Oracle® Cloud

Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel

Version 4.1
F96308-01
July 2024

Oracle Cloud Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel, Version 4.1

F96308-01

Primary Author: Oracle Corporation

Contents

5 Manage Catalogs and Business Objects

6 Configure Search Options for Download

7 Download Data

8 Custom Actions

9 Use Lists of Values in an Excel Workbook

10 Key Flexfield Support

Preface

Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel describes how to develop Excel workbooks that can retrieve and modify business data exposed by a REST service and send modified data back to the service.

Topics:

Audience

Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel is intended for developers who want to create and publish Excel workbooks that integrate with enterprise applications that they use.

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.

Third-Party Content, Products, and Services Disclaimer

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.

Related Resources

For more information, see these Oracle resources:

- Oracle Visual Builder Add-in for Excel homepage.
- Oracle Visual Builder for Excel Add-in Downloads
- View and Edit Data Using an Excel Workbook in *Managing Data Using the Oracle Visual Builder Add-in for Excel*

Conventions

The following text conventions are used in this document.

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

1

What's New in Release 4.1.0

Here's an overview of new features and enhancements added to Oracle Visual Builder Add-in for Excel in Release 4.1.0.

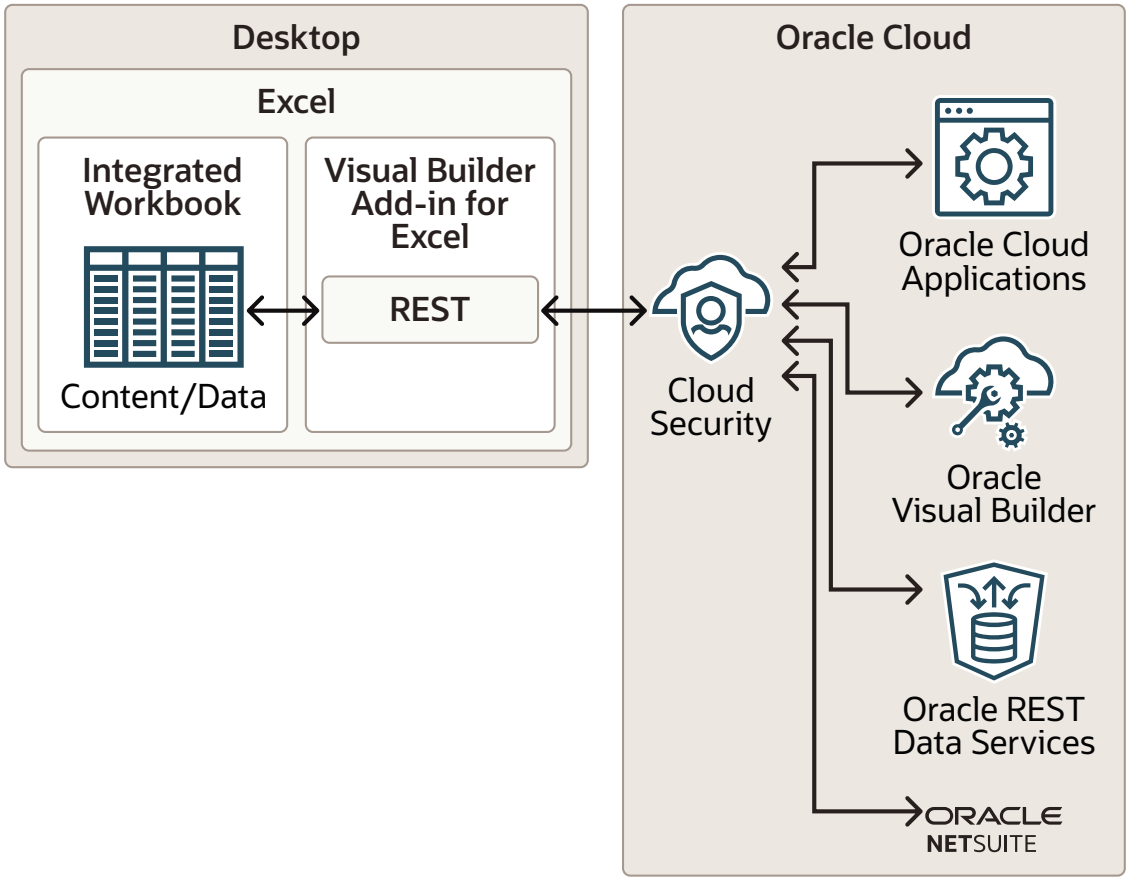
New and Changed Features

Area	Feature
List of values	Ability to write values to other fields (including the Structure Instance Number for key flexfield pickers) on selection. See Configure a List of Values to Populate Related Fields.
Dependent layouts	Support for specifying parents by combination of all ancestor column values. Add a Parent Column to Support Row Creation.
Expressions	Date-time value support added. See Dates and Times in Expressions.
	Improved handling of null values in expressions. See Handling Null Values in Expressions.
	Support for conditional operators added. See Operators in Expressions.
Validation rules	Support added for date-time fields. See About Field Validation Rules.
Default values	Support added for date-time fields.
Refresh Catalog	Added option to favor the existing configuration. See Refresh a Business Object Catalog.
Search Editor	Support for the "IN" operator. See The IN Operator in Searches.
UI changes	"Search Parameters" have been renamed to "Download Parameters"
	The "Query" tab in the Layout Designer has been renamed to "Download"
Basic Authentication	Simplified login window. See Basic Authentication.

2

Introduction to Oracle Visual Builder Add-in for Excel

Oracle Visual Builder Add-in for Excel integrates Excel spreadsheets with REST services to retrieve, analyze, and edit business data from the service. You download your data to an Excel spreadsheet, work with it, then upload your changes back to the service.



Key Concepts, Components, and Terms

Before you use Oracle Visual Builder Add-in for Excel, it helps to become familiar with these key concepts, components, and terms.

Term	Description
Integrated workbook	An Excel workbook configured to work with one or more business objects.

Term	Description
Service	A web service that provides access to application data. The add-in works with REST services. Throughout this book, REST service is implied whenever "service" is mentioned. "Web resources" is another equivalent term. See Representational state transfer for an overview of REST.
OpenAPI	The OpenAPI Specification defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. Refer to the OpenAPI specification here: https://swagger.io/specification/.
Business object	A resource - like a purchase order or invoice - that has fields to hold your application's data. A business object includes a collection path, an item path, a set of fields, and other properties.
Business object catalog	A set of one or more business objects with a common host and base path.
Path	A path identifies the specific resource in the host that a web client requests access to, for example: `/fscmRestApi/resources/11.13.18.05/draftPurchaseOrders`.
Collection path	A service path (endpoint) that can be used to fetch multiple rows of data from the business object and/or to perform operations on the collection.
Item path	A service path (endpoint) that can be used to fetch, or operate on, a single row from the business object.
Metadata path	A service path (endpoint) that can be used to fetch the service metadata for the business object.
Layout	A way to display a business object in an Excel worksheet. Each worksheet supports one of two layouts: Table or Form-over-Table. Layouts are created by workbook developers and are visible to business users in published workbooks.

REST Frameworks

For simplicity, the add-in guides use these short forms for the REST frameworks the add-in supports.

REST Framework	Short Form	More Information
NetSuite SuiteTalk REST web services	NetSuite	About NetSuite Services
Oracle ADF REST Resource	ADF REST	Oracle ADF REST Resource
Oracle REST Data Services	ORDS	Oracle REST Data Services
Visual Builder Business Objects	VBBO	Visual Builder Business Objects

Installation

To install the latest version of Oracle Visual Builder Add-in for Excel, download and run the installer.

You can find the latest version of the installer at the Downloads page on Oracle.com.

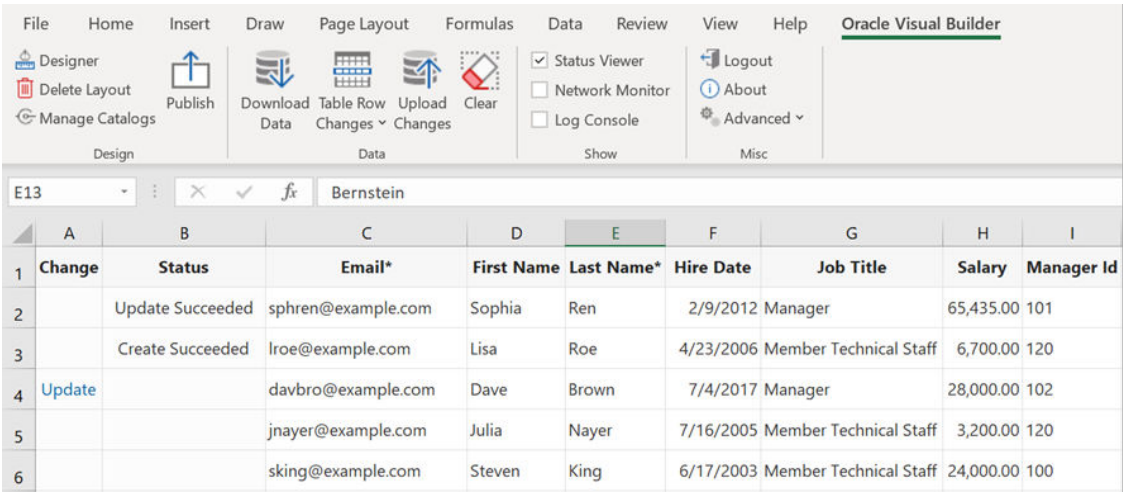
For more information, refer to Install Oracle Visual Builder Add-in for Excel.

Next Steps

After you install the add-in, a new **Oracle Visual Builder** ribbon tab appears in Microsoft Excel. As a workbook developer, you use the options in this ribbon tab to configure a worksheet to integrate with a service and download data to a data table that you create in the

worksheet. Once the data table is created and populated with data, you can review, modify, and create data, then upload your changes to the service.

This image shows a worksheet that is integrated with an REST service that manages employees:



	A	B	C	D	E	F	G	H	I
1	**Change**	**Status**	**Email***	**First Name**	**Last Name***	**Hire Date**	**Job Title**	**Salary**	**Manager Id**
2		Update Succeeded	sphren@example.com	Sophia	Ren	2/9/2012	Manager	65,435.00	101
3		Create Succeeded	lroe@example.com	Lisa	Roe	4/23/2006	Member Technical Staff	6,700.00	120
4	Update		davbro@example.com	Dave	Brown	7/4/2017	Manager	28,000.00	102
5			jnayer@example.com	Julia	Nayer	7/16/2005	Member Technical Staff	3,200.00	120
6			sking@example.com	Steven	King	6/17/2003	Member Technical Staff	24,000.00	100

Here are the high-level steps you'll need to follow to create a similar data table in a worksheet:

1. In the **Oracle Visual Builder** tab, click **Designer**.
2. Provide the URL of a service metadata document that complies with the OpenAPI specification.
3. Pick a business object.
4. Download data.

Review subsequent sections in this guide to understand available layout types and other add-in capabilities. For Excel specifications and limits, see Microsoft documentation.

3

Install Oracle Visual Builder Add-in for Excel

To install Oracle Visual Builder Add-in for Excel, download and run one of the two available installers. You can download these installers from the Oracle Downloads page.

There are two available installers for the add-in: a "Current User" installer and an "All Users" installer. Use the Current User installer to install the add-in on your local desktop for your own use. The All Users installer is intended for IT administrators.

The add-in runs in Excel on a Windows environment and requires some additional Microsoft components. Check out the Software Dependencies topic for details.

Installer Types

The Current User installer installs the add-in for the current user's Windows profile only. If there are other people using this computer, they will need to install the add-in for their own Windows profile.

When you use the All Users installer, the add-in is installed in the Programs Files folder and is available for all users on the target Windows machine.

Refer to this table for a full comparison of the two installers:

Comparison	Current User	All Users
File name	`vbafe-installer-current-user.msi`	`vbafe-installer-all-users.msi`
Installation Location	Current Windows user profile	Program Files
Target Audience	Business users	IT administrators
Windows Registry Entries	`HKEY_CURRENT_USER`	`HKEY_LOCAL_MACHINE`
Elevated Privileges	Not required	Required

Install Using the All Users Installer

To install Oracle Visual Builder Add-in for Excel for all users, download the All Users installer from the Oracle Downloads page and run the installer.

If any required software is missing, the installation terminates without installing the add-in. Refer to Software Dependencies for details including information on how to check for and install required components.

You must have elevated privileges for this installation.

The add-in includes designer tools for developing workbooks by default. If the user doesn't need these tools, you can disable them when you install the add-in. If these tools are needed later on, simply rerun the installer and enable them.

This installation is available to all users on the Windows machine. You do not need to install the add-in separately for each user profile.

1. Double-click the `vbafe-installer-all-users.msi` file that you downloaded previously to launch the installation wizard.

If you are not logged in with elevated privileges, you'll be prompted to provide credentials with elevated privileges.

2. To install the add-in without the available developer tools, click **Developer Options** and select **Disabled**.

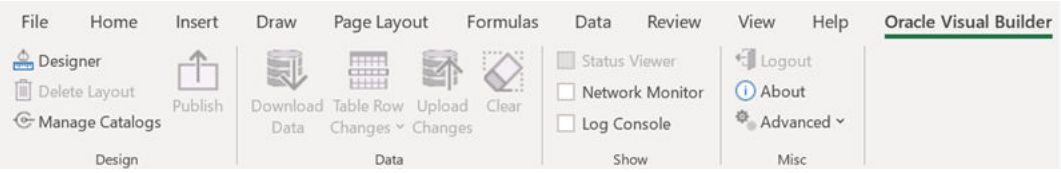
   If you need to enable these tools after initial installation, re-run the installer, choose the option to repair your installation, and select **Enabled**.

3. Click **Install** to install the add-in.

   When the installation is complete, click **Close**.

4. Start Excel and open a new workbook.

   A new Oracle Visual Builder ribbon tab appears, with commands to integrate the Excel spreadsheet with a REST service. If you disabled the design tools during installation, you won't see the Design options in the first group on the ribbon tab.



> **Note:**
>
> If you are unable to install the add-in, refer to Troubleshooting the Installation.

Install Using the Current User Installer

To install Oracle Visual Builder Add-in for Excel for the current user's profile, download the Current User installer (`vbafe-installer-current-user.msi`) from the Oracle Downloads page and run the installer.

If any required software is missing, the installation terminates without installing the add-in. Refer to Software Dependencies for details including information on how to check for and install required components.

The add-in includes designer tools for developing workbooks. These tools are included by default. If you are an application developer, make sure to install the designer tools with the add-in. If you need these tools but don't have them, simply rerun the installer and enable the designer tools.

This installation is specific to the current Windows user profile. If multiple users on a Windows machine need the add-in, consider using the All Users installer instead. See Install Using the All Users Installer.

> **Note:**
>
> Running both the All Users installer and the Current User installer on the same machine is not recommended. If you do install the add-in using both installers, Excel loads the add-in installed in the Program Files folder ("All Users") and not the version in the current user's Windows profile ("Current User"). If you decide to switch the installation type, the best practice is to uninstall the previous add-in installation type first to avoid confusion.

