region **When & Why Section** to Hidden

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 "cost" so you enter that in the Filter field. The BudgetDetailsSection region is displayed, but the other two disappear. It's not until you expand BudgetDetailsSection that you see the field **CostCenter**:

Country is Germany

Hide Degree Conferred when Manager is in Germany

Conditions

Job Role equals "Line Manager" and
Country equals "Germany"

Regions and Fields

Q cost X

▼ BudgetDetailsSection

A CostCenter



Note:

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

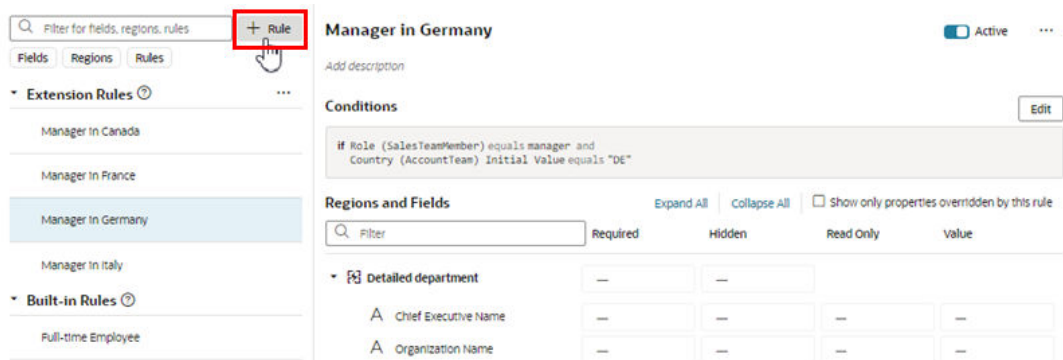
Create an Extension Rule

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

To create an extension rule:

1. Open the page you want to configure.
2. In the Properties pane, click **Configure Fields and Regions** in the Business Rules pane.

3. Click **+ Rule**.



Rather than starting from scratch, you can duplicate an extension rule and use it as the basis for your own extension rule. Duplicating a rule also duplicates any JavaScript files used in the rule to define expressions. To do this, right-click the rule, click **Duplicate**, then proceed to the next step.

Note:

You cannot duplicate rules defined in extension dependencies (the rules listed under Built-in Rules), which includes the Default rule.

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.

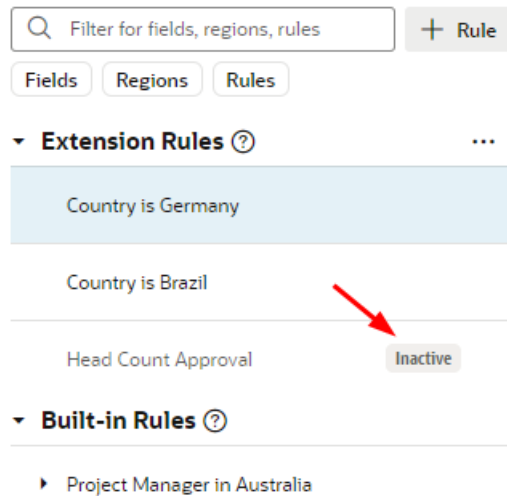
5. Click **Create**

Your new rule is listed under **Extension Rules** at the top of the list of rules.

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, using the grab handles (six dots) beside the rules to drag-and-drop them to new positions. You can 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 a little badge appears next to it, like this:



Filter for fields, regions, rules + Rule

Fields Regions Rules

▼ Extension Rules ? ...

Country is Germany

Country is Brazil

Head Count Approval Inactive

▼ Built-in Rules ?

▶ Project Manager in Australia

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 Extension Rules heading, 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.

Set Conditions for an Extension 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 the Canada and has the Manager role.

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.

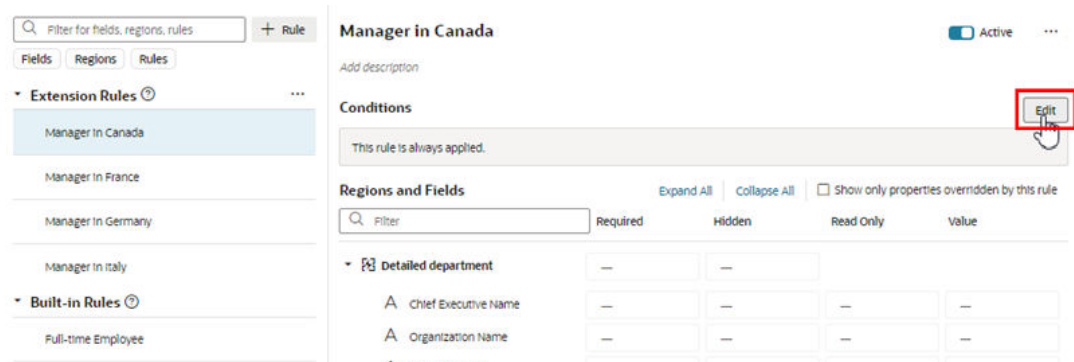
 **Note:**

When using `$user.role` in a condition, the Value drop-down lists the available Oracle Applications Cloud job and abstract roles. (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.)

You must be granted the `ASE_REST_SERVICE_ACCESS_IDENTITY_INTEGRATION_PRIV` privilege to see the user roles in the drop-down list. Contact your instance administrator if you require this user privilege.

To create conditions for a rule:

1. Select the extension rule, 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 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:

Rule applies if **all** of these conditions are satisfied:

Job Role	contains		
Country	contains		
aut	contains		
<div style="border: 1px solid #ccc; padding: 2px;"> <div style="display: flex; align-items: center;"> - { } User <div style="margin-left: 10px;"> <input checked="" type="checkbox"/> Is Authenticated </div> </div> </div>			
+ Condition		Use Advanced Expression	

b. Select an operator.

The options available in the operator dropdown menu are pre-defined based upon the criterion's type. So for a criterion like User Authenticated, the operator menu only has equals, and the only available values in the value menu are yes and no:

Rule applies if **all** of these conditions are satisfied:

Job Role	contains		
Country (Component Context)	contains		
Is Authenticated	equals	Yes	
BusinessUnit	contains		
+ Condition		Use Advanced Expression	

c. Specify the values.

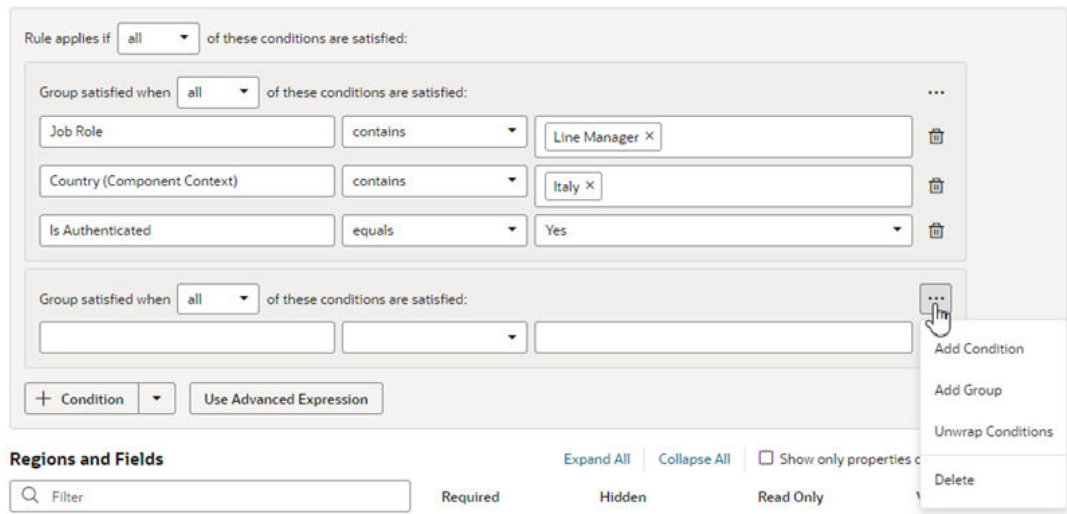
Select a value from the dropdown list, or type a value in the field. Depending on the criterion, you might be able to choose multiple values.

- 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.

- 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 under Extension Rules, then click **Edit** above the Conditions pane.

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

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**.

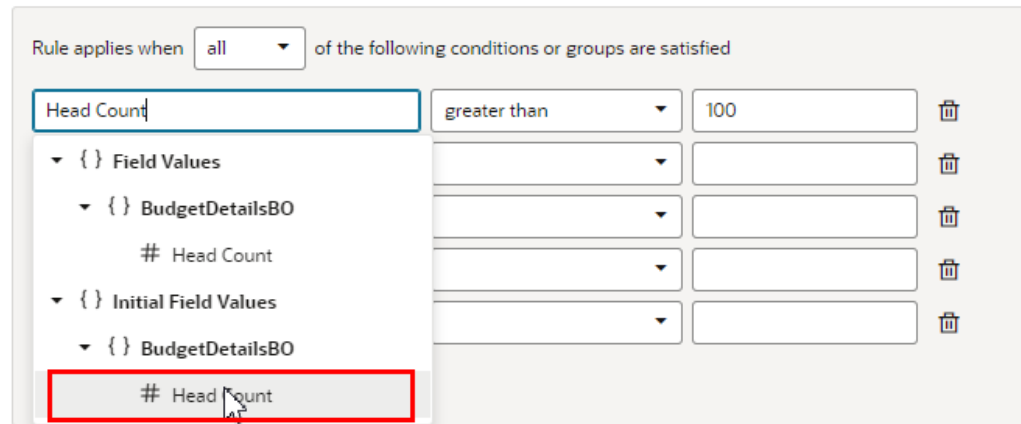
When creating a condition, you might want to use the value retrieved from the data source when the page loads—that is, the Initial Field Value. Once the page loads, the Initial Field

Value doesn't change. The Field Value, on the other hand, is the value displayed or cached in a page, which may already have been modified by a rule or user. For example, say the value for the Head Count field retrieved from the data source (the Initial Field Value) is 50. 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 unaffected by changes made by rules or users, so it is still 50.

Let's look at how to add a rule so that users cannot edit the Head Count value in a form if the field's initial value is greater than 100.

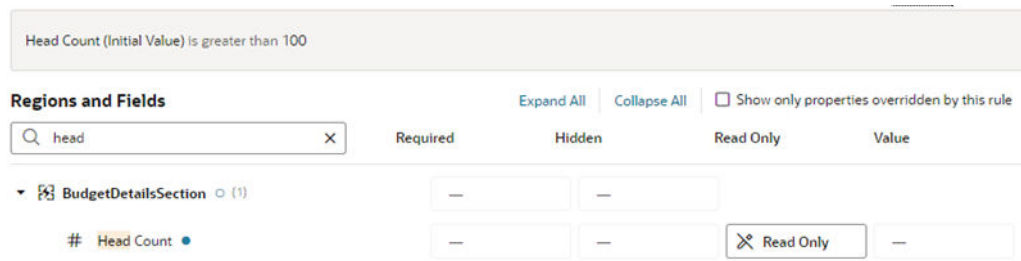
1. Open the page in the Designer.
2. Create an extension rule to set the field's Read-Only property:
 - a. In the Properties pane, click **Configure Fields and Regions** in the Business Rules pane, and then create a new extension rule.
 - b. Create a condition where the initial value of Head Count is greater than 100.

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



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

To help you locate where the Head Count field is used, you can type 'head' in the filter field to filter the list of fields:



In this example, the Head Count field in the form might display a value greater than 100 and *still* be editable, because a rule is modifying the Head Count value. If you wish, you could create a validation rule that displays a message when the Head Count is over 100, telling the user the Head Count field value must be 100 or less. In the condition for the validation rule,

make sure the rule is evaluating the Field Value (not the Initial Field Value) for the Head Count field.

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:

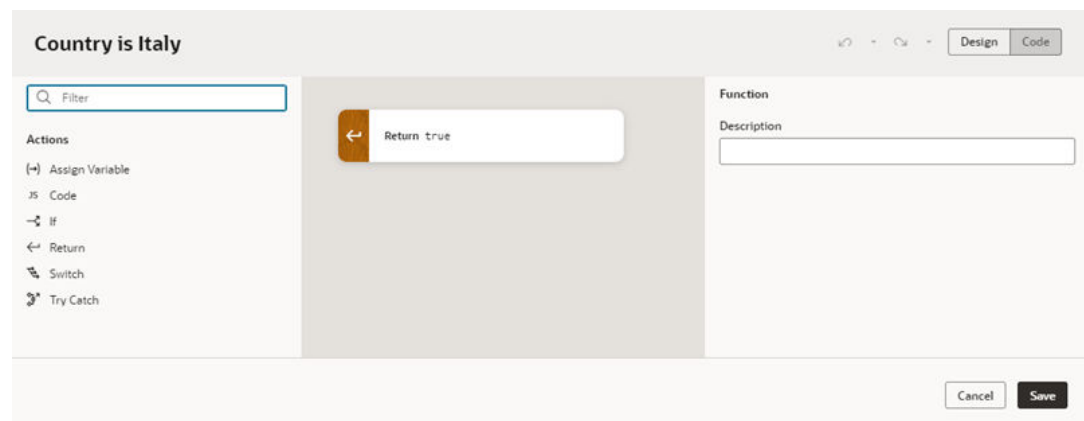
If you use advanced expressions in the condition builder, you will no longer be able to use the basic condition builder. If you have already defined some conditions in the basic condition builder, they will be displayed as actions in the Advanced Expression editor.