To install the add-in:

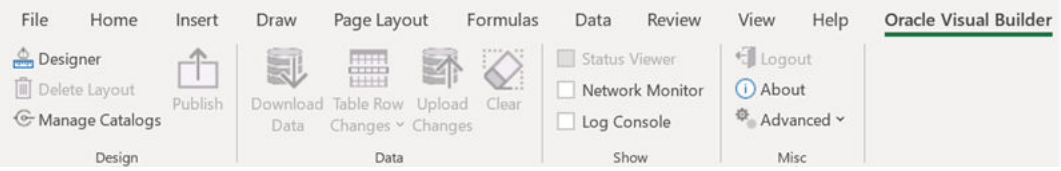
1. Sign in to the Windows user profile that will be using the add-in with Excel.
2. Quit Excel before you begin installation.
3. Double-click the `vbafe-installer-current-user.msi` file that you downloaded previously to launch the installation wizard.
4. To install the add-in without the available developer tools, click **Developer Options** and select **Disabled**.

> **Tip:**
>
> If you need these tools after initial installation, re-run the installer, choose the option to repair your installation, and select **Enabled**.

5. Click **Install** to install the add-in.

   When the installation is complete, click **Close**.
6. Start Excel and open a new workbook.

   A new Oracle Visual Builder ribbon tab appears with commands to integrate the Excel spreadsheet with a REST service. If you disabled the design tools during installation, you won't see the Design options in the first pane on the ribbon.



When you first run Excel after the current-user installation, you may be prompted to confirm the installation of the "Office customization". This prompt generally appears the first time for each profile and whenever the digital certificate (used to sign the add-in) is changed.

> **Note:**
>
> If you are unable to install the add-in, refer to Troubleshooting the Installation.

Run the Installer from the Command Line

Since both installers are standard Windows Installer Packages (MSI), you can run `msiexec` on the command line to install Oracle Visual Builder Add-in for Excel.

`msiexec` includes a number of options that let you control or customize your installation. See Standard Installer Command-Line Options on the Microsoft web site.

Examples

The following example shows a silent installation with logging enabled:

```
msiexec /package vbafe-installer-all-users.msi /quiet /log vbafe-install-log.txt
```

The following example shows a silent uninstall:

```
msiexec /uninstall vbafe-installer-all-users.msi /quiet
```

Unsupported Options

Not all `msiexec` options are relevant or supported for Add-in installation. Unsupported options include `/j` and `/a`.

Enable Designer Tools

The installers define a public property, “DESIGNER”, that can be used to enable or disable designer tools during command-line installation. To disable designer tools from the command-line, set `DESIGNER=0`. Refer to the `msiexec` documentation for details on how to set public properties during installation.

When performing an upgrade or repair, the installer uses the previously set value if this property is omitted.

Upgrade to the Latest Version

To take advantage of all the latest Oracle Visual Builder Add-in for Excel features, make sure you are running an up-to-date version of the add-in. To upgrade to a new version, simply download and run the installer.

When a new version is available, the add-in automatically prompts you to upgrade. You can also manually check for a new version from the **Advanced** menu. See Check for Updates.

For recommendations on when you should upgrade, check the Upgrade Policy.

You do not need to uninstall the previous version unless the installer instructs you to do so.

If you want to upgrade from one installation type to the other, uninstall the existing instance of the add-in first.

To determine which type of installation you have, run a diagnostic report and check the "Code Base" property under Properties. If the path is under the current user profile, the add-in was installed using the Current User installer. If the path is under Program Files, the add-in was installed using the All Users installer.

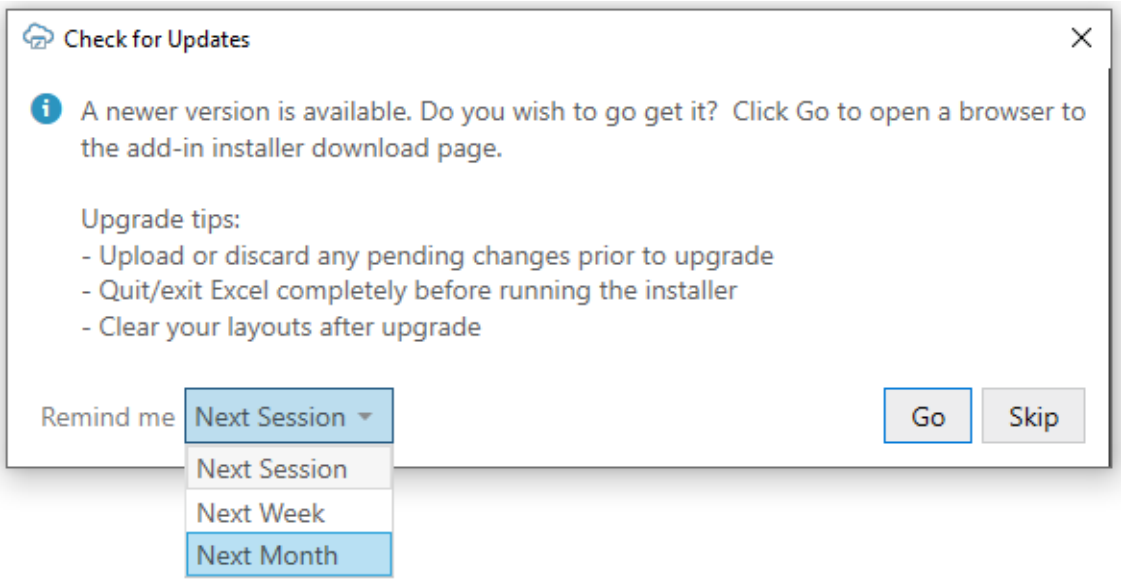
To run the diagnostic report, select **Diagnostic Report** from the **Advanced** menu on the Oracle Visual Builder ribbon.

To ensure a clean upgrade, follow these instructions when upgrading your installation.

1. Before you upgrade the add-in:
    a. Upload any pending changes using the current add-in version.
    b. Save changes in open workbooks, then close Excel.
2. Run the installer for the new version and follow the instructions in the wizard. The installer automatically replaces the previous version with the new version.
3. After you upgrade:
    a. Launch Excel to complete any final installation steps.
    b. Open your integrated workbook.
    c. Clear any layouts of old data and download data again as required.

Check for Updates

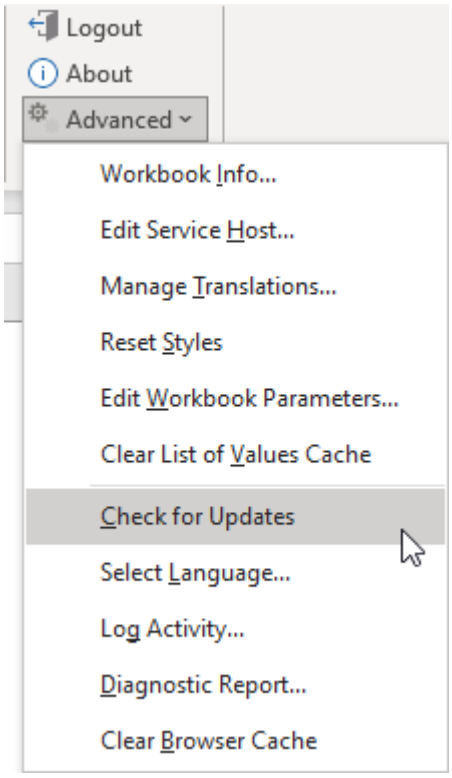
Oracle Visual Builder Add-in for Excel automatically checks for updates once per Excel session when the first integrated workbook is opened. If there is an update available, you are prompted to upgrade.You can also check for a newer version of the add-in using the **Check for Updates** command available from the **Advanced** menu. If there is a newer version, you can get the latest version of the add-in from the Downloads page.
If prompted to upgrade, you can choose to get the latest version right away or instead skip the upgrade.



If you choose to skip an upgrade, you'll be reminded again based on the duration you set in the **Remind me** list. If you select "Next Session", you'll be prompted to upgrade when you next launch Excel and open an integrated workbook.

To check for an update manually:

1. From the Advanced menu in your existing installation, select **Check for Updates**.



2. If a newer version is available, when prompted to open the downloads page in your browser, click **Go**.
3. Download the installer for the latest version, then install the update. Before you update, be sure to review best practices as described in the previous section.

Security

- The add-in does not send any information about the user or computer to Oracle (or anyone else) during the upgrade check.
- To perform the upgrade check, the add-in fetches this static resource: `https://www.oracle.com/a/ocom/docs/vbafe-info.json`.
- The resource request does include a `User-Agent` header whose value includes the current version of the Excel add-in (as with any request from the add-in).
- Default system credentials may be provided to proxy servers for authentication (as with any request from the add-in).

Upgrade Policy

When a new version of Oracle Visual Builder Add-in for Excel is released publicly, customers should upgrade at their earliest opportunity to take advantage of the latest fixes and improvements.

However, IT departments may need some time to evaluate and deploy a new version to business user desktops. The Oracle upgrade policy allows some flexibility to accommodate this process.

Customers may defer the add-in upgrade by up to six months from the time a new version becomes publicly available. After six months, the old version of the add-in is no longer supported and must be upgraded. So, if the latest version is released on May 15, customers

may defer upgrading from the previous version until November 15 at the very latest. After this date, upgrade is mandatory.

Limitations

This upgrade deferral policy is limited in several ways:

- Fixes for defects: To obtain a fix for an issue, it is necessary to install a new version of the add-in.
- Troubleshooting: When investigating a potential problem, one step in the troubleshooting process is to upgrade to the latest version. Deferral is not an option during the support process.
- New Workbooks: Frequently, new versions of the add-in offer new functionality. If your business users have integrated workbooks that leverage new functionality available only in the latest version of the add-in, then add-in upgrade becomes mandatory since the new functionality is not available in the previous version of the add-in.

Uninstall the Oracle Visual Builder Add-in for Excel

Uninstall either version of the add-in (Current-User or All-Users) from the Windows Settings app.

To uninstall the add-in:

1. If you're uninstalling the current user version, sign in to the Windows user profile where the add-in is installed.
2. From the Start Menu, select **Settings**, then **Apps**.
3. From the Apps & Features page, select Oracle Visual Builder Add-in for Excel from the list of programs.

> **Tip:**
>
> Type "Oracle" in the search box to filter on Oracle applications.

4. Click **Uninstall** and follow the instructions.
5. If you have performed a current user installation for multiple Windows user profiles and you want to uninstall them all, repeat these steps for each profile.

Software Dependencies

Oracle Visual Builder Add-in for Excel runs in Excel on a Windows environment and requires some Microsoft components to operate.

> **Note:**
>
> Oracle Visual Builder Add-in for Excel relies on a number of Microsoft technologies. These Microsoft technologies are subject to Microsoft's privacy policies and other Microsoft terms. By installing and using this add-in, you are agreeing to those policies and terms and this add-in's direct or indirect usage of these technologies. See the Microsoft Privacy Statement.

Components	Notes
Microsoft .NET Framework 4.8	The .NET Framework 4.8 is included since the May 2019 update of Windows 10. You can download NET Framework 4.8 here: https://dotnet.microsoft.com/en-us/download/dotnet-framework/net48.
Microsoft Visual Studio 2010 Tools for Office (VSTO) Runtime	The VSTO Runtime is included with most recent versions of Excel. You can download VSTO Runtime here: https://www.microsoft.com/en-us/download/details.aspx?id=105522.
Microsoft Edge WebView2	The add-in requires Microsoft Edge WebView2 for use as an embedded browser for displaying log-in pages. See The Embedded Browser.

The installer checks your system for required software and quits without installing the add-in if any required software is missing.

To install the WebView2 embedded browser, download one of the "evergreen" installers here and run it: Download Microsoft Edge WebView2.

> **Note:**
>
> For information about supported Excel and Windows versions, see Supported Platforms.

Supported Platforms

Oracle Visual Builder Add-in for Excel runs in Excel on Windows. This section provides details on which versions are supported. Review this topic carefully to make sure your configuration is supported.

Microsoft Excel

Refer to this table for supported versions of Excel.

> **Note:**
>
> Oracle recommends the 32-bit version of Excel whenever possible, as the 64-bit editions have been found to be less stable. Refer to the Microsoft article, 64-bit editions of Office 2013.

Version	Support Expires
Excel for Microsoft 365 (desktop installation only) *	Refer to Microsoft's Modern Lifecycle Policy.
Excel 2021	2026
Excel 2019	2025
Excel 2016	2025

* Please make sure that your version of Microsoft 365 Apps is still supported. If not, take steps to upgrade to a supported version. See Update history for Microsoft 365 Apps.

The following editions of Excel don't support VSTO/COM add-ins and are, therefore, *not supported*:

- Excel Online
- Excel for Microsoft 365 installed from the Microsoft Store

Microsoft 365 "Beta" and "Preview" update channels provide experimental versions of Excel. Oracle cannot provide support for the add-in when used with software from these channels.

Microsoft Windows

The add-in is supported on Windows 10 and Windows 11.

> **WARNING:**
>
> Windows 10 will reach end of support on October 14, 2025.

> **Note:**
>
> Please ensure that your version (or "feature update") of Windows is still in service according to the Microsoft policy. Oracle cannot provide support for versions beyond Microsoft's end of servicing date. See Windows 10 Lifecycle Policy.

The add-in isn't supported on:

- Windows server editions
- MacOS, iOS, Linux, or any other operating system

Virtual Desktop Infrastructure

The Visual Builder Add-in for Excel was designed and tested to work on regular Windows desktop and laptop computers. A number of vendors offer virtual desktop infrastructure (VDI) solutions to provide virtual machines that mimic a standard Windows desktop environment. Some VDI implementations reproduce the standard desktop very closely whereas others are significantly different. Refer to Desktop Virtualization for an overview of this technology.

Oracle does not test the add-in with VDI products formally. VDI products are not supported on an official basis.

Even though it may be possible to run the add-in on some virtual machines successfully, doing so is strictly at your own risk. If you run into problems, you'll have to reproduce the problem on a standard desktop computer before Oracle Support can assist you. If the problem appears to be a VDI issue, contact your VDI vendor for help.

Microsoft Application Virtualization

Microsoft Application Virtualization (App-V) is not supported in any version at this time.

Notes

- If a software application or version isn't listed here, it is not supported.

- Later versions of Excel and Windows are not automatically supported when they become available. Support can only be added through an enhancement request.
- Oracle doesn't support the add-in on unsupported software. If a vendor drops support for a given software version, Oracle support ends at the same time. This is true even if the software is listed here.
- Microsoft may not support all possible combinations of their software and operating systems listed here. If Microsoft doesn't support a given combination, Oracle doesn't either. If you're unsure if your versions of Excel and Windows are compatible, consult your software or operating system documentation.
- Support expiration dates for Windows and Excel are determined by Microsoft support policies. See Microsoft Lifecycle Policy.

Troubleshooting the Installation

If you experience issues when installing Oracle Visual Builder Add-in for Excel, follow the steps here to identify and resolve issues.