To create an advanced expression:

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

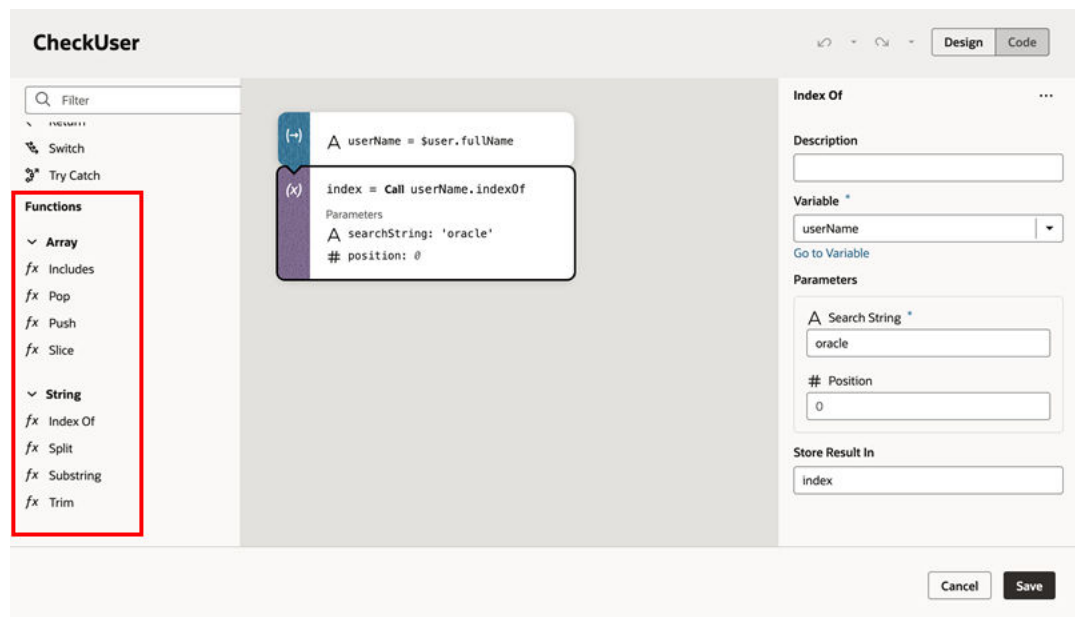
The Advanced Expression editor has a Design mode, which you use to visually create an expression, and a Code mode where you can type the expression. You toggle between modes using the **Design** and **Code** buttons in the header.

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

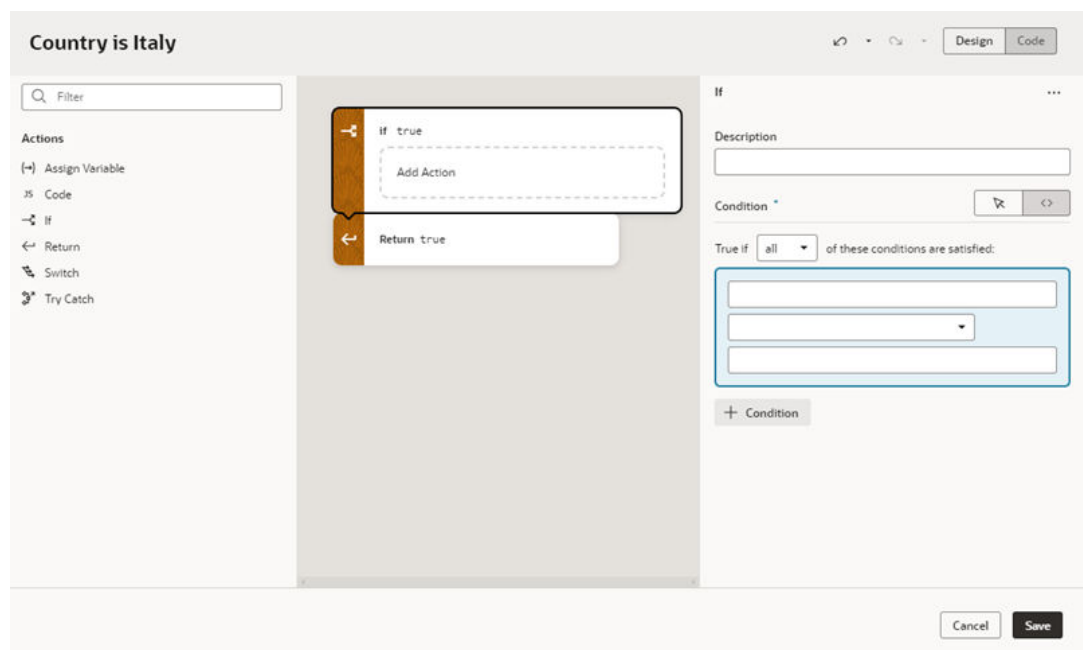


2. Drag an action from the palette and position it on the canvas.

Visual Builder 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:

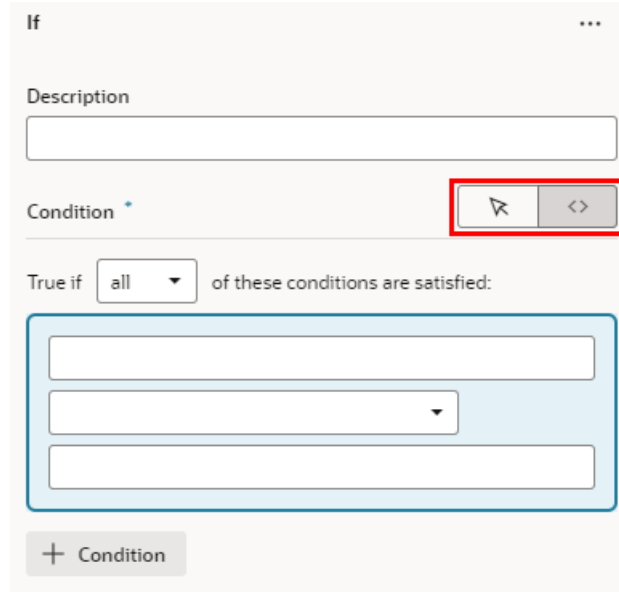


When the If action element is selected on the canvas, the Properties pane contains a Description field and a condition builder.

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.

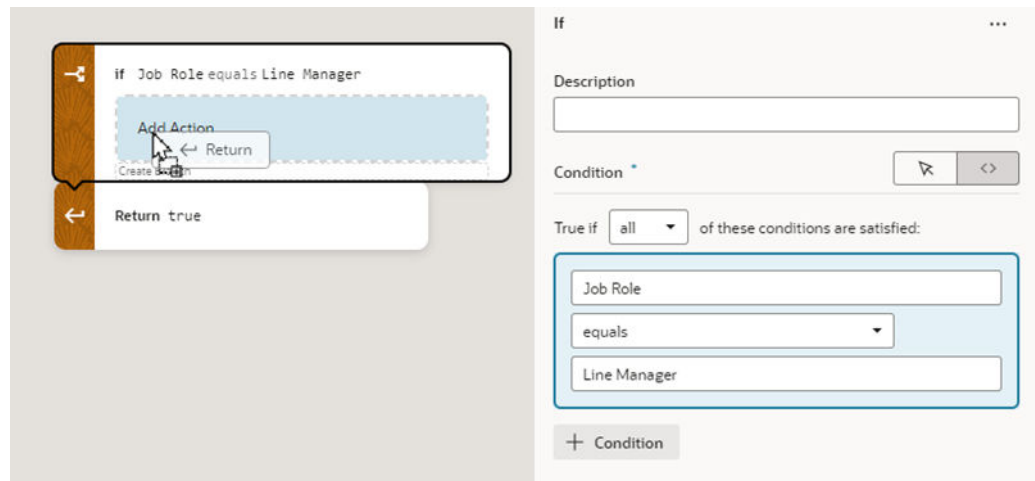
For an If action, the Properties pane has a condition builder for selecting the condition's criterion, operator, and value. Click **+ Condition** in the Properties pane to add a condition. Right-clicking a condition opens an options menu for adding, moving, and deleting the condition. You can also reposition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

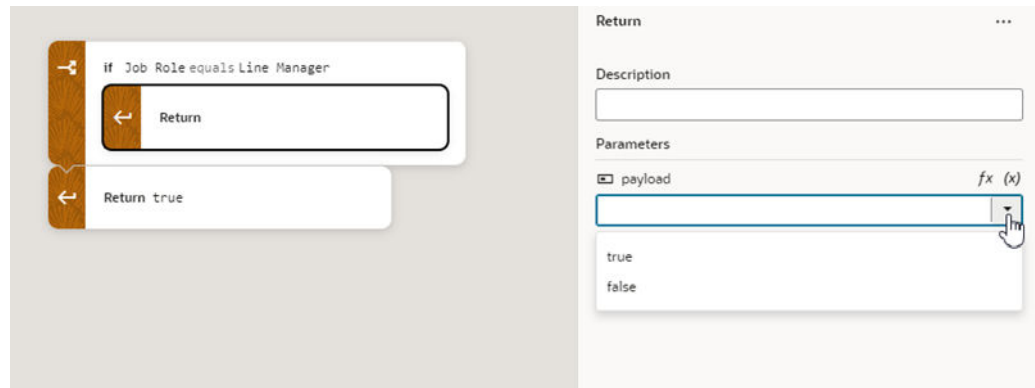


- b. On the canvas, drag actions from the palette (for example, Return, Assign Variable, Try Catch) into the action's Add Action drop target.

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:

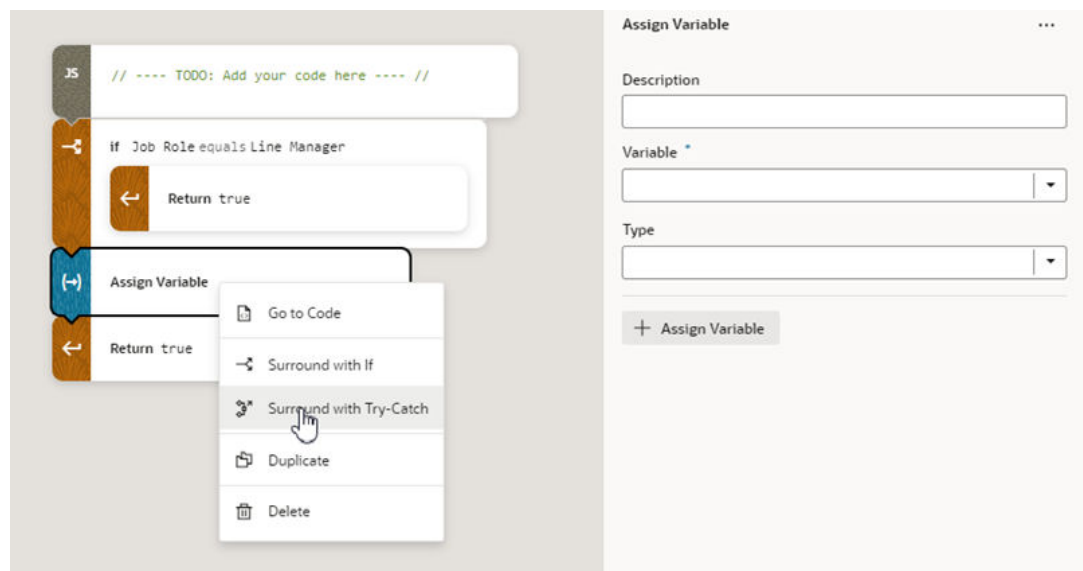


4. Repeat steps 2 and 3 to add more actions to the expression.

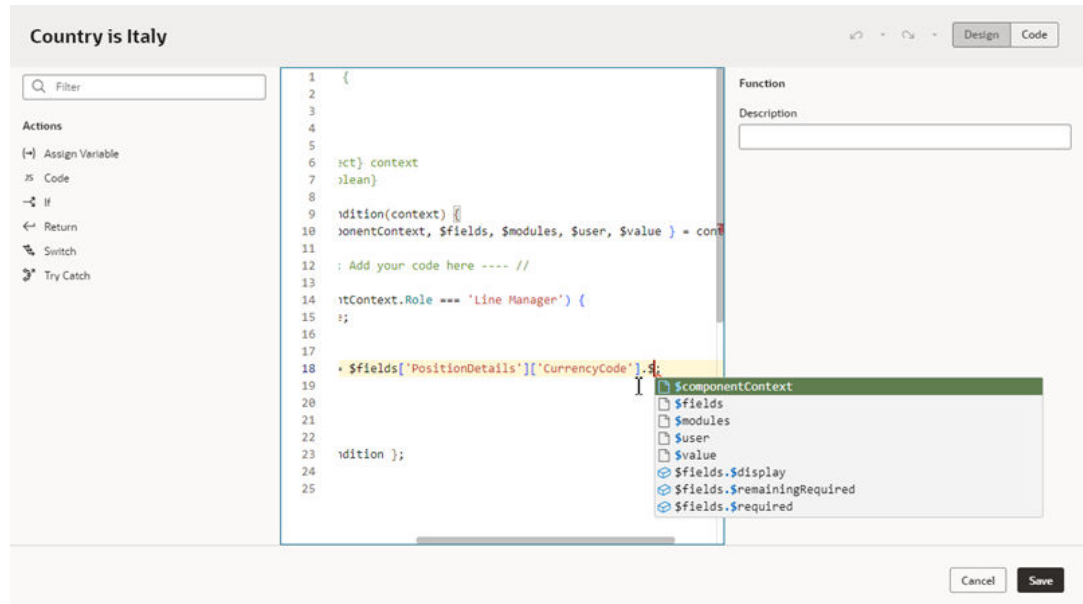
As you add more actions, you can reorganize the order by grabbing the front of the action element on the canvas, and then moving it into a new position:



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 mode. In Code mode you can type your expression directly, and take advantage of the 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:



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).

Here's a simple example:

Filter for fields, regions, rules... + Rule

Fields Regions Rules

▼ **Extension Rules** ⓘ ...

▼ Country is USA 2

▼ Full-time Employees 3 +

Hire Date before 2023 5

Hire Date after 2023 4

Country is Canada 1

▼ **Built-in Rules** ⓘ

Rule Role Regional Manager budgetDetails Hidden

Rule defaultValue Hidden

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.

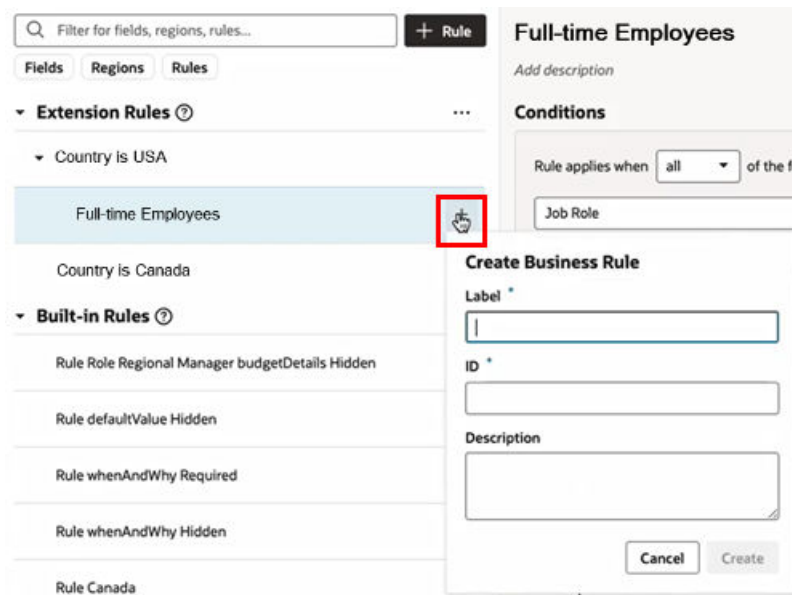
- If the rule's conditions are NOT met, the rule doesn't override any properties. In addition, its child rules (**Full-time Employees**) are never evaluated.
 - If the conditions for **Country is USA** ARE met, the property overrides defined in the rule are applied, and **Full-time Employees** is evaluated.
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.

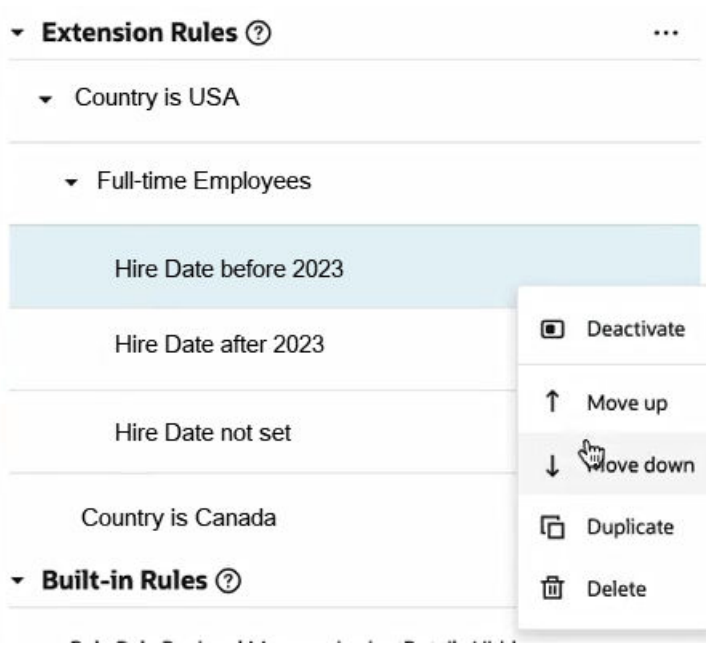
Add a child rule

You can add a child rule to any of your extension rules:

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 **Hire Date not set**, **Hire Date before 2023**, and **Hire Date after 2023**.



Note:

You cannot duplicate rules defined in extension dependencies (the rules listed under Built-in Rules), which includes the Default rule.

Set Properties For Regions and Fields

For each field or region 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.

Property settings affect the display only if the currently logged-in user meets the specified conditions. 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.

To set a property on a field or region:

1. Locate the field or region you want to modify.
2. 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:

The screenshot shows the 'Is Active' configuration page. On the left, there are tabs for 'Fields', 'Regions', and 'Rules', with 'Fields' selected. Below these are sections for 'Extension Rules' and 'Built-in Rules', with 'Is Active' selected under 'Built-in Rules'. The main area is titled 'Is Active' and contains a 'Conditions' section with the rule 'ActiveStatus equals "Active"'. Below that is the 'Regions and Fields' section, which has a table with columns: Required, Hidden, Read Only, and Value. The table lists various fields, including 'ActiveStatus', which is currently set to 'Required' and 'Hidden'. A dropdown menu is open for the 'Hidden' property of 'ActiveStatus', showing 'Visible' and 'Hidden' options. The 'Visible' option is selected.

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

If you change your mind after setting a property, use the **Remove Override** option to remove your setting and 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.

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.

 **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.

Identify the Regions and Fields Impacted By a Rule

Use the **Show only properties overridden by this rule** check box 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 properties overridden by this rule**.

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 **Project Manager in Australia** rule.

The screenshot shows the Oracle Business Rules Editor interface. On the left, a list of rules is displayed under 'Built-in Rules', with 'Project Manager in Australia' selected. The main panel shows the configuration for this rule, including its conditions and the 'Regions and Fields' section. The 'Regions and Fields' section is expanded, showing a table of impacted fields with their properties.

Field	Required	Hidden	Read Only	Value
When & Why Section (4)				
Action Reason	Optional			
customAAField		Hidden	Read Only	
customWWField		Hidden	Read Only	
Hire Date	Required			

Set a Default Value for a Field

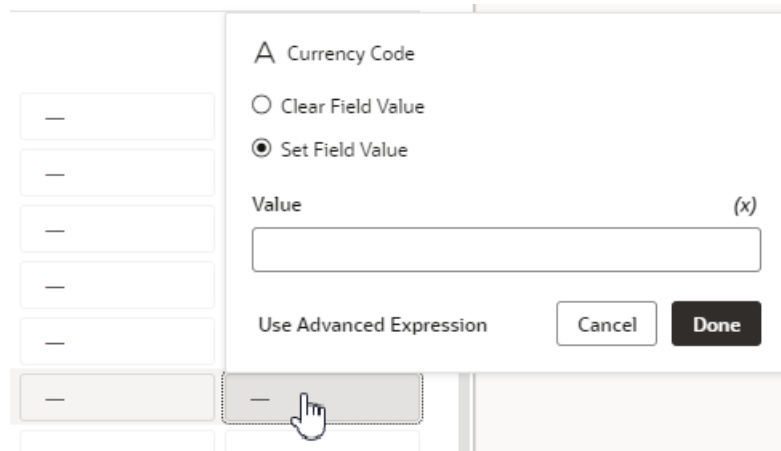
Use the *Value* property in a business rule to set a default value for a field.

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.

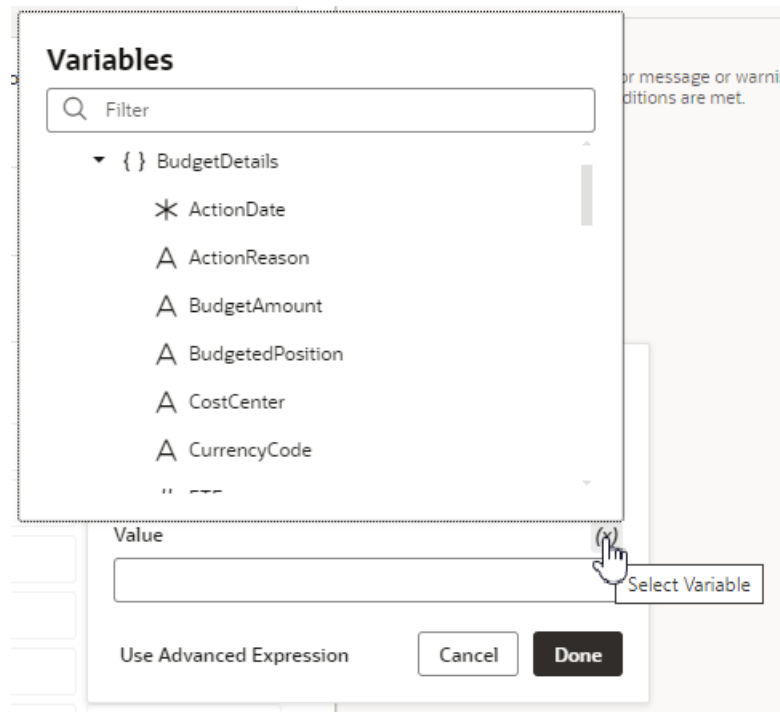
To set a value for a field:

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.

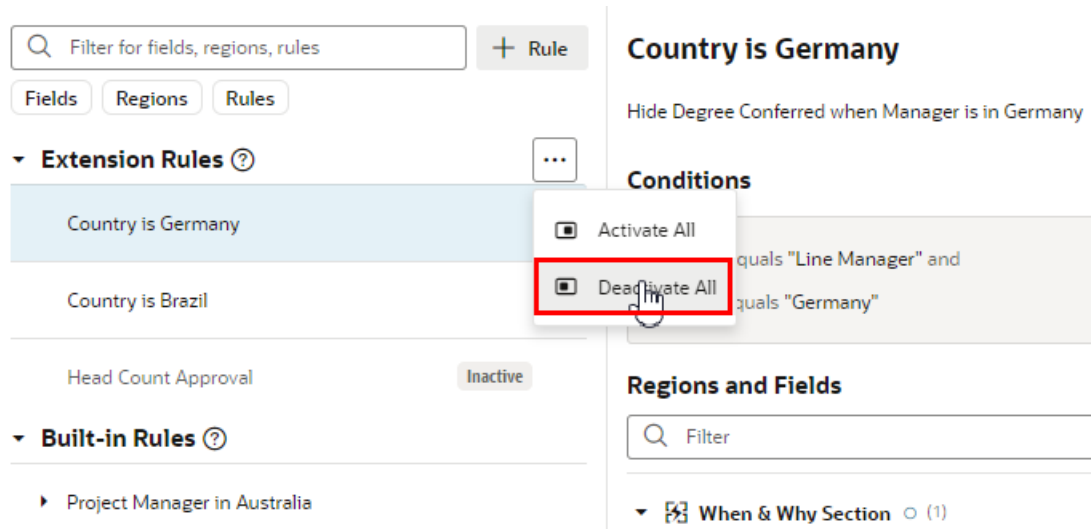
Set Properties at the Region Level

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.

See Out of the Box Behavior

You can deactivate all the extension rules by clicking the three dots next to the Extension Rules heading, then clicking **Deactivate All**:



This can be useful when debugging a page, allowing you see the page with only the built-in rules applied.

Note that 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.

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.

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.

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:

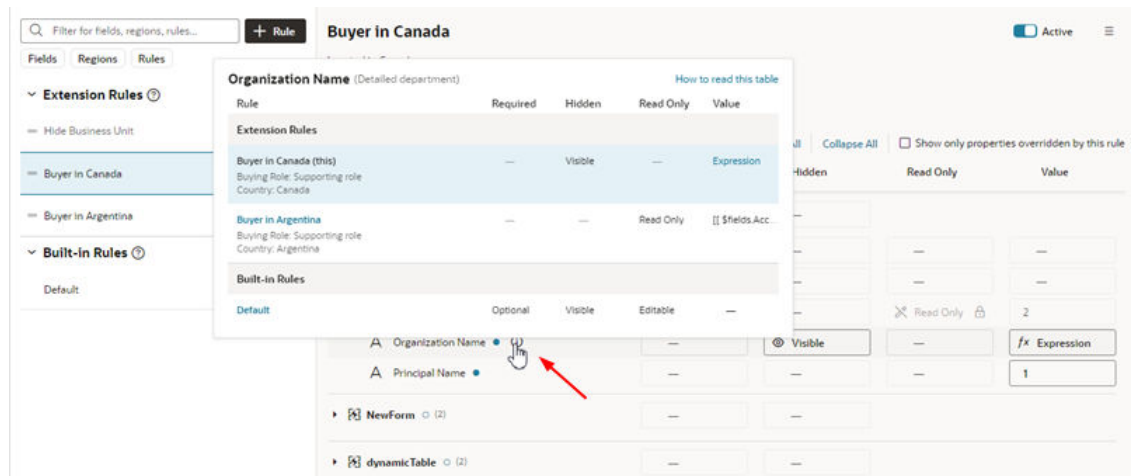
- A dash indicates that a value has not yet been set for the property by this rule. At runtime VB Studio evaluates all the rules from bottom to top, so a property setting for one rule can be overridden—and then overridden again—by rules that are higher up in the list.
- A lock icon means that a value 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:

1 Organization Name (Detailed department) [How to read this table](#)

Rule	Required	Hidden	Read Only	Value
Extension Rules				
Buyer in Canada (this) Buying Role: Supporting role Country: Canada	—	Visible	—	Expression
Buyer in Argentina Buying Role: Supporting role Country: Argentina	—	—	Read Only	[[\$fields.Acc...
Built-in Rules				
Default	Optional	Visible	Editable	—

2

3

4

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

1. Name of the field (Organization Name) and the region (Detailed department) 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:

Organization Name (Detailed department) [How to read this table](#)

Rule	Required	Hidden	Read Only	Value
Extension Rules				
Buyer in Canada (this) Buying Role: Supporting role Country: Canada	—	Visible	—	Expression
Buyer in Argentina Buying Role: Supporting role Country: Argentina	—	—	Read Only	[[\$fields.Acc...
Built-in Rules				
Default	Optional	Visible	Editable	—

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 its 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.

Display Messages When Conditions Are Met

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, and not 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:

The screenshot shows a form titled "Budget Details" with an information icon. At the top, there are radio button options for "Role" (Line Manager, Project Manager, Regional Manager, HR Specialist) and "Country" (USA, Canada, Japan, Australia). The "Project Manager" and "Canada" options are selected. Below the title, there are several input fields: "Budget Amount" (highlighted with a red box), "Currency Code", "Funded from Existing Positions", "Budgeted Position", "Head Count" (containing "1,001"), "CostCenter", and "FTE". A warning message "update budget" with a yellow triangle icon is displayed below the "Budget Amount" field. A "Required" label is visible at the bottom right of the form.

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** in the Business Rules pane.

Business Rules

Fields and Regions

Conditionally make a region or field visible, readonly, or required, or set the value for a field.

[Configure Fields and Regions](#)

Validations

Set up a response (usually an error message or warning) that's triggered when certain conditions are met.