- Review the software dependencies to verify that all prerequisite software is installed.
- Make sure you're on a supported platform.
- Run the installer make sure you have the latest version of the add-in.
- If you don't have elevated (admin) privileges, use the current user installer.
- If you need additional support, generate an installation log and provide it to Oracle Support for review.

4

Create Layouts in an Excel Workbook

To integrate a workbook with a REST service, create a layout for a business object on a new worksheet. You can then download data for the business object to the layout and start working with it.

You create a layout by clicking **Designer** in the Oracle Visual Builder tab to launch the New Layout Setup wizard, as described in subsequent sections.

When you create a layout in a new workbook, you'll need to provide the service metadata for the REST service you want to use. This service metadata must comply with the OpenAPI specification. You can provide a URL to the service metadata at the REST service metadata endpoint or import a local service metadata file instead. You can find more information about service metadata URLs in Create a Table Layout in an Excel Workbook.

The service metadata that you provide helps generate a **business object catalog** for the workbook. A business object catalog is essentially a list of business objects. As a workbook developer, you can edit portions of the business catalog as desired, or use it as is to create layouts.

You can provide the service metadata when you create a layout. You can also provide the service metadata by clicking **Manage Catalogs** in the Oracle Visual Builder tab (see Manage Catalogs and Business Objects).

You create one layout per worksheet in your Excel workbook.

Layouts

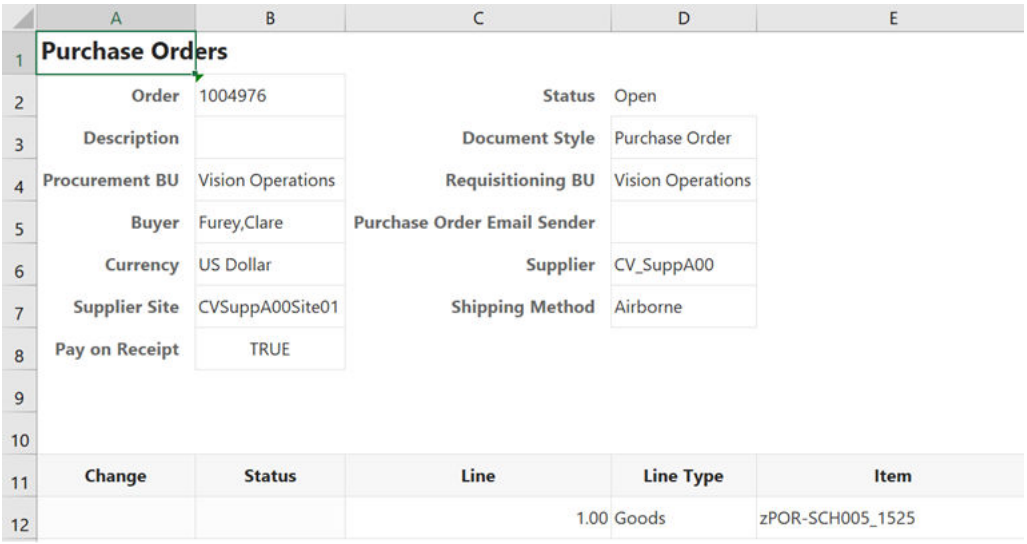
Oracle Visual Builder Add-in for Excel provides two different kinds of layouts you can use to work with data in an Excel worksheet: **Table** layouts and **Form-over-Table** layouts.

Use a Table layout to view and edit data from a REST service in a tabular format. Use a Form-over-Table layout when a parent-child relationship exists in the business objects used by your web application.

Here's an example of a worksheet showing employee data in a Table layout:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Change	Status	First Name	Last Name*	Email*	Hire Date	Job Title	Salary	Manager Id	Department*	Id*	Key
2			Sophie	Ren	sphren@example.com	2012-02-09	Manager	65,435.00	101	Research	2	
3			Dave	Brown	davbro@example.com	2017-07-04	Manager	28,000.00	102	Accounting	3	
4			Julia	Nayer	jnayer@example.com	2005-07-16	Member Technical Staff	3,200.00	120	Accounting	6	
5			Steven	King	sking@example.com	2003-06-17	Member Technical Staff	24,000.00	100	Accounting	7	

Here's an example showing purchase order and line data in a Form-over-Table Layout, where the parent object's data (Purchase Orders) is shown in the form and the child object's data (Lines) is shown in the table:



	A	B	C	D	E
1	**Purchase Orders**				
2	**Order**	1004976	**Status**	Open	
3	**Description**		**Document Style**	Purchase Order	
4	**Procurement BU**	Vision Operations	**Requisitioning BU**	Vision Operations	
5	**Buyer**	Furey,Clare	**Purchase Order Email Sender**		
6	**Currency**	US Dollar	**Supplier**	CV_SuppA00	
7	**Supplier Site**	CVSuppA00Site01	**Shipping Method**	Airborne	
8	**Pay on Receipt**	TRUE			
9					
10					
11	**Change**	**Status**	**Line**	**Line Type**	**Item**
12			1.00	Goods	zPOR-SCH005_1525

> **Note:**
>
> For multi-level business objects, there are additional options. See Use Multiple Layouts for Multi-level Business Objects.

Create a Table Layout in an Excel Workbook

Create a Table layout in the Excel worksheet when you want to view and edit data from a REST service in a tabular format.

When you create a Table layout, you'll be prompted to point to the service metadata. The service metadata document can be stored on your local drive or accessed remotely using a URL.

These sample URLs return the service metadata the add-in needs to create a business object catalog and then a table layout.

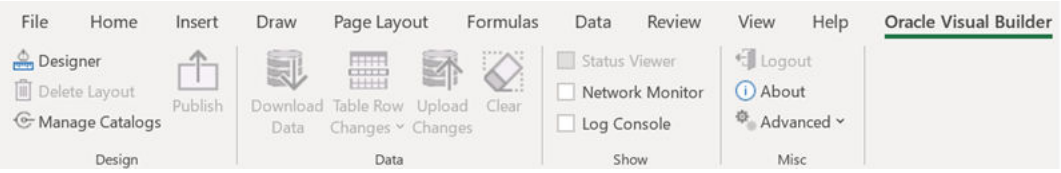
REST Framework	Sample URL
ADF REST (including many REST APIs for Oracle Cloud Applications)	`https://<host>/fscmRestApi/resources/11.13.18.05/purchaseRequisitions/`**`describe`**
VBBO	`https://<host>/ic/builder/design/<app>/1.0/resources/data/ExpenseReports/`**`describe`**
ORDS	`https://<host>/ords/<app>/`**`open-api-catalog`**`/employees/`
NetSuite	`https://<account>.suitetalk.api.netsuite.com/services/rest/record/v1/`**`metadata-catalog`**`?select=contact`

> **Note:**
>
> You cannot use a data endpoint to create a business object catalog. The metadata endpoint is required at this stage.

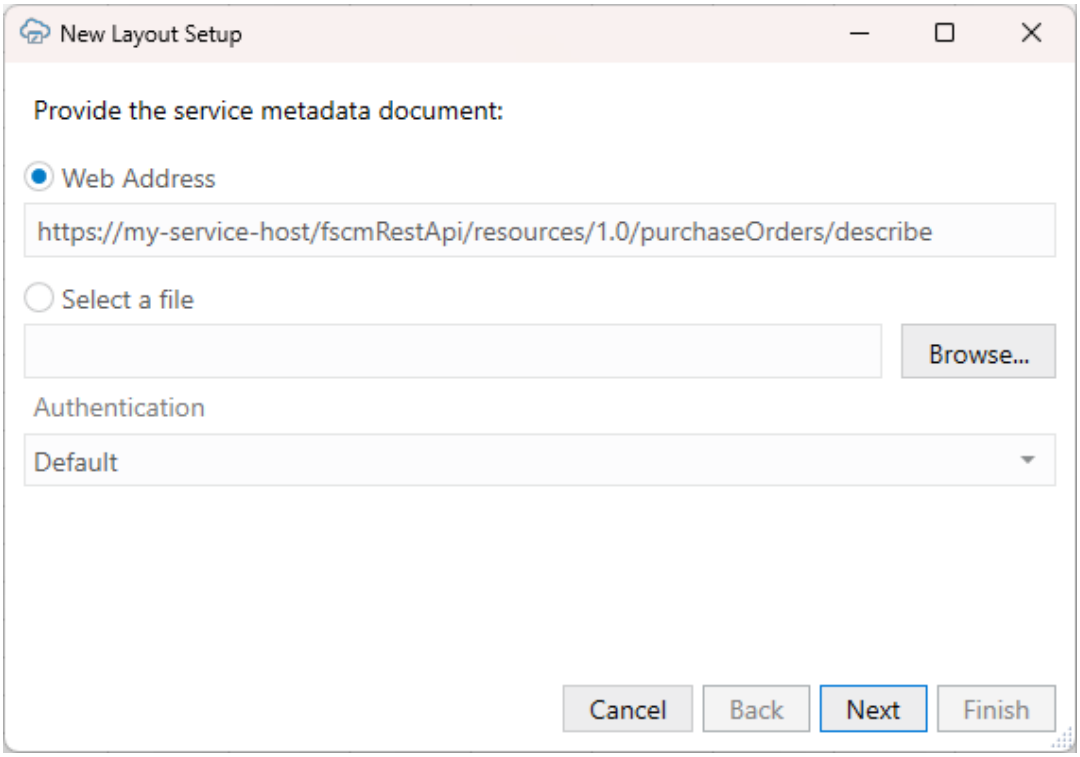
You'll also have the option to provide authentication details for accessing your REST service. Consult with your REST service owner for access requirements.

To create a Table layout:

1. Create a blank Excel workbook using the standard `.XLSX` file format or the macro-enabled `.XLSM` format. Other Excel formats (`.XLS` and so on) are not supported.
2. Click the cell where you want to locate the data table.
3. Open the **Oracle Visual Builder** tab from the Excel ribbon.



4. Click **Designer** to launch the **New Layout Setup** wizard.



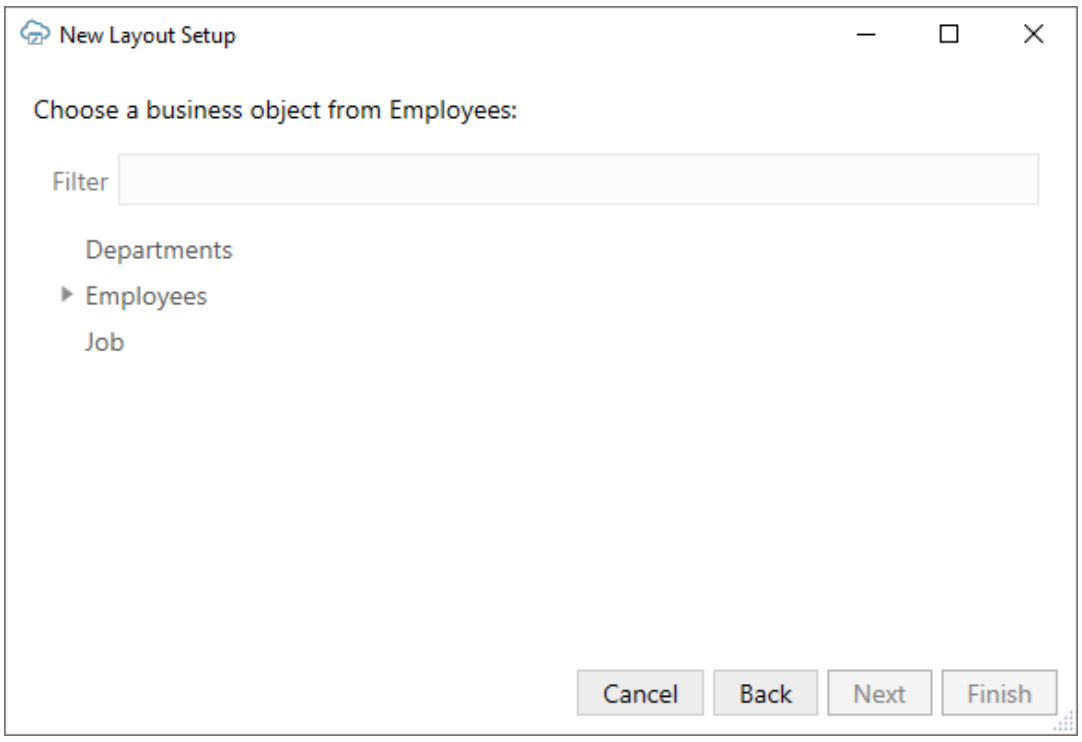
5. From the first screen, provide the service metadata document using one of these options:
   - **Web Address** option (the default) if you access the service metadata from a URL. Note that you can't provide a data URL (a URL that returns data) as the starting point to creating a layout.

- **Select a file** option if the service metadata document is a local file on your computer.

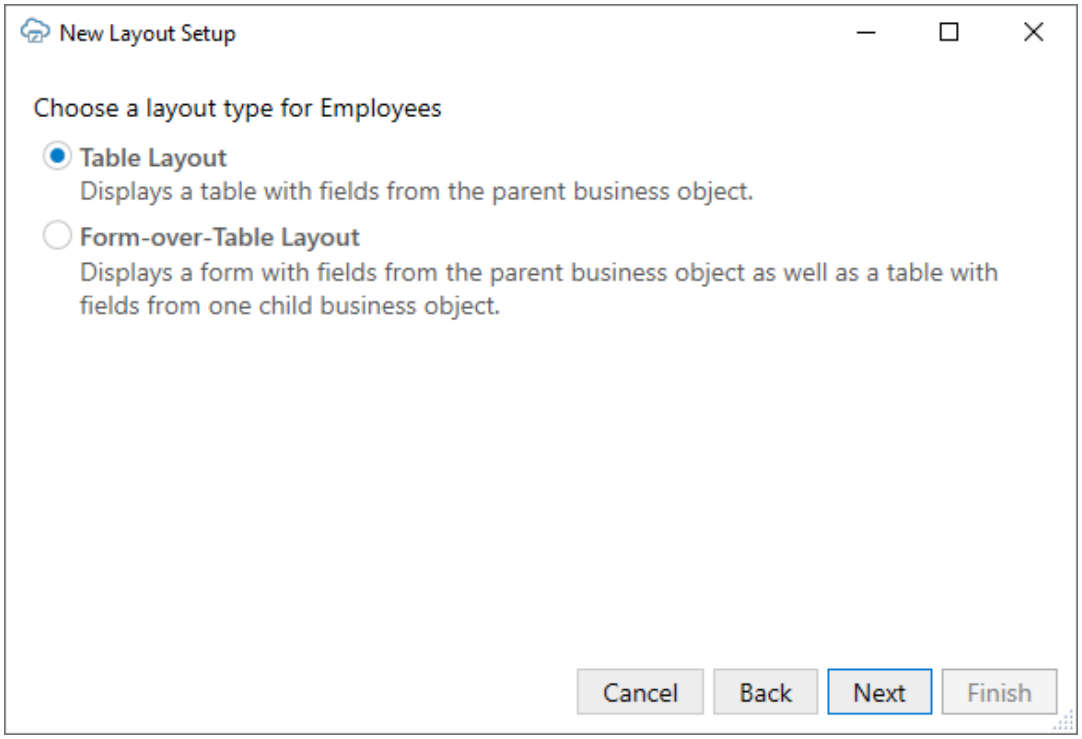
6. Select the authentication method for your service from the **Authentication** list and click **Next**.
   See Authentication Options for more information.
7. If you selected OAuth 2.0 Authorization Code, enter the required properties and click **Next**. Required fields are outlined in red. Refer to OAuth 2.0 Authorization Code Flow with PKCE for descriptions of the fields.
8. If the service includes five or more business objects, select the business objects you want to include in the catalog, then click **Next**.
   The wizard displays details of the newly-created catalog, such as the catalog name, service host and base path, and number of business objects.
9. Review the new catalog details.

> **Note:**
>
> If there are any errors in the service metadata document, click **Save Report** to save the report to your local drive. Share this report with the service owner.

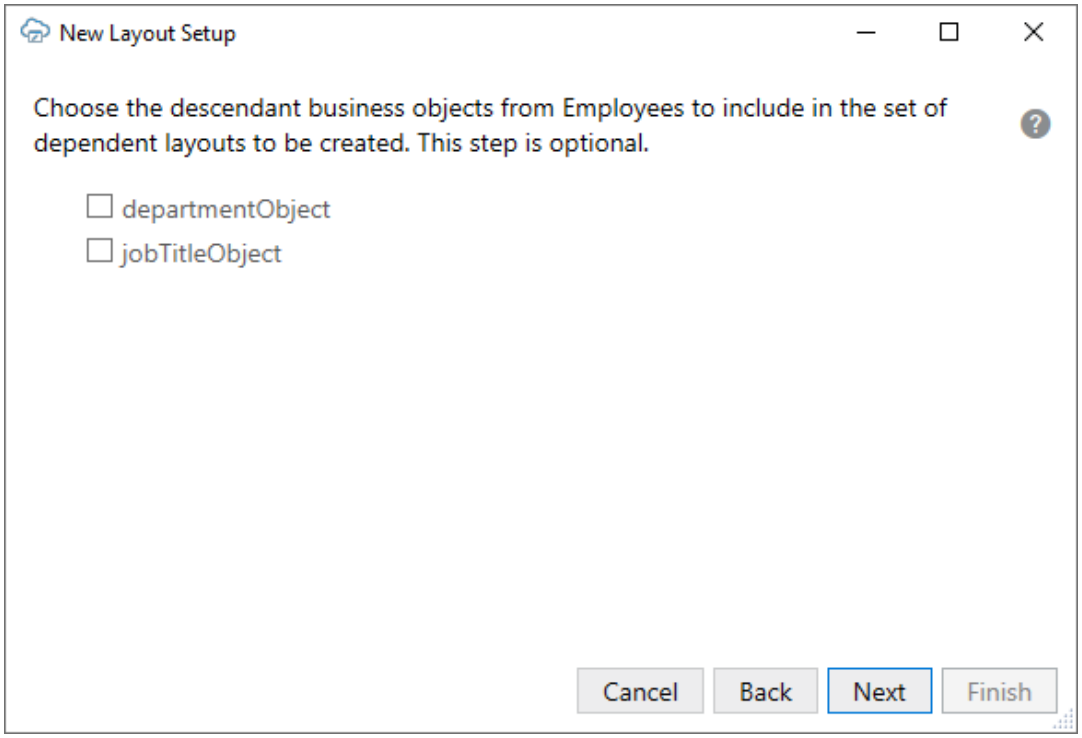
10. Click **Next** to proceed.
11. Select a business object and click **Next**.



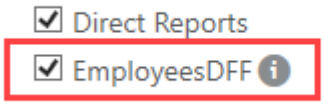
12. Select **Table Layout** and click **Next**.



The next screen prompts you to select child business objects if you are creating a set of dependent layouts. See Create a Set of Dependent Layouts.

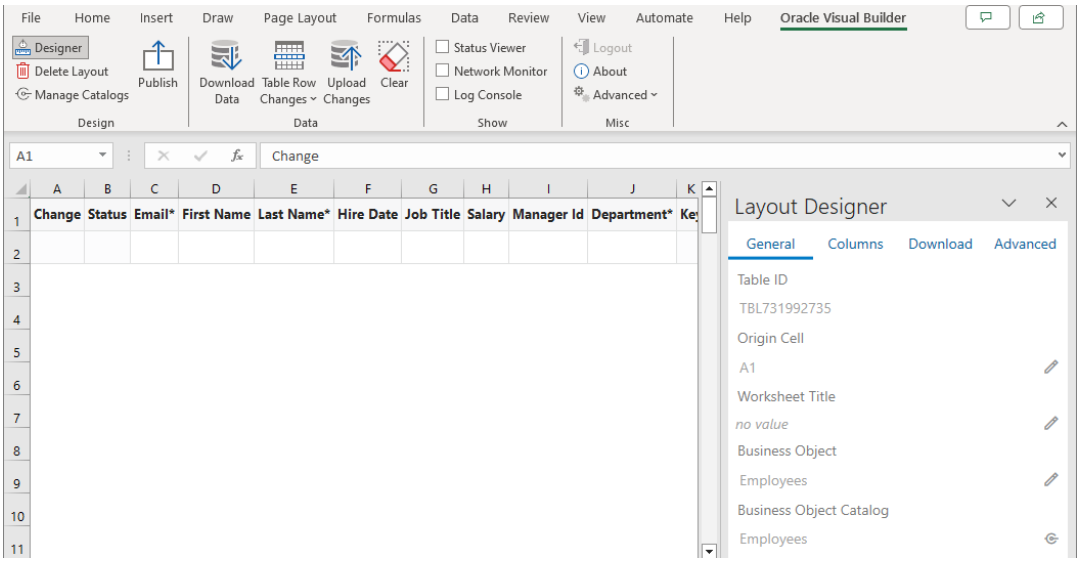


If there are any available descriptive flexfields, these are displayed in the list and are indicated by the Information icon ( ).



If you select this business object, the descriptive flexfields are added as fields to the respective parent layout. See Create a Layout Using Descriptive Flexfields.

13. For a standalone Table layout, click **Next** without selecting any child business objects.
14. Review the Table layout details and then click **Finish**.
    The add-in creates a Table layout in the Excel workbook that includes column headers and a placeholder data row. The Layout Designer opens in the Excel Task Pane.



> **Note:**
>
> If the origin cell of the layout is in the first 10 rows, the header row is frozen so that you always see the column headers when you scroll up and down in the worksheet. If desired, you can unfreeze the header row from Excel's **View** tab.

Before you proceed to publishing your workbook, you may want to configure the workbook in various ways to make it easier for your business users to use. For example, you might consider:

- Configuring the workbook to limit the data that the add-in downloads. See Configure Search Options for Download.
- Adding, removing, or reordering columns in your layout. See Manage Fields in a Form or Table.
- Modifying a field associated with a column to, for example, show help text or display a list of input values for your business users. See Configure Business Object Fields and Configure a List of Values with a Business Object.

Before you publish, it's a good idea to perform various data operations, such as download, update, and upload, to test the workbook before you distribute it to users. For information on managing data, see View and Edit Data Using an Excel Workbook in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Create a Form-over-Table Layout in an Excel Workbook

You can create a Form-over-Table layout in an Excel worksheet when a parent-child relationship exists in the chosen service.

A Form-over-Table layout can only be created for the top-level business object in a business object hierarchy. Suppose you have a hierarchy with three levels: `purchaseOrders`, `lines`, and `schedules`. In this hierarchy, `purchaseOrders` is a collection of top-level purchase orders each with one or more lines for managing the details of each order. Each of these lines may include one or more schedules for tracking shipping details.

In this scenario, you can only create a Form-over-Table layout for the `purchaseOrders` and `lines` business objects. You can't use `lines` for the form and `schedules` for the table in a Form-over-Table layout. See Use Multiple Layouts for Multi-level Business Objects.

> **Note:**
>
> You can create a layout that references polymorphic business objects and includes descriptive flexfields. If the business object includes descriptive flexfields, they are appended with "DFF" (for example, "EmployeesDFF") and included in the list of descendant business objects. See Use Polymorphic Business Objects and Fields.

A parent-child relationship at the service level requires:

- A parent service path, for example, `fscmRestApi/resources/1.0/purchaseOrders`
- A child service path with a parameter, for example, `fscmRestApi/resources/1.0/purchaseOrders/{purchaseOrder-id}/child/lines`

In your workbook, both business objects must be declared in the same catalog. Continuing our example, `lines` must appear as a child of `purchaseOrders`. To allow for data retrieval, updates, creation, deletion, and action invocation with the parent and child business objects using this layout, the service must support the corresponding GET, PUT/PATCH, POST, and DELETE operations on these paths.

When you create a Form-over-Table layout, you may be prompted to point to the service metadata document. The service metadata document can be stored on your local drive or accessed remotely using a URL.

You'll also have the option to provide authentication details for accessing your REST service. Consult with your REST service owner for access requirements.

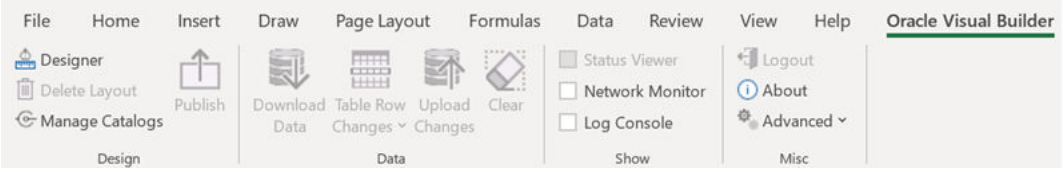
Before you begin, review these support topics for your REST service:

- For Oracle REST Data Services (ORDS), see the notes on ORDS support in Requirements for Dependent Layouts.
- For NetSuite SuiteTalk REST web services, see Configure a NetSuite Catalog for Parent-Child Business Objects.

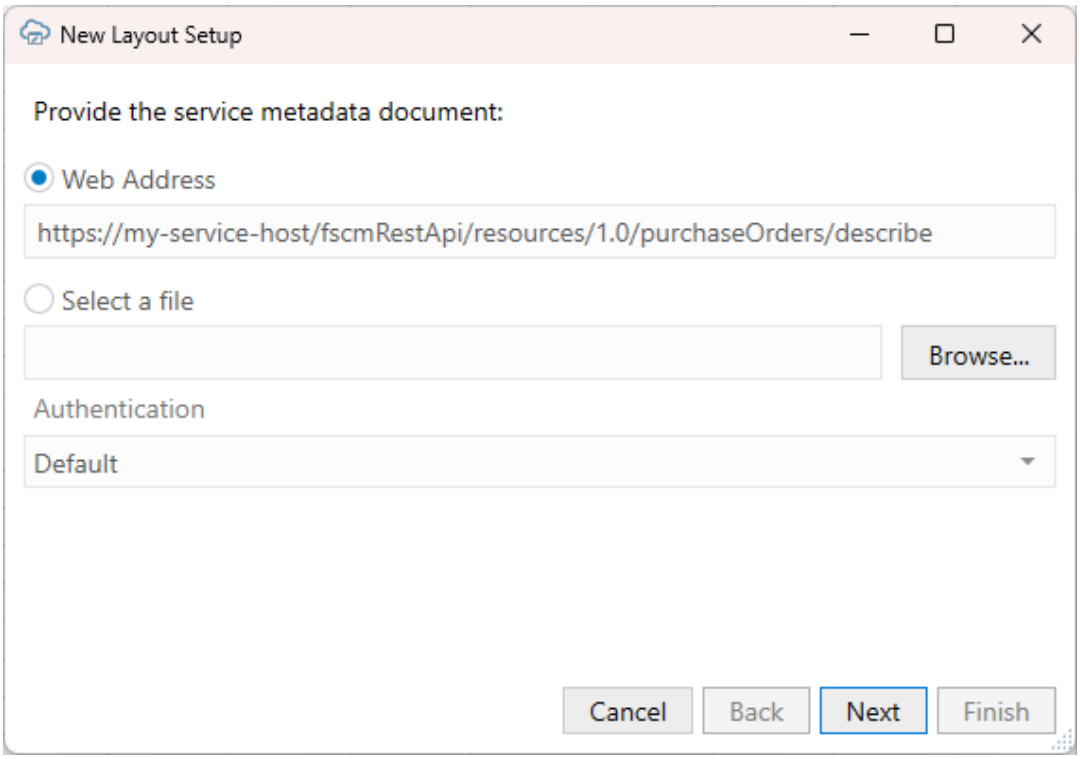
To create a Form-over-Table layout:

1. Create a blank Excel workbook using the standard `.XLSX` file format or the macro-enabled `.XLSM` format. Other Excel formats (`.XLS` and so on) are not supported.
2. Click the cell where you want to locate the form and table.

3. Open the **Oracle Visual Builder** tab from the Excel ribbon.



4. Click **Designer** to launch the **New Layout Setup** wizard.



5. From the first screen, provide the service metadata document using one of these options:
   - **Web Address** option (the default) if you access the service metadata from a URL. Note that you can't provide a data URL (a URL that returns data) as the starting point to creating a layout.
   - **Select a file** option if the service metadata document is a local file on your computer.

> **Tip:**
>
> If you are working with ADF REST services, the URL for the service metadata usually ends with `/describe`, for example, `https://my-service-host/fscmRestApi/resources/1.0/purchaseOrders/describe`.

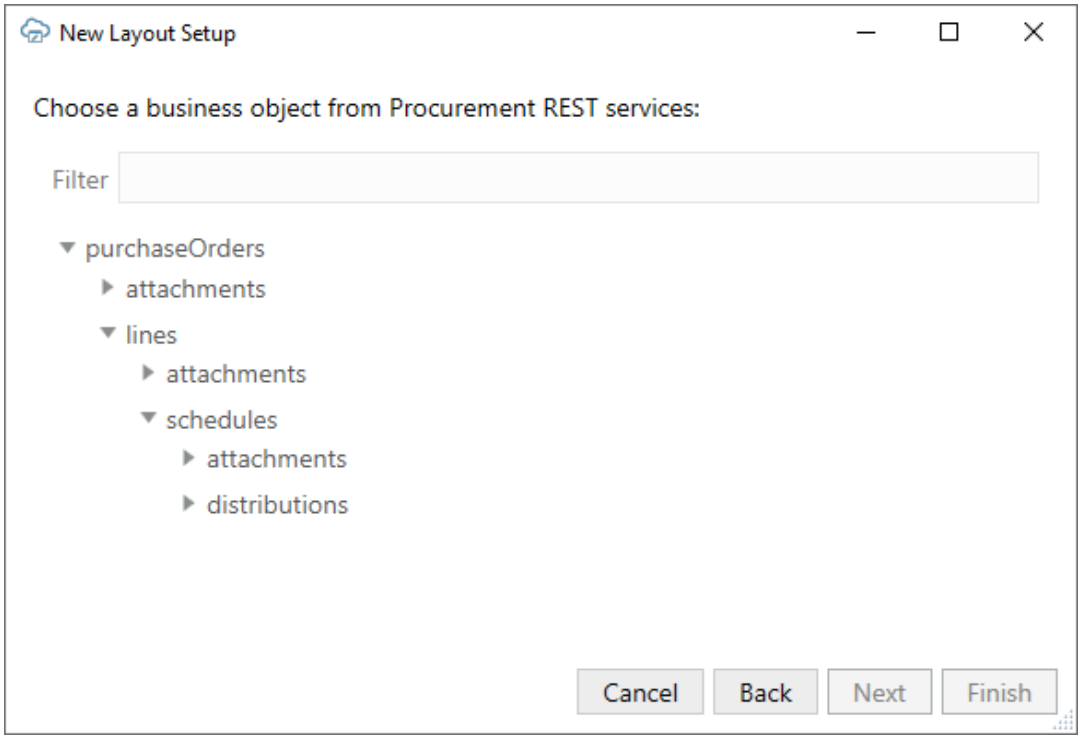
6. Select the authentication method for your service from the **Authentication** list and click **Next**.
   See Authentication Options for more information.
7. If you selected OAuth 2.0 Authorization Code, enter the required properties and click **Next**.