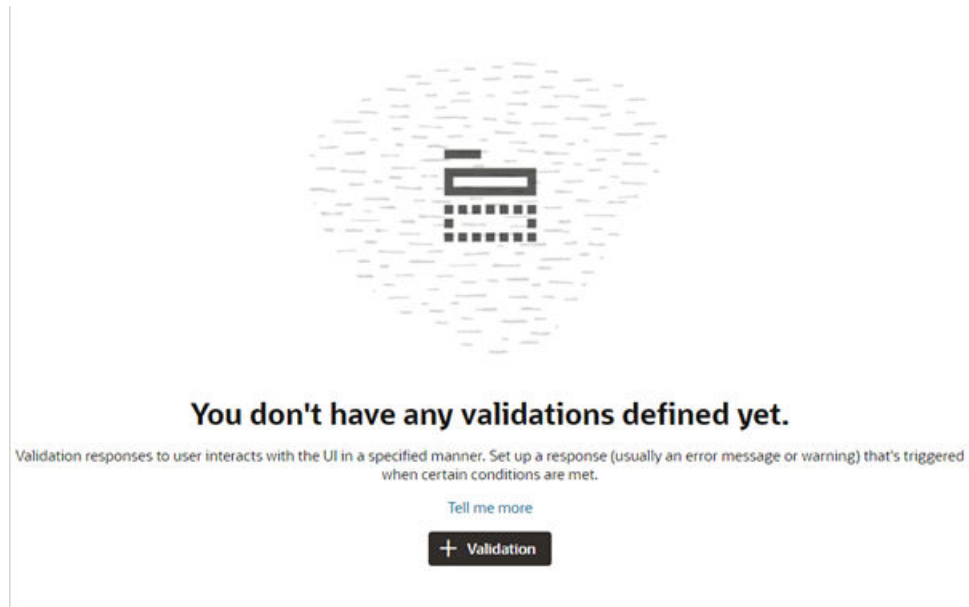
[Validate Field Values](#)

2. Create a new extension rule.

- If some rules are already defined, click [+ Rule](#), then enter a label and description.

The screenshot shows the Oracle Business Rules configuration interface. On the left, there is a search bar labeled 'Filter for fields, rules' and a '+ Rule' button highlighted with a red box. Below the search bar are tabs for 'Fields' and 'Rules'. Under the 'Rules' tab, there is a section for 'Built-in Rules' with a list of rules: 'Validate Full-time Employee' (highlighted) and 'Validate Line Manager Role'. The main area displays the configuration for the 'Validate Full-time Employee' rule. It includes a description, a 'Conditions' section with the expression '\$componentContext.StatusType === 'FullTime'', and a 'Messages' section with fields for 'Summary' (Security clearance), 'Severity' (Warning), 'Target Fields', and 'Detail' (Full-time employees must have a security code).

- If there are no rules yet, click **+ Validation**, and then enter a label and description in the popup.

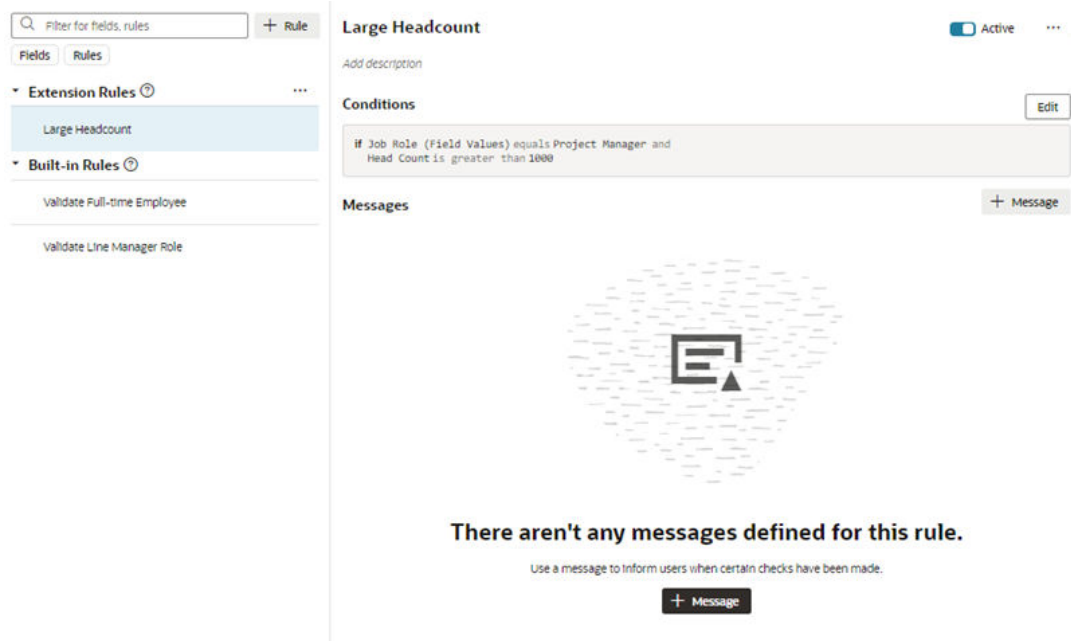


Click **Create**.

3. 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:



4. 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.]]
```

Messages

+ Message

The screenshot shows a configuration form for messages. It includes a 'Summary' text input field containing 'Feedback not allowed', a 'Severity' dropdown menu currently set to 'Warning', a 'Target Fields' dropdown menu which is empty, and a 'Detail' text area containing the message text: '[[Restricted feedback applies to \${fields.PersonName.\$value()} who is part of \${objectContext.Department}. Select someone from department without feedback restriction.]]'.

- 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.

The screenshot shows a configuration window for messages. At the top right is a '+ Message' button. Below it, the 'Summary' field is set to 'Budget' and the 'Severity' dropdown is set to 'Warning'. The 'Target Fields' dropdown is open, displaying a list of fields: Action Date, Action Reason, Budget Amount (highlighted by a mouse cursor), Budget Position, CostCenter, Currency Code, FTE, Funded from Existing Positions, and Head Count.

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 mode 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

i Budget Details

Budget Amount

⚠ update budget

Currency Code i

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:

Succession Plans

Business Rules


Fields and Regions
Conditionally make a region or field visible, readonly, or required, or set the value for a field.


Validations
Set up a response (usually an error message or warning) that's triggered when certain conditions are met.

Page Properties

A greetingText

A avatarInitials

>  Title

✓  Avatar

A avatarInitials

A avatarSize

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":

Business Rules

Fields and Regions
Conditionally make a region or field visible, readonly, or required, or set the value for a field.

[Configure Fields and Regions](#)

Validations
Set up a response (usually an error message or warning) that's triggered when certain conditions are met.

[Validate Field Values](#)

Page Properties

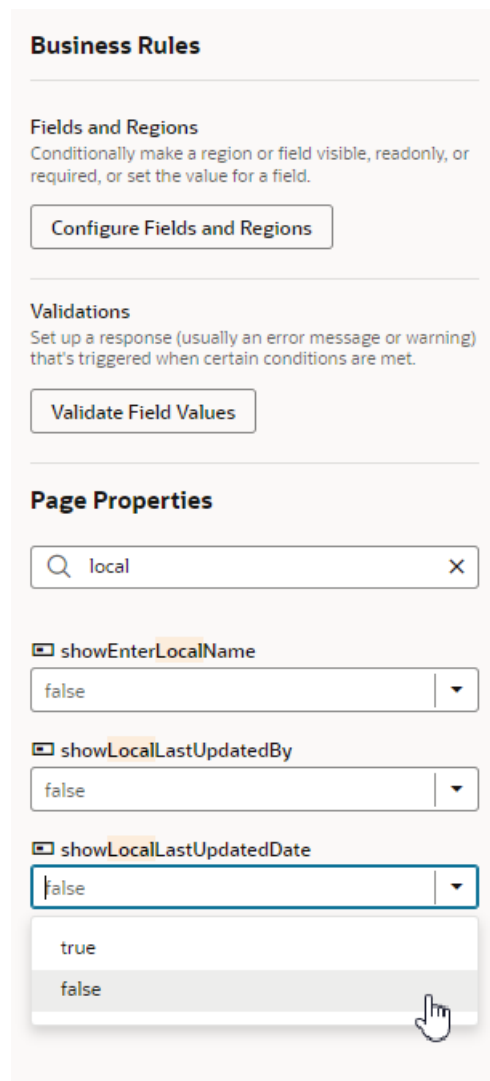
local

showEnterLocalName
false

showLocalLastUpdatedBy
false

showLocalLastUpdatedDate
false

true
false



Some page properties are 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:**

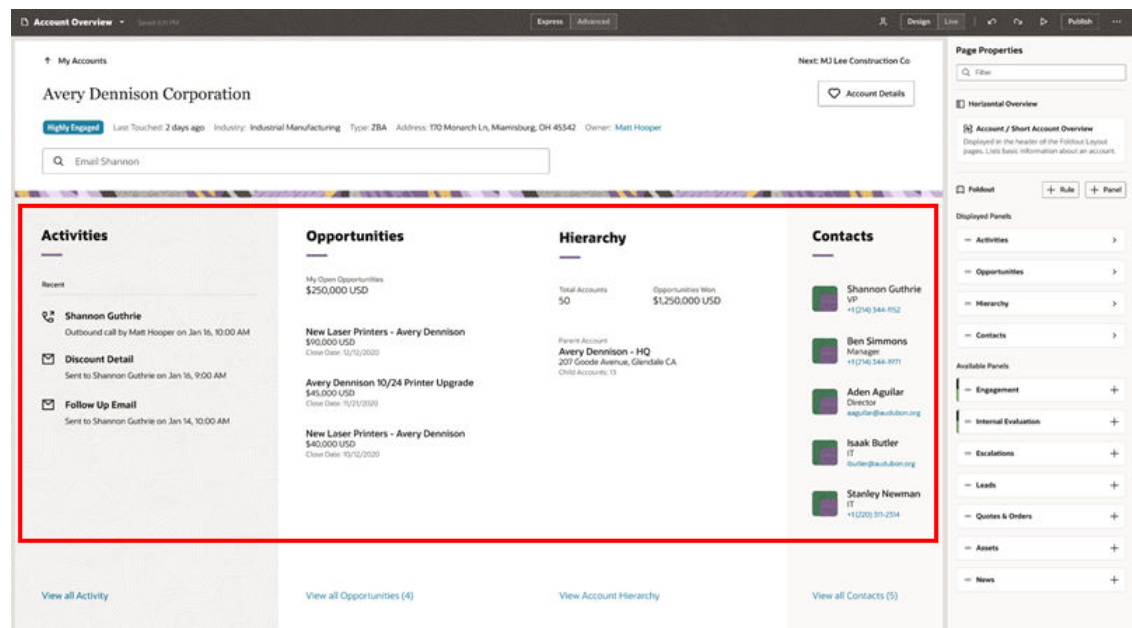
Oracle ensures that each page property is thoroughly documented (at least in the Human Capital Management Cloud Application), so be sure to use these descriptions as a resource as you work with page properties. It's a good idea to emulate this practice when creating your own page properties as well.

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.

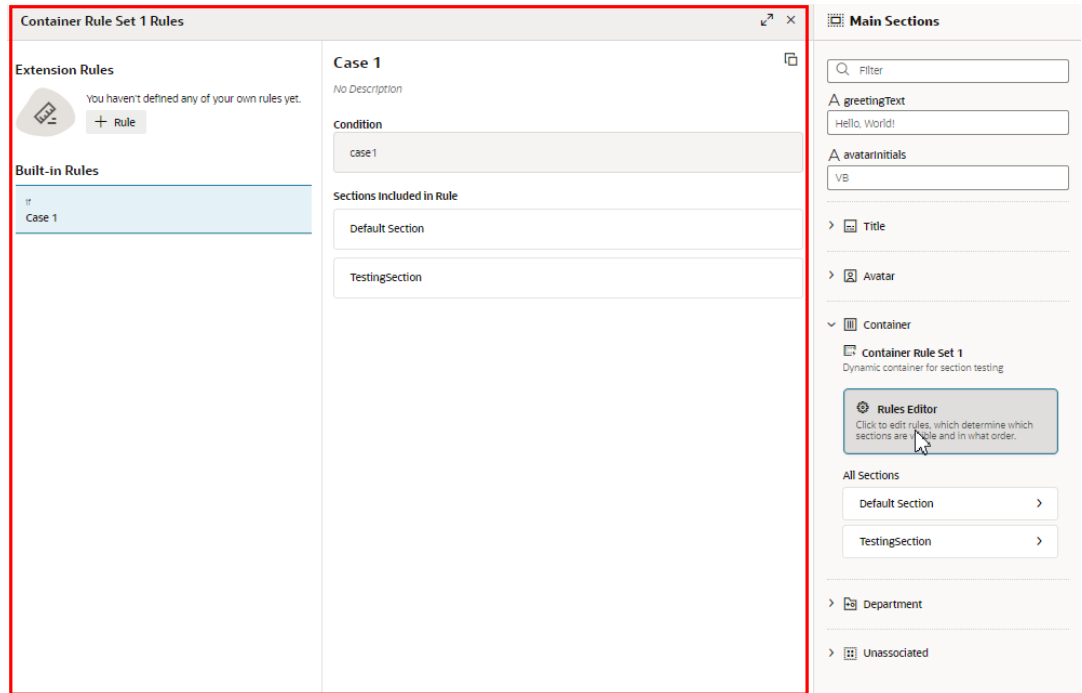
Control the Sections Displayed on the Page

The Properties pane lists all the sections that you can display on the page. To change the sections displayed in the container, you use rules to define which sections are displayed.

To change the sections displayed in a container:

1. In the Properties pane, click **Rules Editor** to open the editor.

The rules editor displays a list of rules for determining the sections displayed on the page. The rules listed under Built-in Rules are defined in the extension dependency, and you cannot edit them. When you create a rule it is listed under Extension Rules.



2. Click **+ Rule** in the editor, and then provide a name and description for the new rule.
If you want to duplicate a rule, right-click the rule in the list and select **Duplicate** in the popup options menu.
3. Change the order your extension rules are listed by selecting the rule, and then dragging it into the position you want.

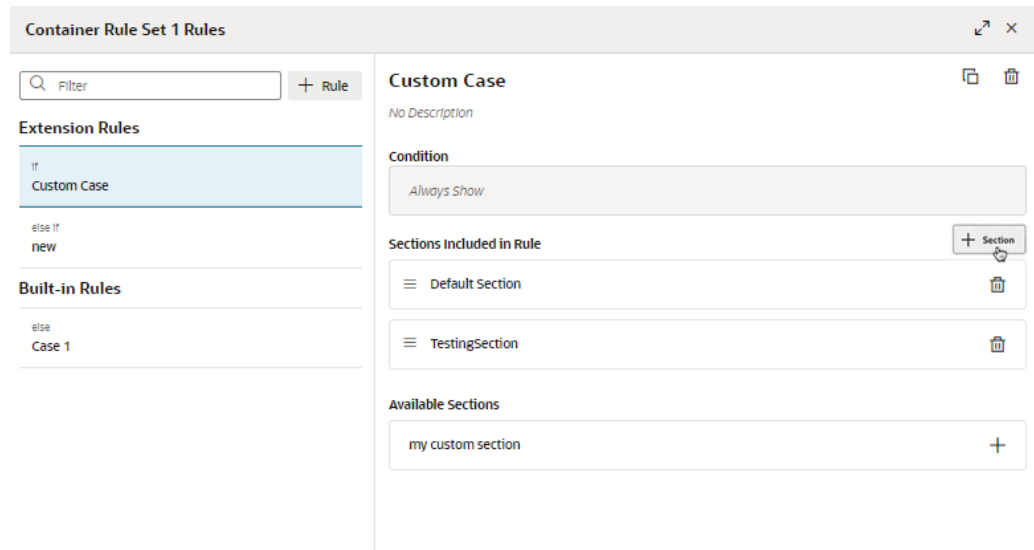
Rules are evaluated in order, from the top down. If a rule's conditions are met, the rule is applied, and none of the rules below it are evaluated. Your new rule is added to the top of the list of Extension Rules, so you need to make sure the rule is in the correct position.

If you want to delete a rule, right-click the rule and select **Delete** in the options menu.
4. Click **Edit Condition** in your rule, and then define its conditions in the condition builder.

The condition builder is the same one used to create conditions for extension rules in business rules. If you're not familiar with using the condition builder, see [Set Conditions for an Extension Rule](#).
5. Choose the sections you want displayed in the container.
 - 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 to a different position in the list.
6. Create new sections in the container.

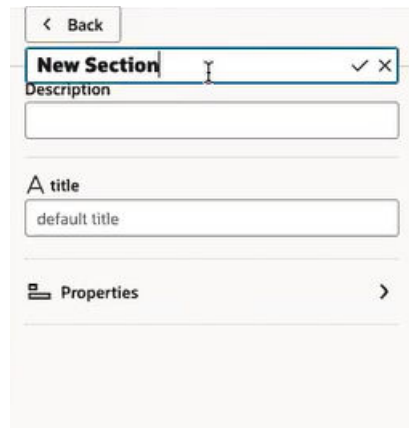
You can create sections when you are editing an extension rule. New sections are based on the container's templates defined in the extension dependency.
 - a. Click **+ Section**.



When you add 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. The template is defined in the extension dependency.

- d. Click **Back**.
7. Click X in the top right of the editor to close it.

The Designer displays the container with the sections defined in your extension rule.

Configure the Content Displayed in a Section

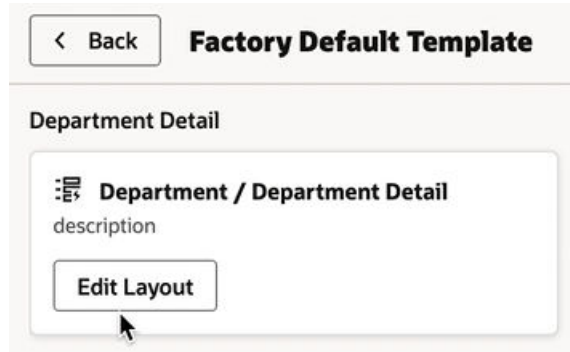