Required fields are outlined in red. Refer to OAuth 2.0 Authorization Code Flow with PKCE for descriptions of the fields.

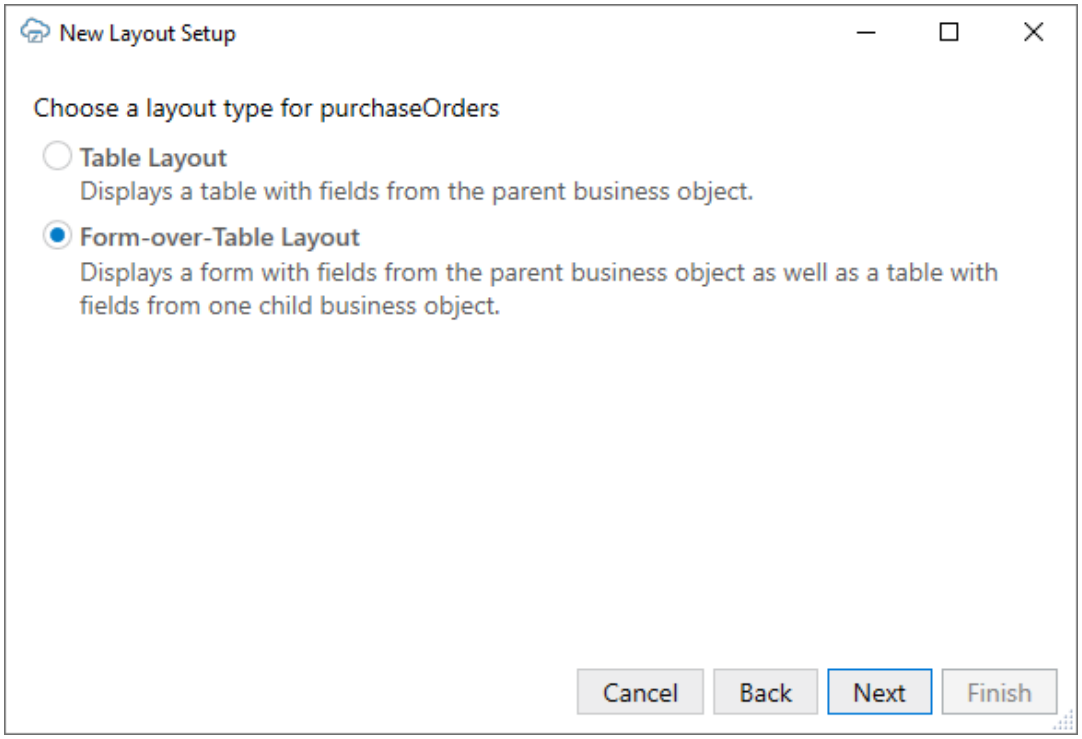
8. If the service includes five or more business objects, select the business objects you want to include in the catalog, then click **Next**.
   The wizard displays details of the newly-created catalog, such as the catalog name, service host and base path, and number of business objects.
9. Review the new catalog details.

> **Note:**
>
> If there are any errors in the service metadata document, click **Save Report** to save the report to your local drive. Share this report with the service owner.

10. Click **Next** to proceed.
11. Choose the top-level business object for the form (in this case, `purchaseOrders`), and click **Next**.

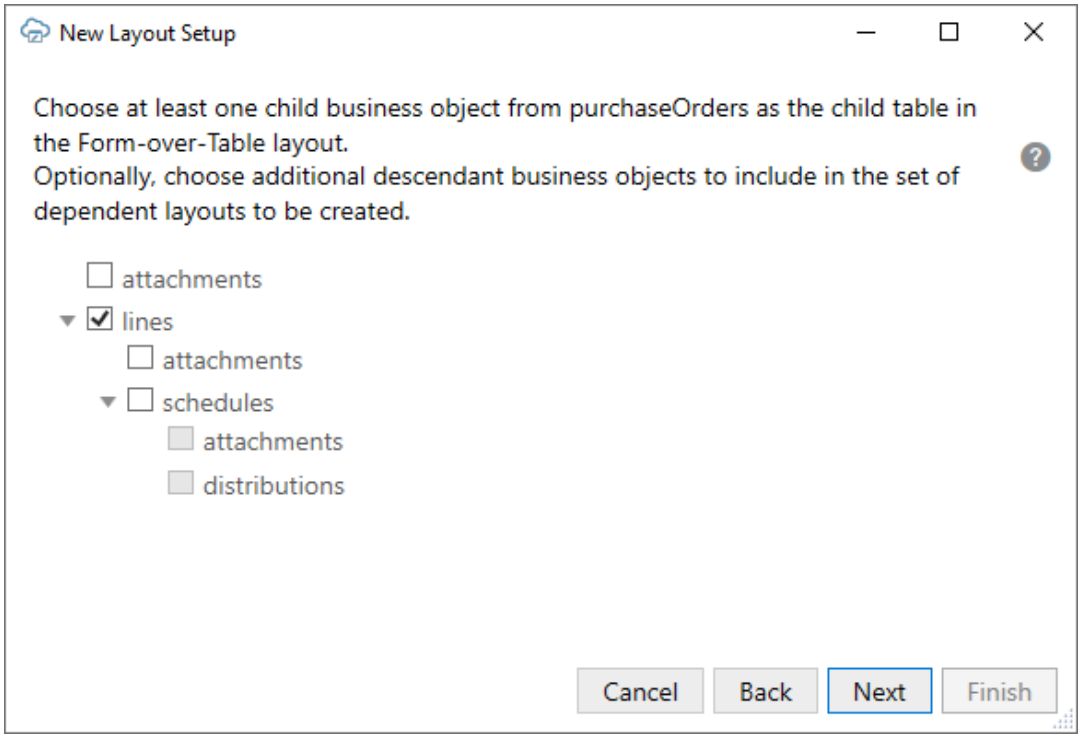


12. Choose **Form-over-Table Layout**, and click **Next**.

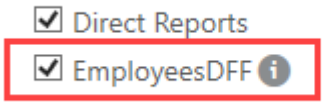


The Form-over-Table layout option is unavailable if you didn't select the top-level business object or there is no available child business object.

13. Choose a child business object for the table part of your Form-over-Table layout (in this case, `lines`), and click **Next**.



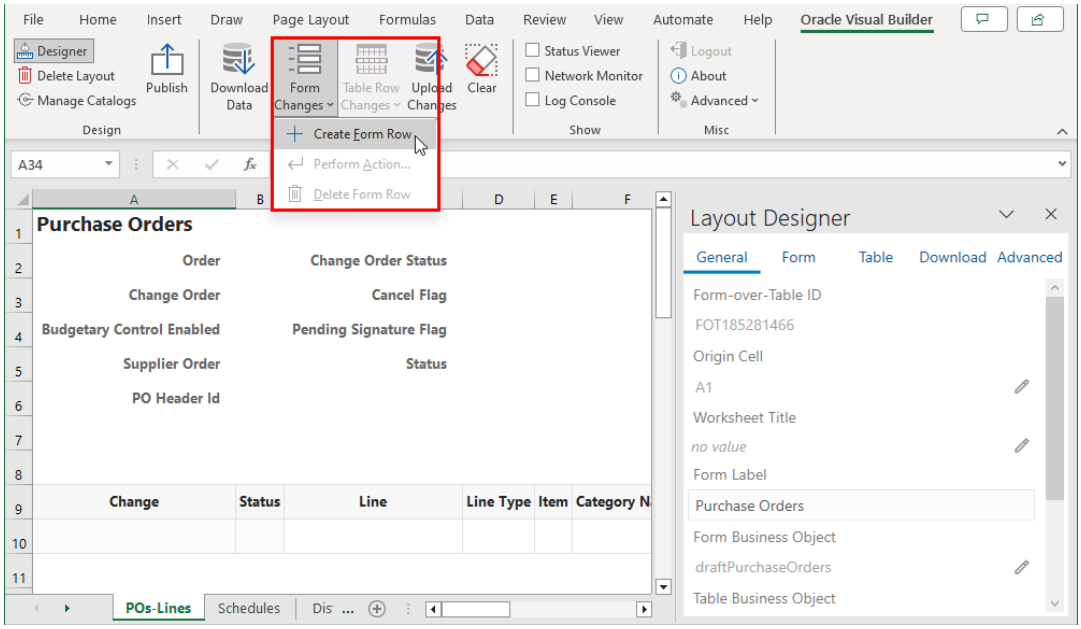
If there are any available descriptive flexfields, these are displayed in the list and display a "DFF" ending.



If you select this business object, the descriptive flexfield segments are added as fields to the parent layout. See Create a Layout Using Descriptive Flexfields.

> **Note:**
>
> You can select additional child business objects if you want to create a set of dependent layouts. If you select more than one, you'll be prompted to select the business object you want to show in the table of the root layout. See Create a Set of Dependent Layouts.

14. Confirm the details of your Form-over-Table layout, and click **Finish**.
The add-in creates a Form-over-Table in the Excel worksheet and opens the Layout Designer that you use to modify the newly-inserted form and table, as shown here:



If the form business object (in this case, `purchaseOrders`) supports a create action, a **Create Form Row** option appears in the Form Changes menu, as shown in the image. Use this option to create a new form row during your next upload (see Create a Parent Row in a Form-over-Table Layout in *Managing Data Using Oracle Visual Builder Add-in for Excel*).

Before you proceed to publishing your workbook, you may want to configure the workbook in various ways to make it easier for your business users to use. For example, you might consider:

- Configuring a search for the workbook to allow your business users to specify an item for the form, as described in Configure Search Options for Download. If you do not specify a

value for the Search field or Row Finder property, the add-in downloads the first parent item it encounters in the REST service to the form, and the child items, if any, to the table.

- Adding, removing, or reordering form fields or columns in your layout. See Manage Fields in a Form or Table.
- Modifying a field associated with a column to, for example, show help text or display a list of input values for your business users. See Configure Business Object Fields and Configure a List of Values with a Business Object.

Before you publish, it's a good idea to test the workbook before you publish and distribute it to users. For information on managing data in a Form-over-Table layout, see Manage Data in Form-over-Table Layouts in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Manage Fields in a Form or Table

When you create a layout, Oracle Visual Builder Add-in for Excel adds most of the business object fields to your form or table. If desired, you can add or remove fields or change the order they appear in the layout.

Besides adding business object and custom payload fields to a layout, you can also add fields from a parent or higher business object to a layout in a set of dependent layouts. These fields are referred to as "ancestor" fields. For example, you might want to add an ancestor column to a layout to help your business users track which child items are associated with which higher-level items. See Add Ancestor Columns to Provide Additional Context.

If there are any descriptive flexfields (DFFs) available your layout, these are also included in the list of available fields. For more information about DFFs, see Add Descriptive Flexfields to a Layout.

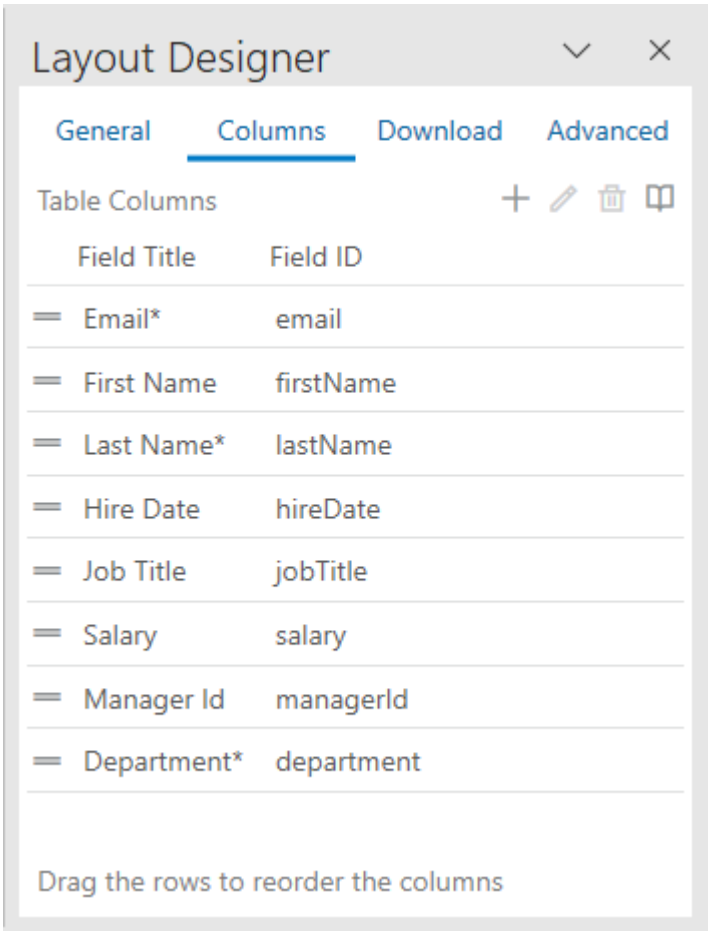
This task shows you how to add, remove, and reorder columns in a Table layout but the steps are the same for form fields and table columns in a Form-over-Table layout.

To manage the columns in a Table layout:

1. Select the layout, then click **Designer** from the Oracle Visual Builder tab to open the Layout Designer.
2. For a Table layout, click the **Columns** tab in the Layout Designer.

   For a Form-over-Table layout, click either the **Form** or **Table** tab.

   The tab displays the table columns in the order they appear in the layout.