Some sections have content that you can modify. For example, a section might contain an editable fragment, or a form that you can edit to hide or add fields.

To configure the content displayed in a section:

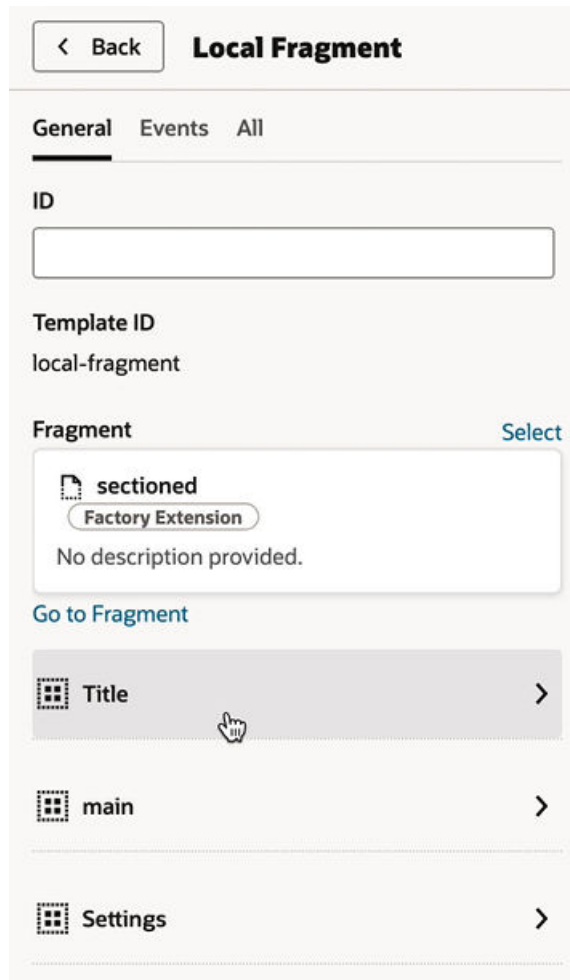
1. In the Properties pane, select the section you want to modify.

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:

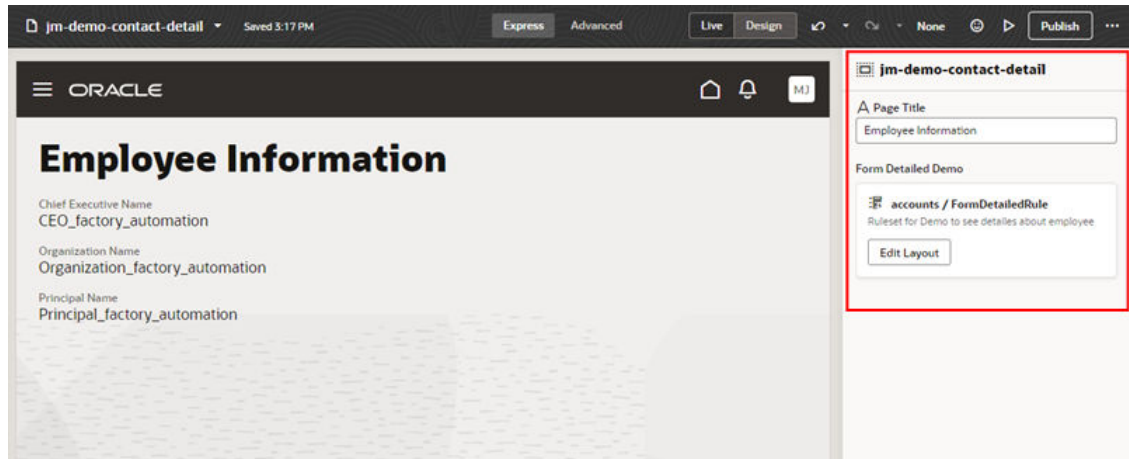


2. Click **Back** to return to the page's Properties pane.

5

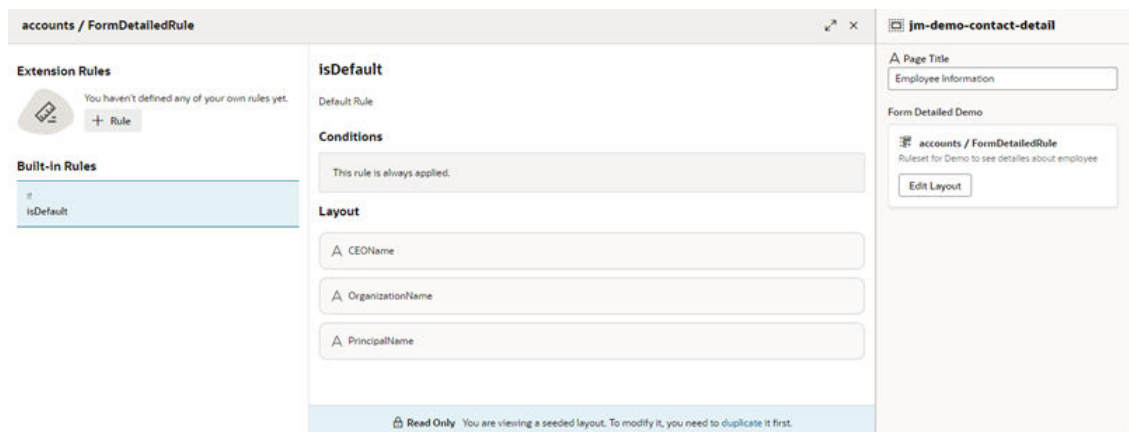
Control Your Display With Rule Sets

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 the FormDetailedRule rule set looks like after you click Edit Layout to 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*.

Create an Extension Rule in a Rule Set

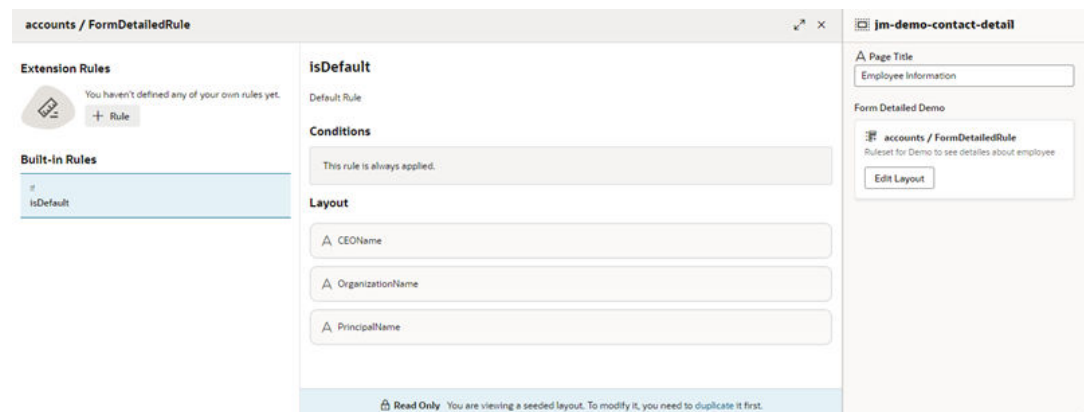
Once you've opened a rule set, you can create a rule by defining the rule's conditions, and which fields are displayed in its layout when the rule's conditions are met.

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.

1. Open the page you want to modify in the Designer.
2. Locate the component layout you want to edit on the page or in the Properties pane, and then click **Edit Layout** to open the 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. This rule set has one built-in rule (isDefault). To the right, the selected rule's conditions are displayed above the list of fields in the rule's layout.

3. Click **+ Rule** in the left pane to create a new extension rule, and then provide a name and description in the popup dialog.

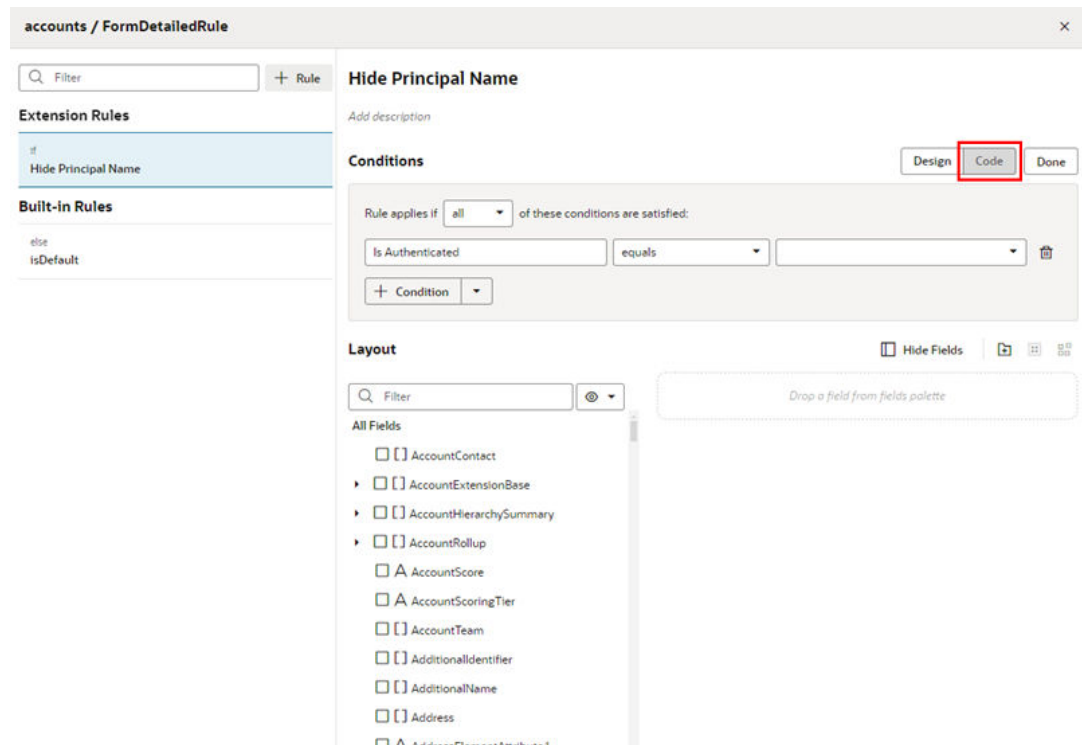
You can delete or duplicate a rule by right-clicking the rule, and then selecting **Delete** or **Duplicate** in the popup options menu.

4. Change the order of the extension rules in the list by grabbing a rule 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. Click **Edit** in your rule to open the condition builder, and then 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 an Extension Rule](#).



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

6. Select the fields you want displayed in the layout.

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:

Hide Principal Name

Add description

Conditions

Edit

if Is Authenticated true

Layout

Hide Fields


The screenshot shows the layout editor interface. On the left, there is a 'Filter' search box and a list of fields with checkboxes. The 'CEO Title' field is checked and highlighted with a red box. On the right, the layout editor shows a vertical stack of fields: 'CEO Name', 'Principal Name', 'Organization Name', and 'CEO Title'. The 'CEO Title' field is currently selected and highlighted with a blue border. Below the layout is a dashed box with the text 'Drop a field from fields palette'.

The layout editor contains a Fields palette listing all the fields that can be displayed in the layout. This palette is displayed by default when you create a new rule. You can click **Hide Fields** in the toolbar to hide the Fields palette. If the palette is hidden, click **Show Fields**.

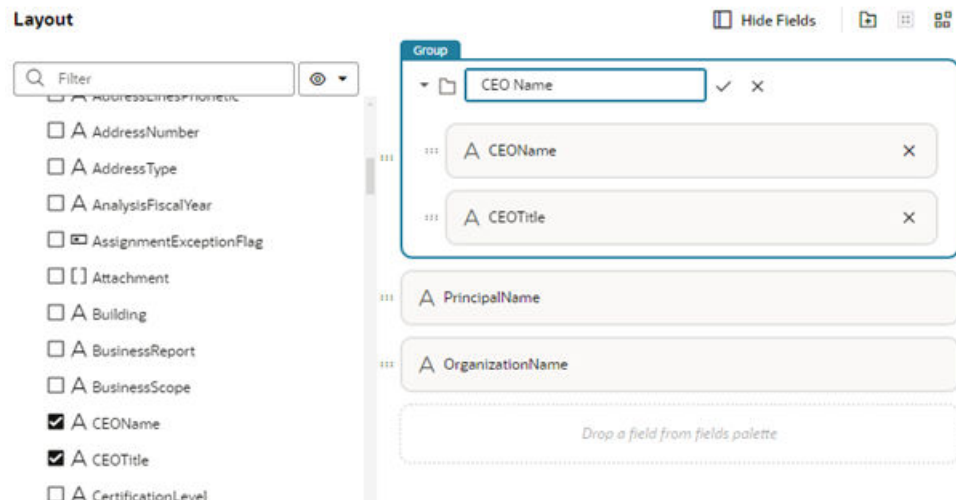
The fields you can display in a rule's layout are determined by the fields available from the data resource used by the component. This data resource is defined in the extension dependency. You can choose any of the fields listed in the Fields palette—and the order in which they should appear—but you 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:

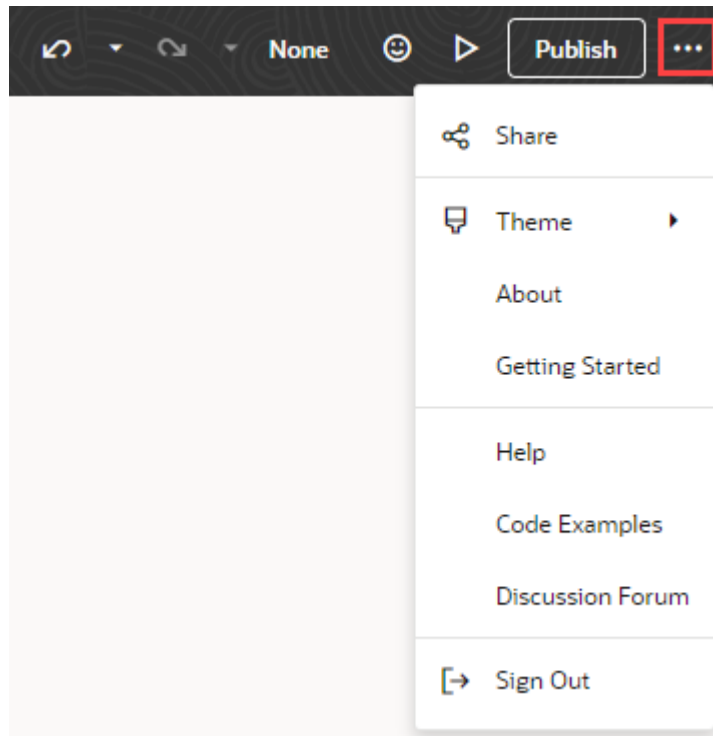
- To preview your page, click **Preview** in the Designer's header:



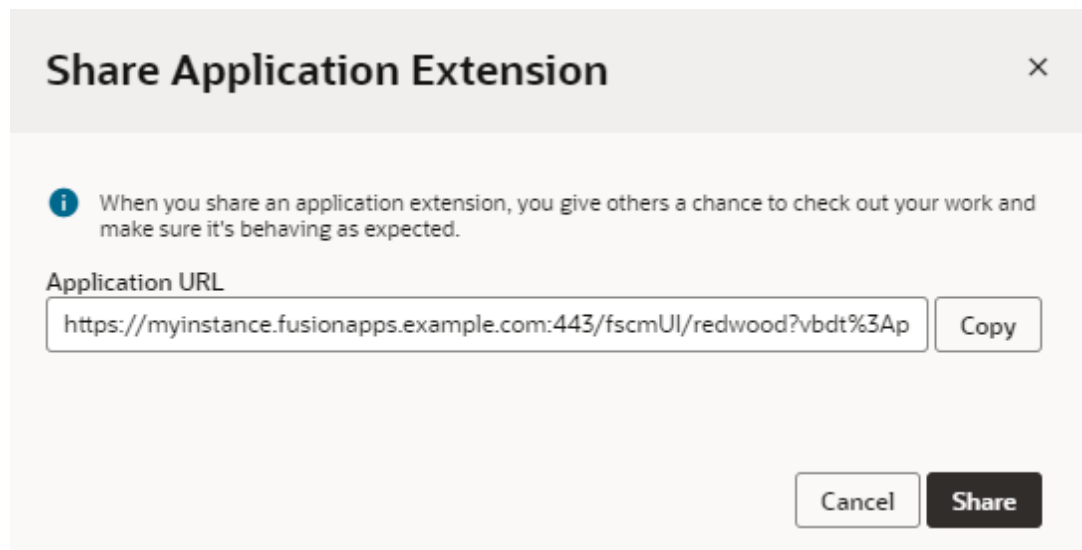
This opens your page in another browser tab. You may be asked to re-enter your credentials for the Oracle Cloud Applications instance before the tab opens.

As the name implies, Preview gives you a chance to test your page before sharing it with others. If you prefer, you can also use Live mode to test your page, though the form factor is a bit more pleasing when using Preview.

- 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:



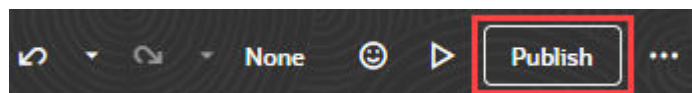
Click **Copy**, then share the URL with the team members who want to test your changes via an email or direct message (DM). Make sure you click **Share** *before* you click **Copy**, as otherwise the URL will not be valid.

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**:

Publish ✕

You're about to publish these changes:

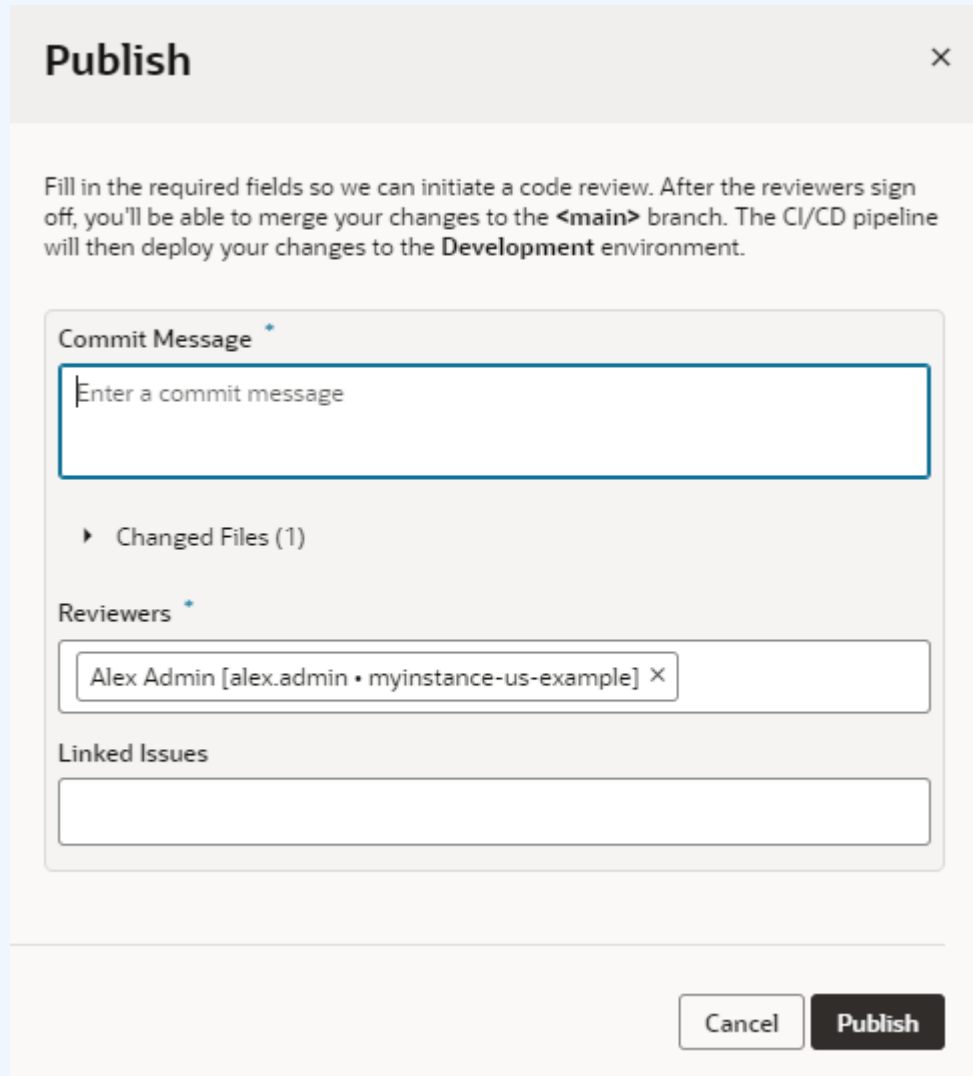
- ✓ Modified Readme.md ⌵

Briefly describe your work: *

Cancel Publish

 **Note:**

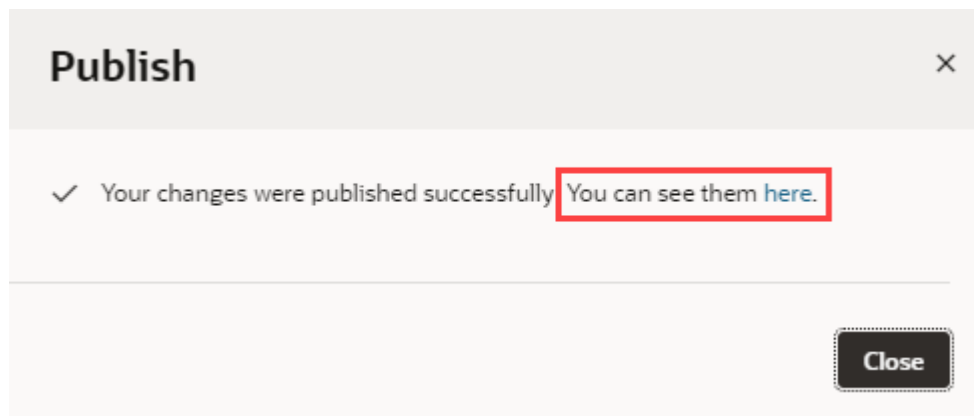
Your view of the Publish dialog may be different based on some advanced settings. If you're asked to enter a *commit* message instead (as shown here), it means your administrator has set things up so that all changes are first reviewed and approved by a developer, who will actually look at the code representing your changes. To get your changes reviewed, enter a message in **Commit Message**, add at least one reviewer if no name is specified, then click **Publish**:



The screenshot shows a 'Publish' dialog box with a close button (X) in the top right corner. Below the title bar, there is a paragraph of text: 'Fill in the required fields so we can initiate a code review. After the reviewers sign off, you'll be able to merge your changes to the <main> branch. The CI/CD pipeline will then deploy your changes to the **Development** environment.' Below this text is a form with several sections: 'Commit Message' with a text input field containing the placeholder 'Enter a commit message'; 'Changed Files (1)' with a right-pointing arrow; 'Reviewers' with a dropdown menu showing 'Alex Admin [alex.admin • myinstance-us-example] X'; and 'Linked Issues' with an empty text input field. At the bottom right of the dialog are two buttons: 'Cancel' and 'Publish'.

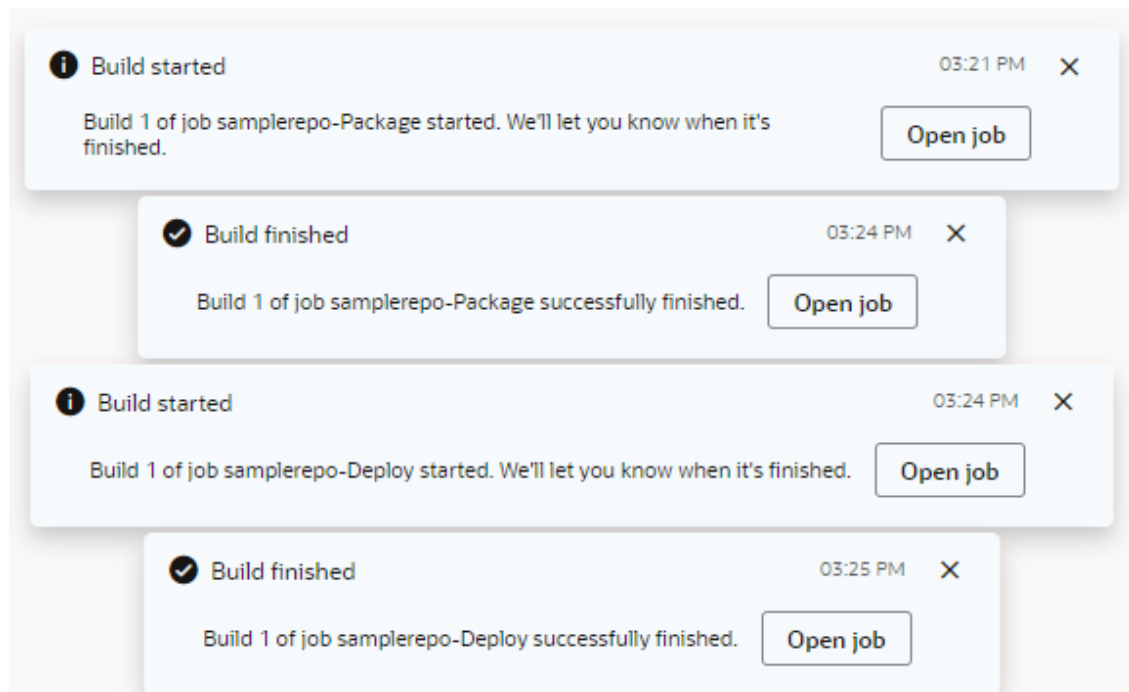
3. If prompted, re-enter your user name and password for the Oracle Cloud Applications instance you're working with and click **Add Credentials and Continue**.

At this point, you'll be notified that your changes were successfully published. You may even be able to simply click a link to see your deployed changes, as shown here:



(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 that's not the case, you'll see notifications indicating that the publish process is progressing smoothly:



When publishing is complete, go to the Oracle Cloud Applications instance in your environment to see your changes.

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

It's usually the responsibility of your VB Studio administrator to do whatever's required to deploy your changes to the **production** environment (by following the steps in Set Up the Project to Deploy to Production). However, if you need to do this yourself, see [Publish Your Extension to Other Instances](#).

Publish Your Extension to Other Instances

By default, the **Publish** button deploys your extension to the Oracle Cloud Applications instance you're currently working in, which is typically an instance used for testing. In some cases, however, you may want to deploy to a different instance, like a development instance or possibly even to production.

As long as you have valid credentials for these instances, you can set up the necessary jobs to publish your extension to them manually (that is, **not** using the Publish button). In VB Studio, a *job* defines where to find the source code files, along with other specifications needed to complete the work. When you click **Publish** in Express mode, VB Studio kicks off two jobs behind the scenes:

- A packaging job, which gathers all your extension's assets into a single bundle
- A deployment job, which actually deploys the extension bundle to the target instance.

To deploy to a different target instance, you can define your own packaging and deployment jobs and then combine them into a single *pipeline*, which you can then run whenever you need to. Just follow this step-by-step procedure and you'll be ready to go!

 **Note:**

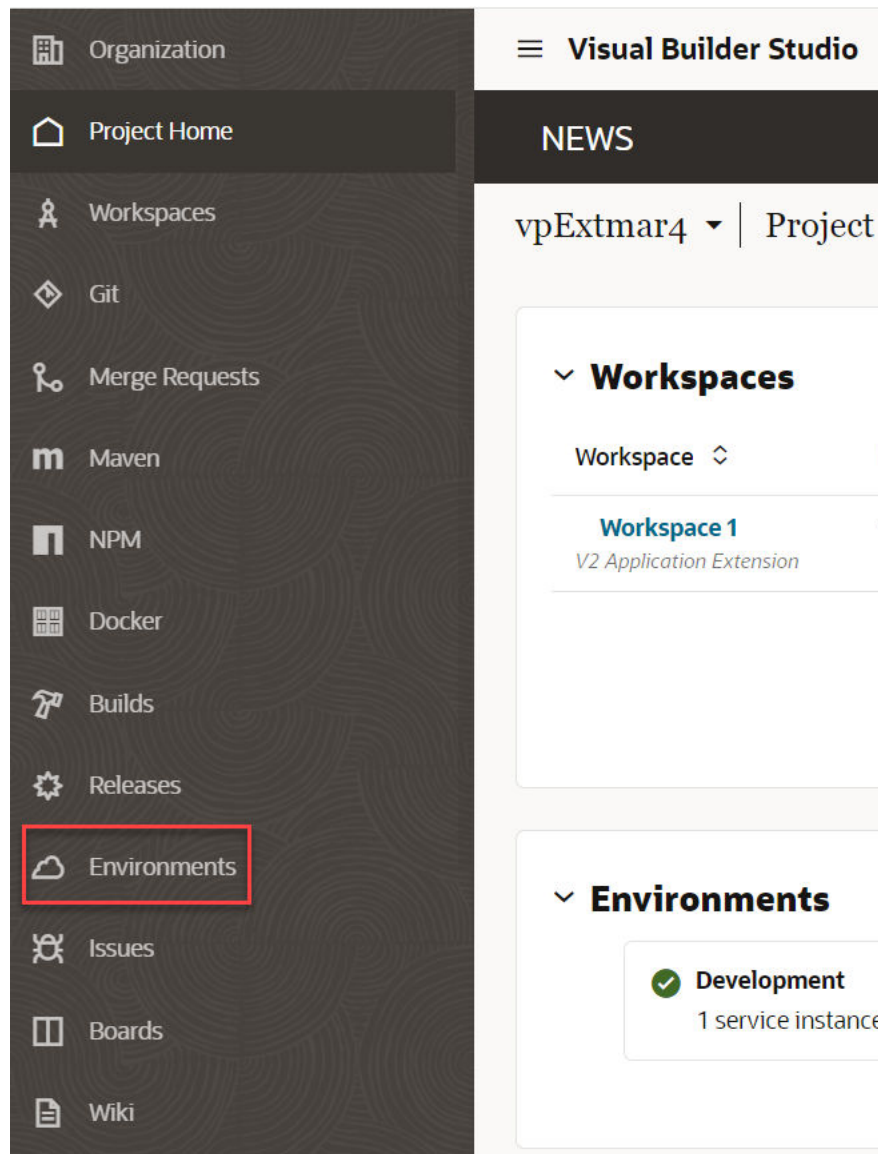
You can use this procedure for another test instance, a development instance, or a production instance, depending on your needs.

Before you begin, make sure you have the URL of the Oracle Cloud Applications instance where you want to deploy your extension, along with a valid user name and password.

1. In the header area, click **Advanced**.

Express mode is meant to be a simple, streamlined experience, so setting up jobs is not supported there. You'll need to use Advanced mode instead.

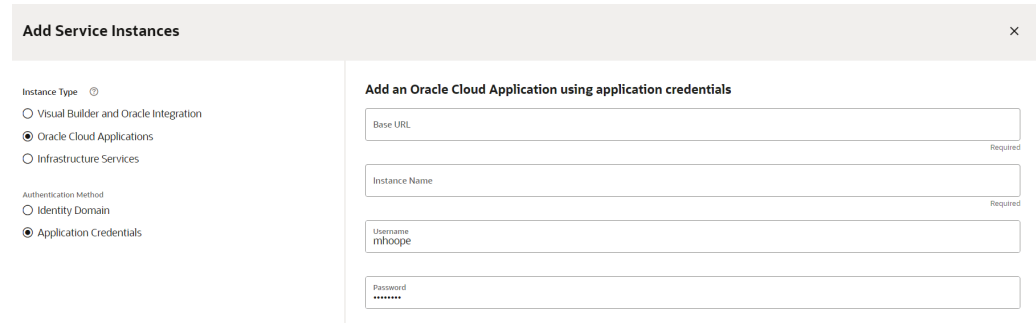
2. In the left navigator, click **Environments**:



In VB Studio, you need an *environment* to define the Oracle Cloud Application 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. You need a new environment to point to the new target instance instead.

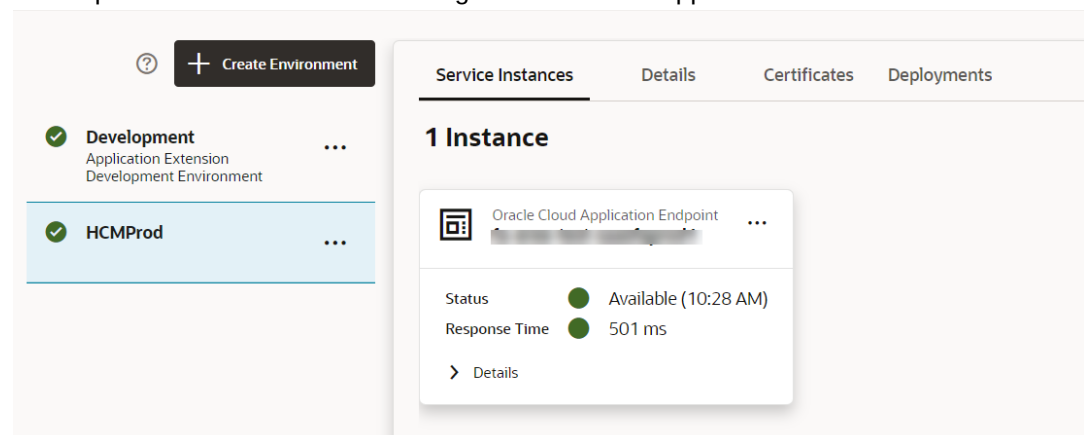
3. Click **+ Create Environment**.
 - a. Assign a name for the environment (like <name>Production for a production instance) and an optional description.
 - b. Click **Create**.
4. Click **+ Add Instance**.
 - a. On the left, under Instance Type, click **Oracle Cloud Applications**.

- b. Under Authentication Method, click **Application Credentials**.



- c. In the **Base URL** field, enter the URL of the Oracle Cloud Applications instance you want to deploy to. The **Instance Name** field will populate automatically.
- d. Enter your user name and password.
- e. Click **Add**.

At this point, a little box appears on the Service Instances tab to show you the status and the response time for the instance. A green dot should appear next to the **Status** field:



If it doesn't, it may be because you mistyped the user name or password. Click the three dots in the upper right corner and select **Edit Credentials** to try again.

Now that you've set up your environment, you're ready to create your own package and deploy jobs.

5. In the left navigator, click **Builds**.

At a minimum, you should see the `Application-Extension-Deploy` and `Application-Extension-Package` jobs already listed, which you can use to make your own task a little easier. To create the package job:

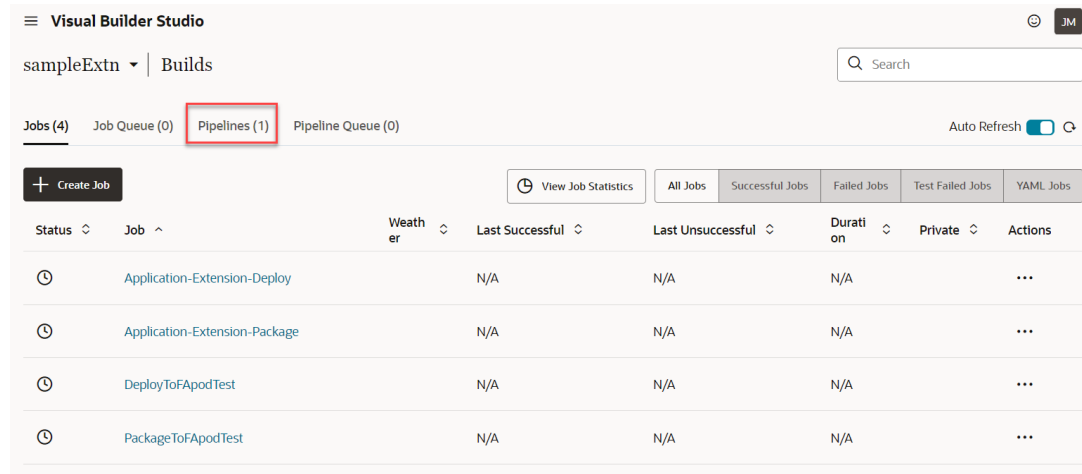
- a. Click **+ Create Job**.
- b. Enter a name for the job. If you have several Oracle Cloud Applications instances, it's a good idea to include the name of the target pod, as well as what it's used for (test, development, or production); for example, `PackageTo<instance name>Test`.