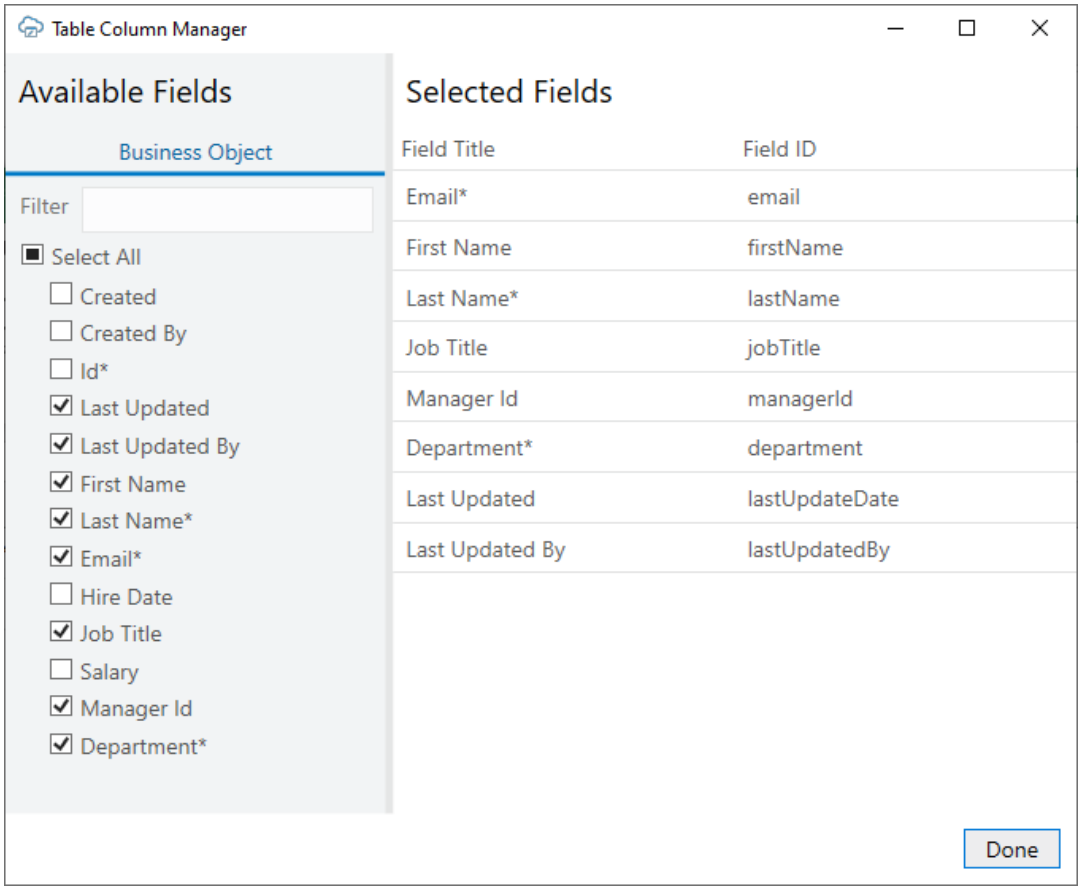
3. Perform one or more of these actions as required:
   - To change the order of the columns, drag a column to another location in the list.

     > **Tip:**
     >
     > You can also right-click a column and select an action (Move Up, Move Down, and so on) from the popup menu.

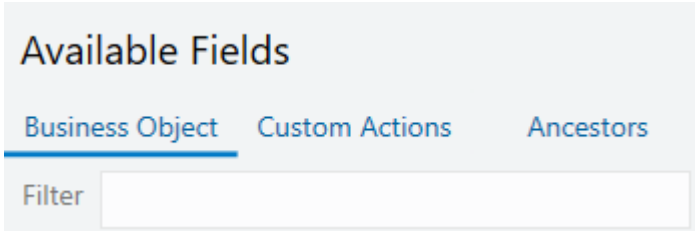
   - To delete a column, select a column and then click the Delete icon ().

     > **Tip:**
     >
     > You can use the Shift and Ctrl keys to select multiple columns for deletion.

   - To edit the field, double-click the column in the list to open the Business Object Field Editor.
4. If you want to add columns to your layout, click Manage Columns () to open the **Table Column Manager**.



If the business object supports custom actions or is a child object in a hierarchy of business objects, you'll see additional tabs (such as **Custom Actions** and **Ancestors**) in the **Available Fields** pane.



5. Click the appropriate tab for the type of field you want to add from the **Available Fields** pane.

   Available ancestor columns are grouped by business object. So ancestor columns available for a grandchild layout are grouped by parent and child.

6. In the **Selected Fields** pane, select the location where you want the columns to be added.

7. Select the columns you want to add from the **Available Fields** pane.

   The columns are inserted before the selected field.

   In this example, you may want to add **Hire Date** and **Salary** to the table before **Manager Id**. To do this, select **Manager Id** in the **Selected Fields** pane, then select the **Hire Date** and **Salary** check boxes in the **Available Fields** pane.

> **Note:**
>
> If you are adding a column that is referenced by a list of values in another column, make sure the added column is positioned before the field with the list of values. See Configure a Filter with a Dynamic Parameter for more information.

8. You can also perform one or more of these actions if required:
   - To add or remove all columns, select or deselect **Select All**.
   - To remove a column, deselect the column's check box.
   - To change the order in the layout, drag a column in the **Selected Fields** pane to another location in the list.

     > **Tip:**
     >
     > You can also move a column by right-clicking it, then selecting an action (Move Up, Move Down, and so on) from the popup menu.

   - To edit the field, double-click the column in the **Selected Fields** pane to open the Business Object Field Editor.
9. When you have made all your changes, click **Done**.

Create Layouts for Attachment Business Objects

Create a Table layout using an attachment business object that lets your business users upload and download attachments.

You can create a standalone Table layout, the Table layout of a Form-over-Table layout, or a Table layout in a set of dependent layouts based on an attachment business object. You cannot use the form in a Form-over-Table layout.

Service Requirements

Oracle Visual Builder Add-in for Excel supports attachments in integrated workbooks if the REST service has:

- An Attachment Record Business Object that contains the metadata for attachments, such as the attachment type, file name, and file size.
- The following fields in the Attachment Record Business Object:
  - **Type**: A string field representing the attachment type. Valid values for this field are `TEXT`, `FILE`, and `WEB_PAGE`. These values are case-sensitive.
  - **File Name**: A string field representing the attachment file name. This field is used by text and file type attachments.
  - **Url**: A string field representing the attachment URL. This field is used by web page type attachments.
- An Attachment Data Business Object that is a child of the attachment record business object and allows for the sending and receiving of attachments

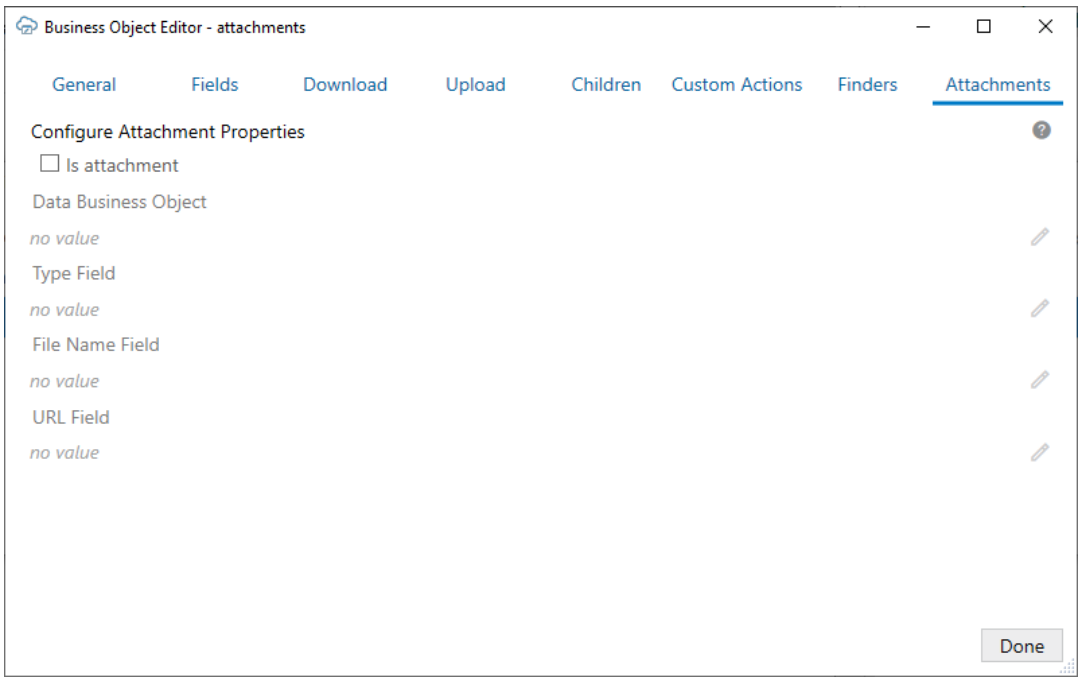
When you create a Table layout based on an attachment business object, the add-in can properly configure the layout as long as the attachment record business object includes:

- An Attachment Data Business Object as a child business object with a path ending in `/enclosure/FileContents`; and,
- Fields with field Ids of `DatatypeCode` (for the Type field), `FileName` (File Name), and `Url` (Url).

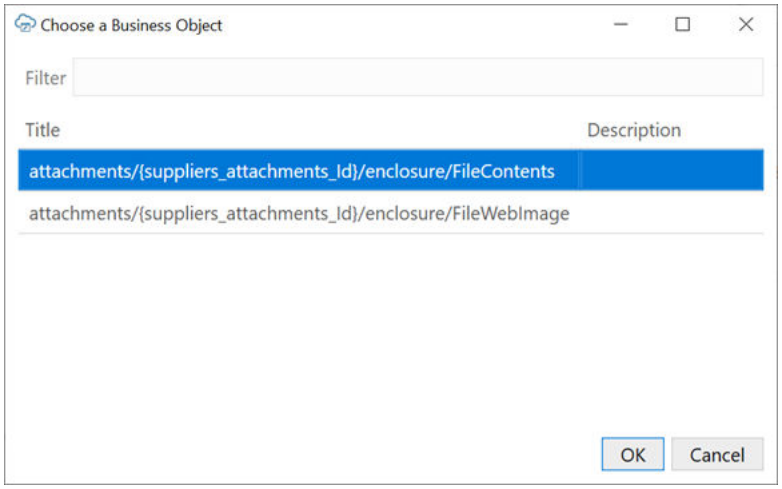
Configure an Attachment

If the workbook was created before version 2.8 of the add-in or the naming does not match, configure the attachment record business object manually from the Business Object Editor.

1. Open the Business Object Editor for an attachment business object and click the **Attachments** tab.



2. Select **Is attachment**.
3. Click the **Edit** icon (✎) next to **Data Business Object** to open the **Choose a Business Object** dialog.
   The dialog displays the paths for all child business objects for the Attachment Record Business Object.
4. Select the required child Attachment Data Business Object from the list, then click **OK**.

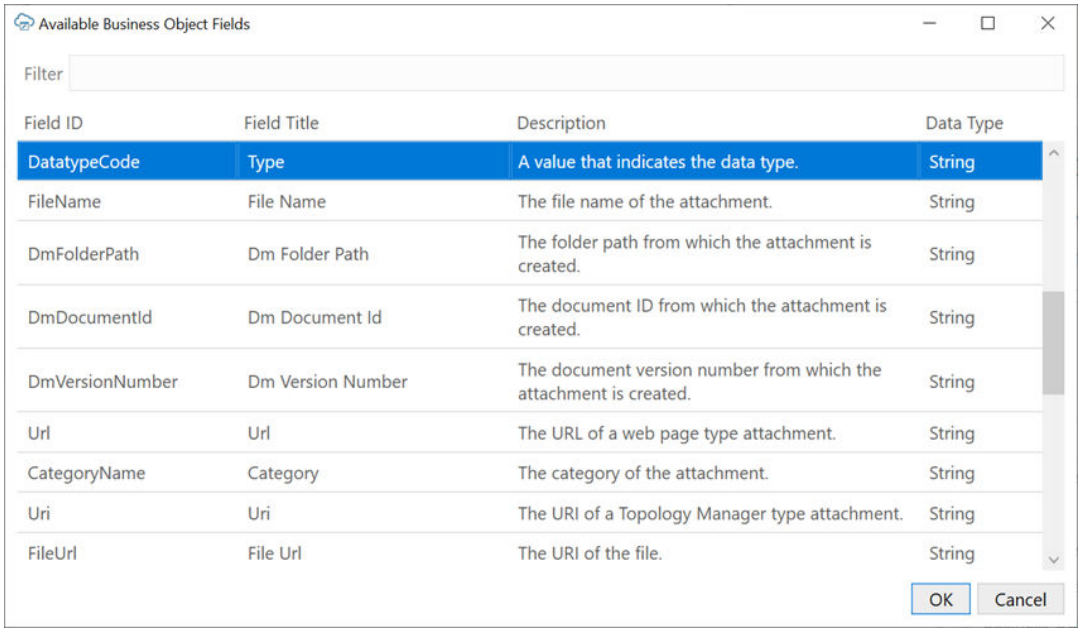


The child business object must support sending and receiving attachment data. This means that the child endpoint has a collection path that supports:

- GET to retrieve attachment data
- PUT to send attachment data
- Request content type of `application/octet-stream`

The path may be similar to this: `{parent attachment record item path}/enclosure/FileContents`.

5. Click the **Edit** icon (✎) next to **Type Field** to open the **Available Business Object Fields** dialog.
6. Select the field to represent the attachment type, then click **OK**.



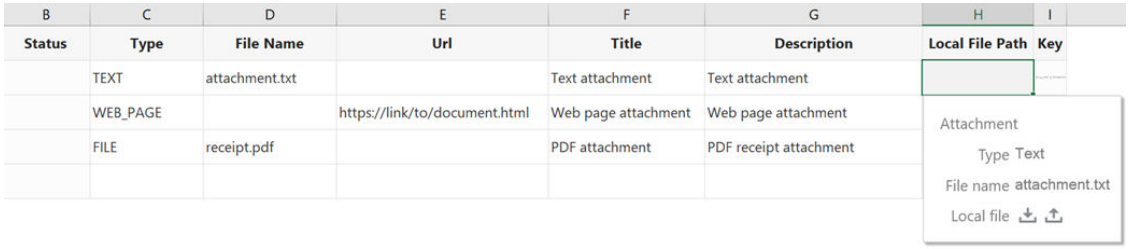
This field must be a string field with valid values of `TEXT` (text type attachment), `FILE` (file type attachment), and `WEB_PAGE` (web url-based attachment). Typically, the field ID for this field is `DatatypeCode`.

7. Use the available **Edit** icons to set the **File Name** and **URL** fields.

Field	Description
File Name Field	A string field representing the attachment file name. This field is used for text and file type attachments. Typically the field ID for this field is `FileName`.
URL Field	A string field representing the attachment URL. This field is used by web page type attachments. Typically the field ID for this field is `Url`.

8. When finished, close the open editors using the **Done** buttons.

If properly configured, the resulting Table layout includes a **Local File Path** column that keeps track of the location of local copies of attachments. Selecting cells in the table opens an attachment pop-up that can be used to interact with attachments. See Manage Attachments in *Managing Data Using Oracle Visual Builder Add-in for Excel*.



Known Limitations

- See **Service Requirements** in this topic for the limitations on services that support this feature.
- Binding an attachment record business object to the form portion of a Form-over-Table layout is not supported.
- The only supported attachment types are file, text, and web page. The specific supported values for the field are `TEXT`, `FILE`, and `WEB_PAGE`. These values are case sensitive. Unknown attachment types are treated as file type attachments.
- File download is not asynchronous and may make the UI appear frozen for large attachment files.
- It is not possible to change the position of the Local File Path column in the table. It always appears at the end of the table before the Key column.
- Manually editing the file path in the Local File Path column is not supported. The attachment pop-up should always be used to specify the local file location.
- For some services, create may fail for text-based attachments. Ensuring all required attachment metadata, such as the `Title`, is present and resubmitting the record generally resolves this issue.
- Attachment metadata records may have additional fields, such as `DmDocumentId`, `UploadedFileContentType`, that should generally be managed by the service and not altered by the business user. It is recommended that you mark these fields as read-only in the Business Object Editor if they aren't already or omit them from the Table layout.

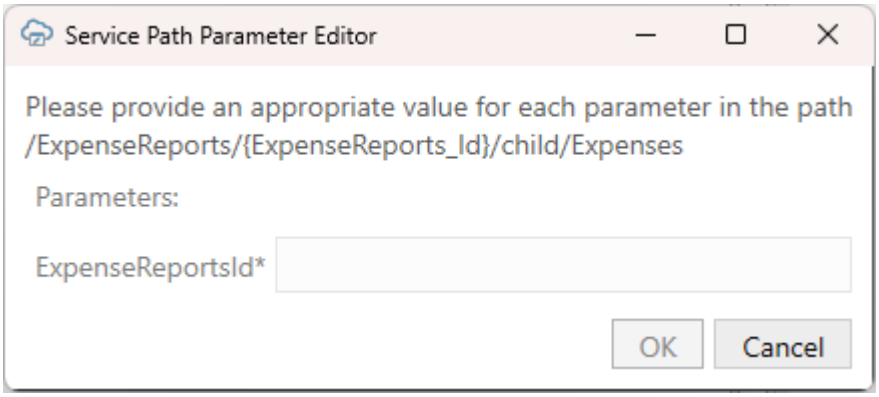
Work with Service Path Parameters in a Table Layout

Some service paths include path parameters. The add-in provides support for configuring a Table layout using a parameterized service path. It automatically extracts the path parameters and prompts the user to provide the corresponding values at download time.

To configure a Table layout with a parameterized service path, first provide an OpenAPI-compliant service metadata document. When prompted, choose a child business object or any parameterized path from the business object picker. Then complete the layout configuration.

> **Tip:**
>
> When working with ADF REST services, you should start with the web address to the parent business object metadata (and not the child address). For example, for a parameterized service path such as `/ExpenseReports/{ExpenseReports_Id}/child/Expenses/`, provide the address to the `ExpenseReports` metadata (not `Expenses`). ADF REST services cannot provide OpenAPI service metadata documents for parameterized service paths.

When a business user clicks **Download Data** from the Oracle Visual Builder ribbon, the add-in displays the Service Path Parameter Editor where the user can provide the required path parameter values that allow the download to proceed.



In this example, the layout uses a service path with an embedded parameter for the expense report Id (`{ExpenseReports_Id}`):

```
/ExpenseReports/{ExpenseReports_Id}/child/Expenses/
```

Note `{ExpenseReports_Id}` is in the middle of the service path.

If the user provides a value for `{ExpenseReports_Id}` in the given field, the add-in substitutes this value for the path parameter when creating the request to the service. If the user provides an expense report Id of `123456`, the add-in uses the following path:

```
/ExpenseReports/123456/child/Expenses/
```

A request using this service path returns all the expenses for expense report `123456`.

Path parameters of type string or integer are supported; other data types are not supported. For string-typed path parameters, values that users enter in the Service Path Parameter Editor

are used verbatim when the add-in constructs the request to the service. For integer-typed values, certain culture-specific formatting is removed (for example, commas for thousands separators, parentheses for negative). In all cases, the add-in applies URL-encoding to the value entered in the prompt.

The Service Path Parameter Editor does not validate the value(s) that users enter. The value(s) that users provide must be valid. If the path includes multiple embedded parameters, the Service Path Parameter Editor prompts the user to provide a value for each embedded parameter.

The add-in remembers the values provided at download time. These values are used again at upload time to construct the upload requests. If you upload without having done a previous download (for example, when exclusively creating new rows), you'll be prompted for the path parameter values at the beginning of the upload.

> **Note:**
>
> Since business users may not know the path parameter values at download time, consider using a Form-over-Table layout or a set of dependent layouts instead. See Create a Form-over-Table Layout in an Excel Workbook or Use Multiple Layouts for Multi-level Business Objects.

Use Polymorphic Business Objects and Fields

Oracle Visual Builder Add-in for Excel supports using polymorphic business objects from ADF REST services in Table and Form-over-Table layouts you create for an integrated workbook. Where and how a polymorphic business object can be used in a layout depends on its relationship with other business objects in the service.

If a polymorphic business object is a top-level business object or is a child business object with a "one-to-many" relationship with its parent, it can be used directly in a layout.

In this case, you create a layout for polymorphic business objects as you do for any other business objects. See Create a Table Layout in an Excel Workbook and Create a Form-over-Table Layout in an Excel Workbook.

If a child polymorphic business object is in a "one-to-one" relationship with its parent, it may be a descriptive flexfield or extensible flexfield. Descriptive flexfield segments can be added as fields to a layout bound to its parent business object during layout creation. See Create a Layout Using Descriptive Flexfields.

You can also add descriptive flexfields to an existing layout as described in Add Descriptive Flexfields to a Layout.

Unlike descriptive flexfields, extensible flexfields are used directly in a separate table layout. See Create a Layout for Extensible Flexfields.

To determine the relationship of a child business object to its parent, see Check the Cardinality of Child Polymorphic Business Objects.

About Polymorphic Business Objects

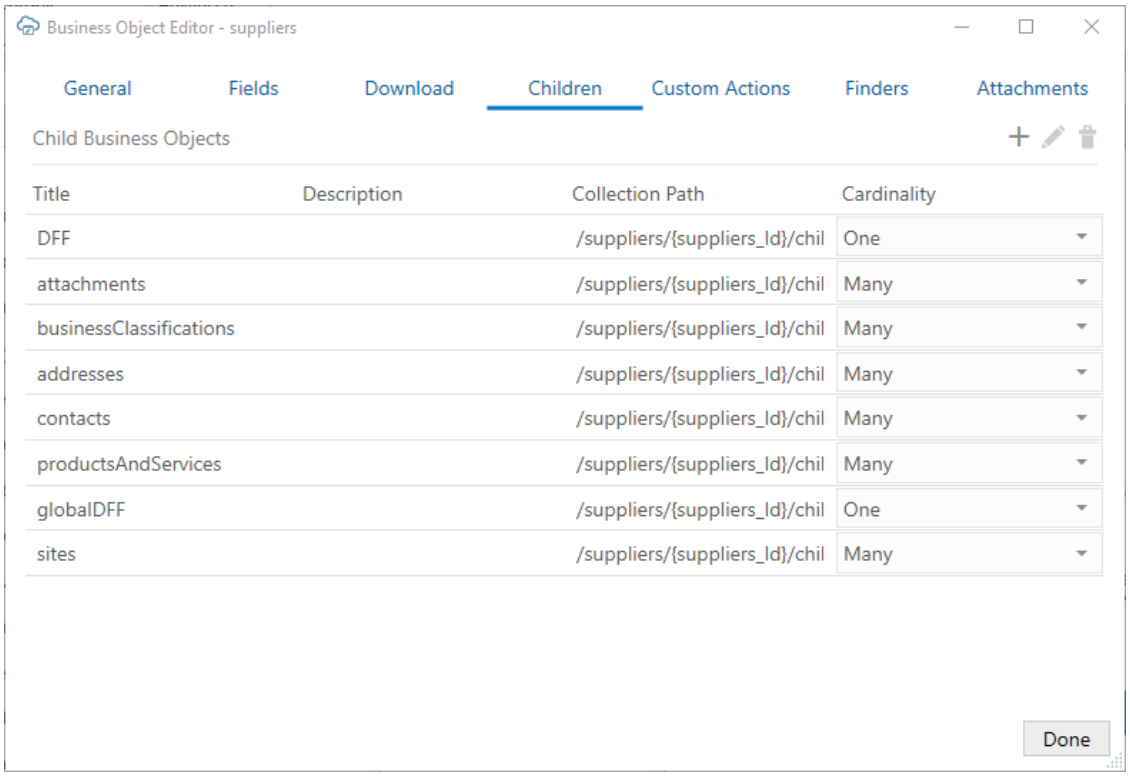
A polymorphic business object is a business object where the set of fields for a particular record differs based on the value of a **discriminator** field (also known as a **context segment**). In addition, a polymorphic business object includes a number of fields representing global or context-sensitive segments. Global segments are available for all values of the discriminator

field, while context-sensitive segments are dynamic based on the value of the discriminator field. Descriptive flexfields (DFFs) and extensible flexfields (EFFs) are types of polymorphic business objects.
For example, an "Employees" business object may include a child polymorphic "Regional Information" business object that defines region-specific information for employee records. The Regional Information business object contains a "Region" field that acts as a discriminator field. It also contains global segments for all records, such as "Site" and "Time Zone", as well as context-sensitive segments such as a "Postal Code/Zip Code" for employees in the North American region.

Check the Cardinality of Child Polymorphic Business Objects

You can determine if a child business object is in a "one-to-one" or "one-to-many" relationship with its parent by checking the "cardinality" setting for a child business object.

To check the cardinality setting for a child business object, open the Business Object Editor for the parent business object, then click the **Children** tab.



Cardinality options are "One", "Many", or "Unknown". Child business objects with a cardinality of "Many" (one-to-many relationship) can be used directly in a layout.

Those with a cardinality of "One" may be descriptive flexfields or extensible flexfields.

> **Note:**
>
> For workbooks created before Oracle Visual Builder Add-in for Excel version 3.2, the cardinality is set to "Unknown". This value behaves identically to a value of "One" for polymorphic business objects. The cardinality value does not impact the behavior of non-polymorphic business objects.

Create a Layout Using Descriptive Flexfields

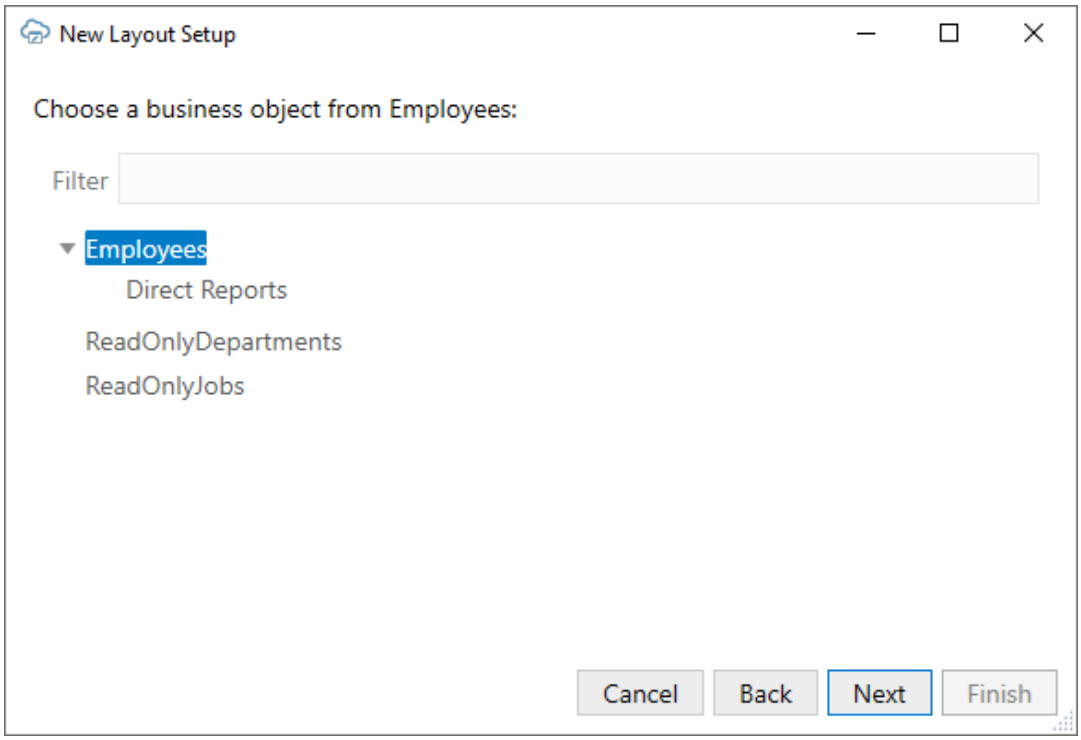
When you create a Table or Form-over-Table layout, you can add descriptive flexfields (DFF) from a child business object to the parent layout if the child is in a "one-to-one" relationship with the parent. DFFs are indicated in the New Layout Setup wizard by an Information icon ( ).

Top-level polymorphic business objects as well as child business objects with a "one-to-many" relationship with their parent can also be added to the layout as you would any other business object. See Create a Table Layout in an Excel Workbook and Create a Form-over-Table Layout in an Excel Workbook.

To determine the relationship of a child business object to its parent, check its Cardinality setting. See Check the Cardinality of Child Polymorphic Business Objects.

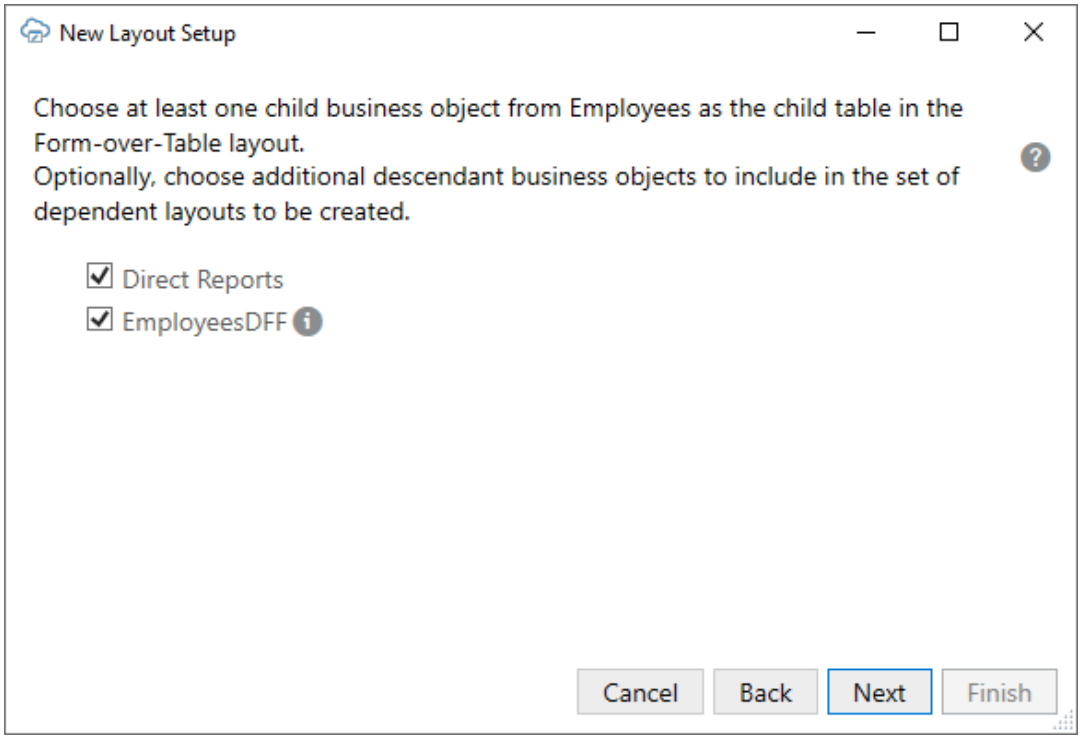
To create a layout with DFFs:

1. Create a new worksheet for your Table or Form-over-Table layout and click **Designer** to launch the **New Layout Setup** wizard.
2. Follow the instructions in the wizard, selecting the DFF's ancestor business object when prompted in the second screen.