- c. Click the **Copy from Existing** checkbox, then select `Application-Extension-Package` from the dropdown:

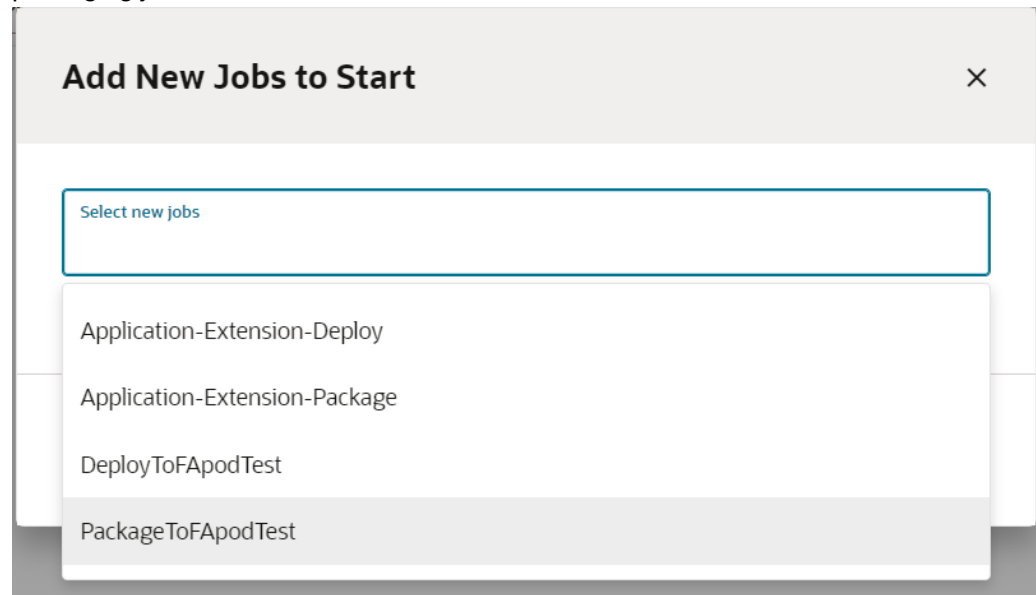
The screenshot shows a 'New Job' configuration window. The 'Name' field is filled with 'PackageToFApodTest'. The 'Description' field is empty. The 'Copy From Existing' checkbox is checked. The 'Copy From' dropdown menu is open, showing 'Application-Extension-Package' as the selected option. The 'Use for Merge Request' checkbox is unchecked. The 'Template' field is set to 'System Default OL7 for Visual Builder', with 'System' and 'Oracle Linux 7' buttons visible. At the bottom right, there are 'Cancel' and 'Create' buttons.

- d. Click **Create**.
- At this point you're sent to the Configure Git screen. A Git repository is where the source files for your extension are stored. You're going to run this job manually (or through the pipeline), so you don't want anything triggered automatically.
- e. De-select **Automatically perform build on SCM commit**, then click **Save**.
Now let's work on creating the deploy job.
- f. Repeat steps a-d, again giving the job an easy-to-identify name (like `DeployTo<instance name>Test`) and selecting the `Application-Extension-Deploy` job to copy.
This time when you get to the Configure Git screen, you'll need to choose the service instance you want to deploy to.
- g. On the Job Configuration screen, click the **Steps** tab.
- h. Select the instance you want to deploy to from the **Target Instance** drop-down.
- i. Enter the user name and password for this instance.
You've now created both of the required jobs to deploy your extension to an Oracle Cloud Applications instance that's different from the default. To make things easier, you may want to create a *pipeline* now, which kicks off first the packaging job, then the deploy job, so you don't have to run each job separately.

6. On the Builds page, click the **Pipelines** tab:

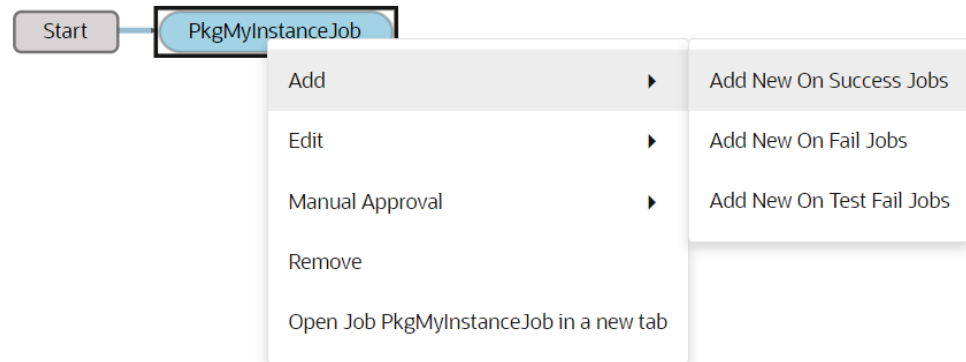


- Click **+ Pipeline**.
- Enter a name, perhaps `PipelineFor<instance name>Test`.
- De-select the **Auto-start when pipeline jobs are built externally** check box, then click **Create**.
- On the Pipeline Configuration page, right-click the **Start** button in the middle and select **Add New Start Jobs**.
- On the Add New Jobs to Start screen, click the **Select new jobs** field, then select your packaging job:



- Click **Save**.

- g. Click the image for the job you just added, then select **Add->Add New On Success Jobs**.



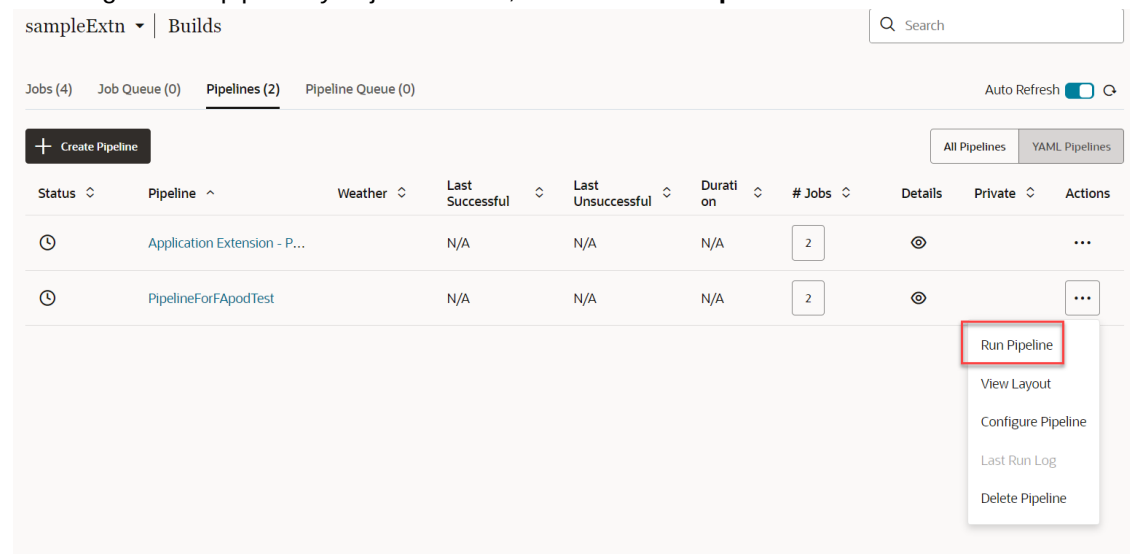
This ensures that the deploy job will run **only** if the packaging job is successful.

- h. In the Add New On Success Jobs dialog, click the field, then select your deploy job.

Note: You're not limited to just one deployment target in a pipeline. If, for example, you want to deploy to two or three instances simultaneously, just create jobs for those instances and repeat step 6g. to add them to the pipeline.

- i. Click **Save** in the upper right to save the pipeline.

Now, whenever you want to run this pipeline, just go to **Builds->Pipelines**, click the three dots to the right of the pipeline you just created, and click **Run Pipeline**:



Use **Configure Pipeline** to add new jobs or make other changes to the pipeline, if needed.

To see the status of the pipeline build, click the name of the pipeline under **Pipeline**.

Resolve Conflicts

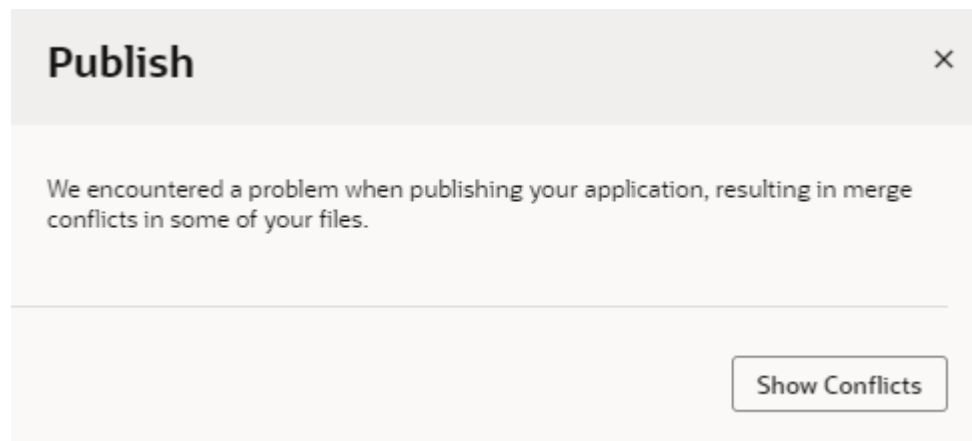
Sometimes when you publish changes, you might run into conflicts. Conflicts usually occur when you and your teammate make changes that overlap or conflict with each other. For example, both of you might have changed the same line in a file in different ways, or one of you might have deleted a file while 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:



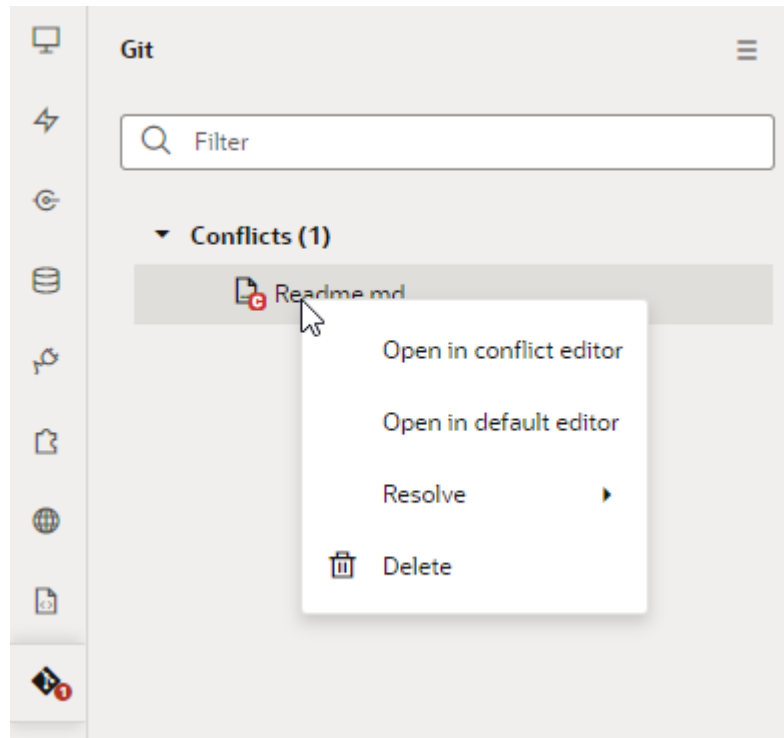
Here's what to do when this happens:

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`):



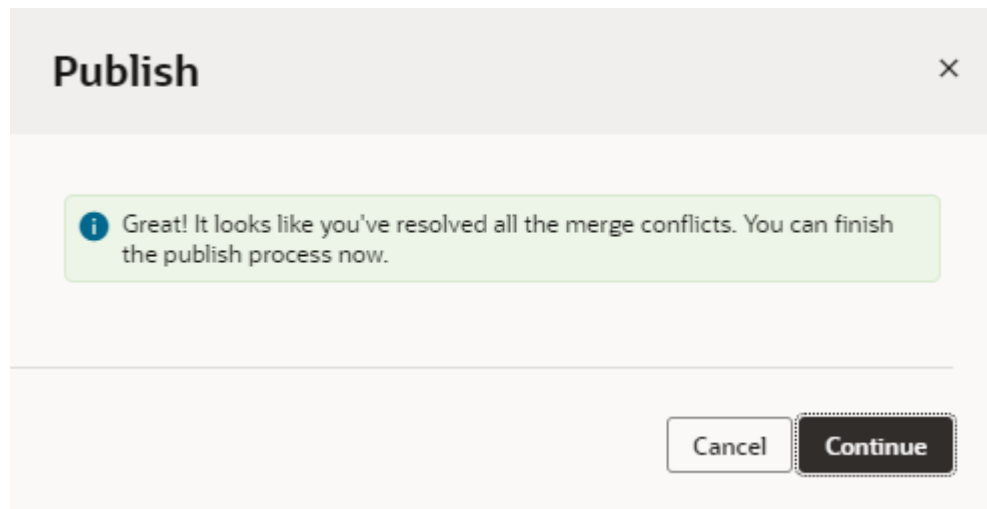
Action	Steps
Open in conflict editor	Select this option to open the file in the conflict editor; you can also just click the file to open it in the conflict editor. The conflict editor has controls to help you navigate between conflicts and provides options to resolve them. See Use the Conflict Editor 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.



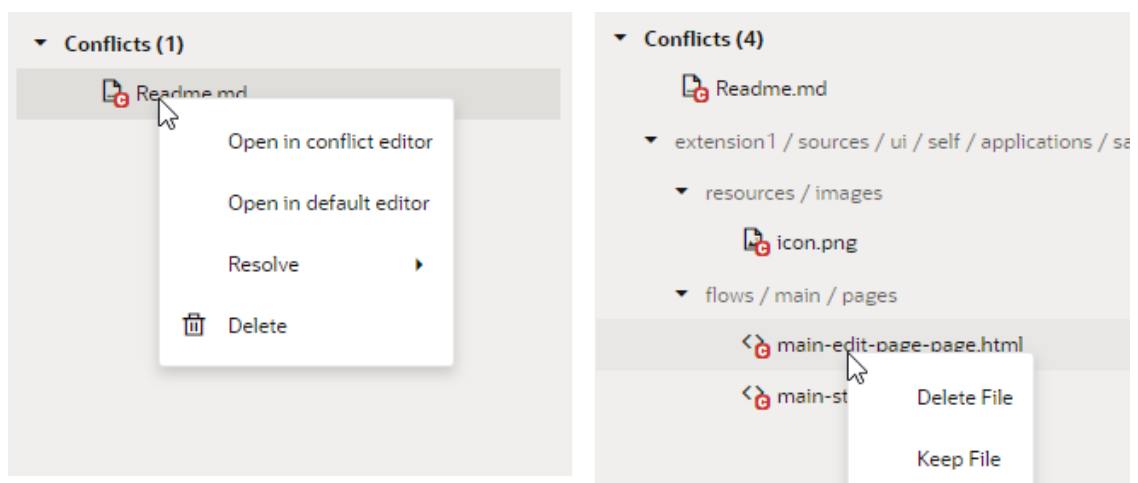
Use the Context Menu to Resolve Conflicts

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:

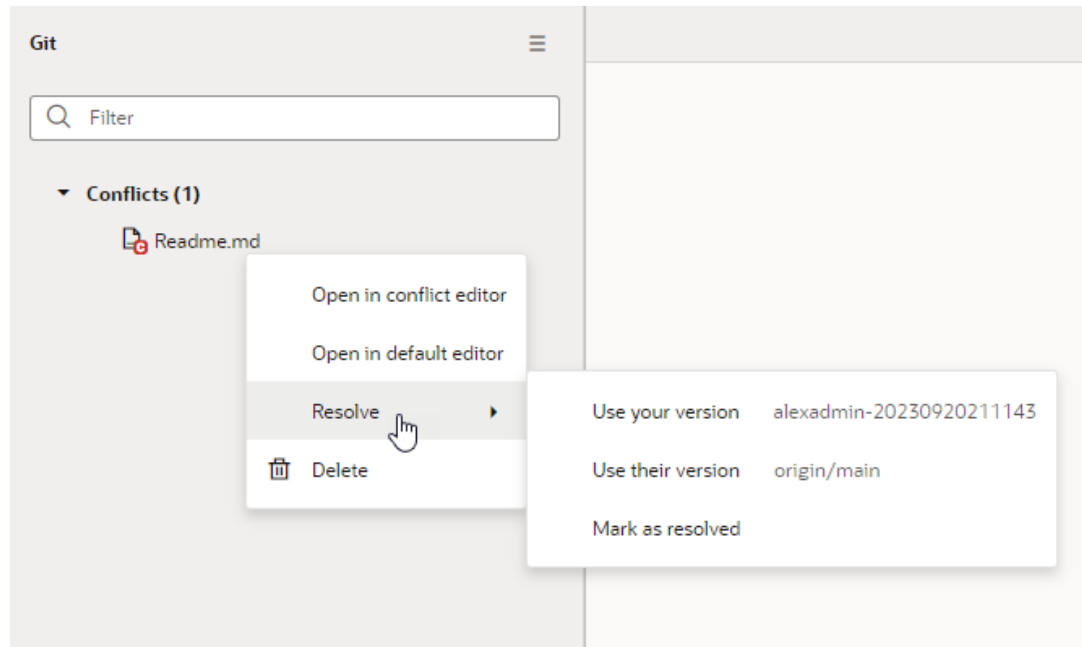
If you want to keep changes from both branches (or a combination of them), or if the file has a large number of conflicts, it's better to [open the file in the conflict editor](#) and resolve each conflict separately.

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:

- 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	<ul style="list-style-type: none"> Select Use their version in the toolbar, or Click  in the canvas and select Their Version.
To keep your changes in the local branch	<ul style="list-style-type: none"> 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.

- 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.