   In this example, the Employees business object is selected for the parent layout.
3. When prompted, select either **Table Layout** or **Form-over-Table**.
4. When you reach the fourth screen of the wizard, you are prompted to select descendant business objects for your layout or layouts.

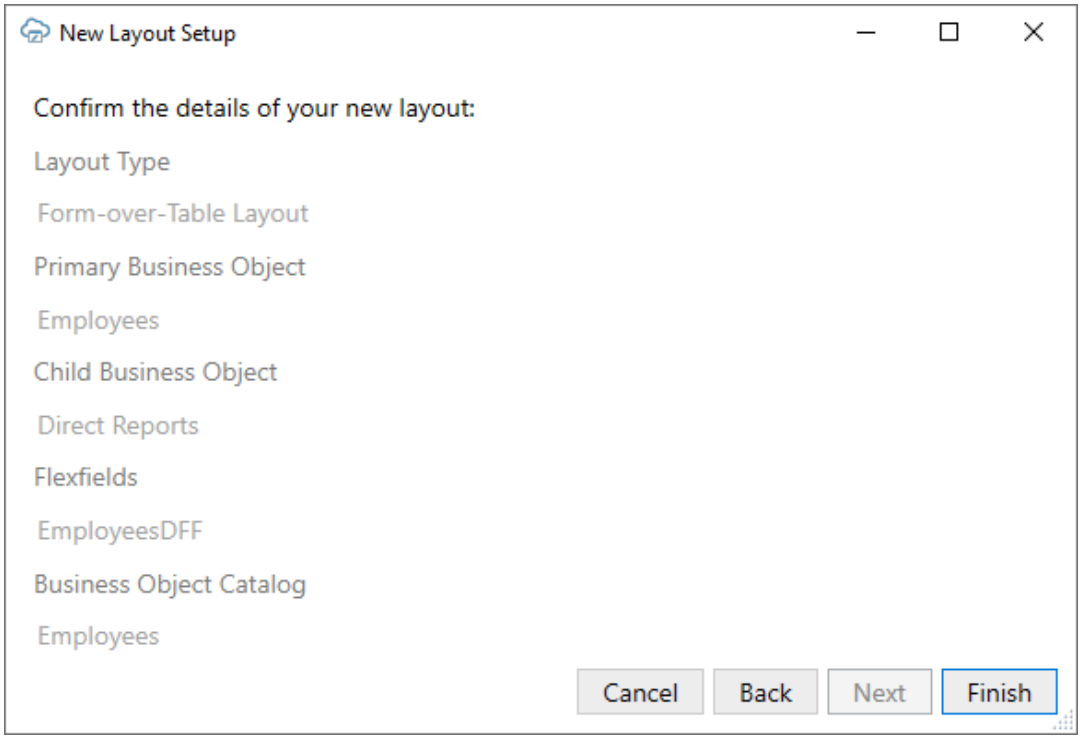
   In this example, both **Direct Reports** and **EmployeesDFF** are selected. EmployeesDFF represents a DFF that is in a one-to-one relationship with Employees, as indicated by the Information icon ( ).



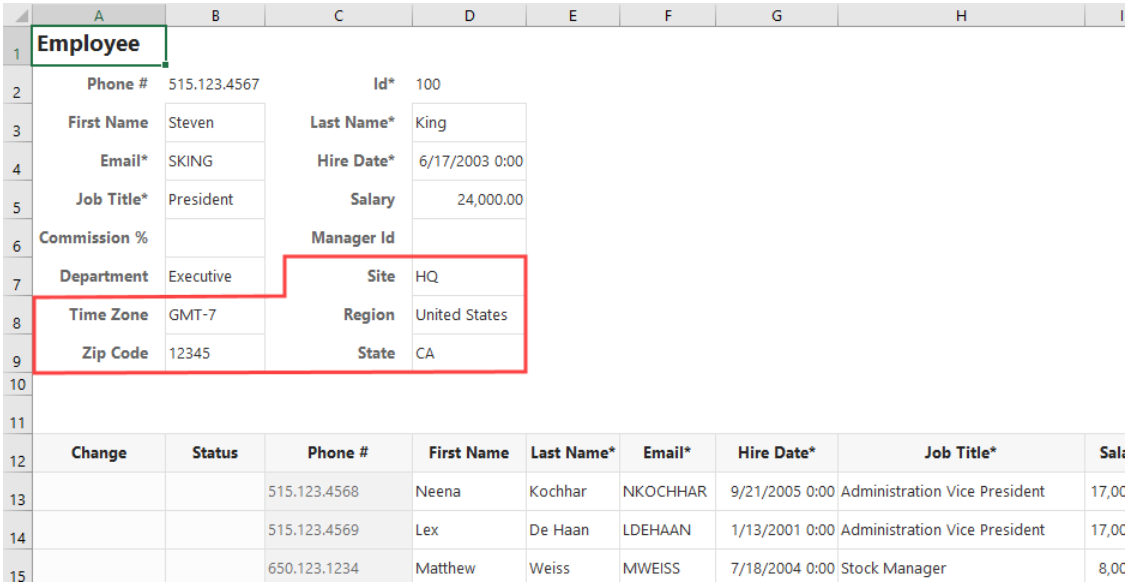
If you select EmployeesDFF, the add-in adds the flexfields to the parent layout, Employees.

If you choose to create a Form-over-Table layout, the add-in creates a layout with Employees in the form and Direct Reports in the table. If you instead choose Table layout, the add-in creates a dependent Table layout for Direct Reports.

5. When you reach the final screen in the wizard, review the details of the new layout, then click **Finish**.



In this example, the add-in creates a Form-over-Table layout with Employees in the form and Direct Reports in the table as shown in this image:



Employee							
Phone #	515.123.4567	**Id***	100				
First Name	Steven	**Last Name***	King				
Email*	SKING	**Hire Date***	6/17/2003 0:00				
Job Title*	President	**Salary**	24,000.00				
Commission %		**Manager Id**					
Department	Executive	**Site**	HQ				
Time Zone	GMT-7	**Region**	United States				
Zip Code	12345	**State**	CA				
Change	**Status**	**Phone #**	**First Name**	**Last Name***	**Email***	**Hire Date***	**Job Title***
		515.123.4568	Neena	Kochhar	NKOCHHAR	9/21/2005 0:00	Administration Vice President
		515.123.4569	Lex	De Haan	LDEHAAN	1/13/2001 0:00	Administration Vice President
		650.123.1234	Matthew	Weiss	MWEISS	7/18/2004 0:00	Stock Manager

The add-in also adds the Region flexfields from EmployeesDFF to the form (bordered in red). The flexfields include a "discriminator" field ("Region"), two global segments ("Site" and "Time Zone"), and two context-sensitive segments ("Zip Code" and "State"). The context-sensitive segments are dependent on the value of the Region field (in this case, "United States").

If the required flexfields do not appear in the layout by default, you can add them to your form or table from the Layout Designer. See Add Descriptive Flexfields to a Layout.

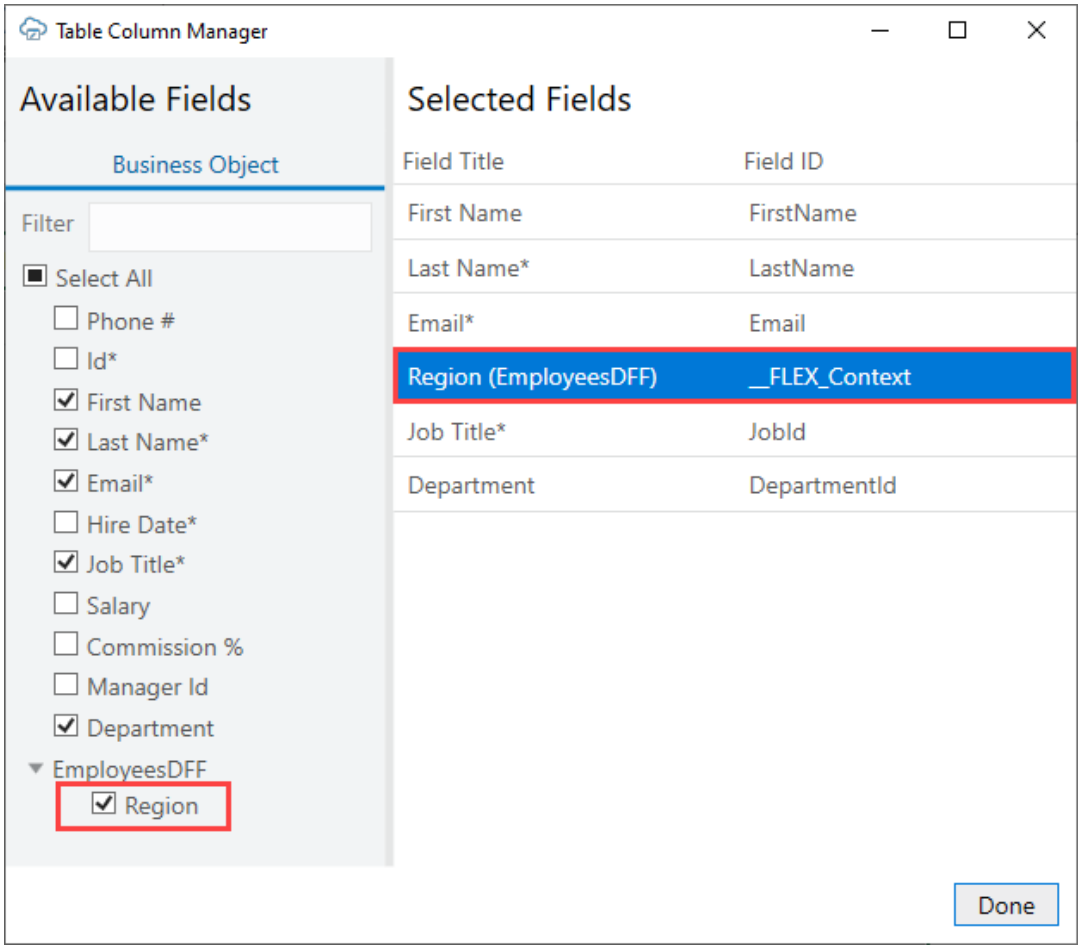
You can also show or hide context-sensitive segments based on the discriminator value. For example, you could choose to show the context-sensitive segments for Canada ("Postal Code" and "Province"), rather than the U.S. segments, ("Zip Code" and "State"). See Show or Hide Context-Sensitive Columns in a Table Layout.

Add Descriptive Flexfields to a Layout

You can add descriptive flexfields (DFF) to forms and tables using the Form Field Manager or Table Column Manager and place them anywhere in the form or table.Descriptive flexfields are a type of polymorphic business object that has a one-to-one relationship with its parent. See Overview of Descriptive Flexfields for more information on DFFs.

> **Note:**
>
> For polymorphic business objects in a one-to-many relationship, refer to Create a Layout Using Descriptive Flexfields instead.

1. Open the worksheet with the layout that you want to modify.
2. Click either the **Form** or **Columns** tab in the Layout Designer as needed.
3. Click the Manage Form Fields or Manage Columns button (+) to add your DFF. Available DFFs are identified by the title of the discriminator field (in this case, "Region") and appear at the bottom of the list of available fields. The field ID is the discriminator field ID ("__FLEX_Context").
4. Select the DFF from the Available Fields list.



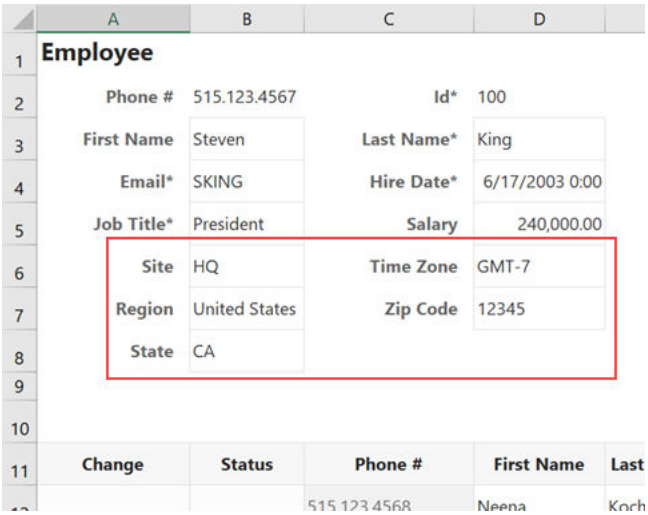
You can also change the order of fields in the form or table by dragging and dropping fields in the Selected Fields list.

5. Click **Done**.
   The associated segments appear in the layout in the following order: global segments, the discriminator, and context-sensitive segments.

Context-sensitive columns for all possible discriminator values are included in the table. However, only those cells in a row that are relevant to the value in the discriminator field are editable. All other context-sensitive cells are read-only (grayed out).

F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Job Title*	Phone #	Email*	Site	Time Zone	Region	Zip Code	State	Postal Code	Province	Hire Date*	Salary	Manager Id	Departme
President	515.123.4567	SKING	HQ	GMT-7	United States	12345	CA			6/17/2003 0:00	240,000.00		Executive
Administration VP	515.123.4568	NKOCHHAR	Burlington	GMT-4	United States	54321	MA			9/21/2005 0:00	170,000.00	100	Executive
Administration VP	515.123.4569	LDEHAAN	Toronto	GMT-4	Canada			A1A 1A1	Ontario	1/13/2001 0:00	172,000.00	100	Executive
Programmer	590.423.4567	AHUNOLD	Toronto	GMT-4	Canada			A1A 1A1	Ontario	1/3/2006 0:00	98,000.00	102	IT

In a form, the context-sensitive form fields relevant to the current discriminator value appear after the discriminator field. Context-sensitive fields that are not relevant are not included in the form. If you change the value in the discriminator field, the form automatically updates to include the relevant context-sensitive field for that value.



Create a Layout for Extensible Flexfields

You can create a standalone table layout that uses an extensible flexfield (EFF) business object as the primary business object. You can also choose a descendant EFF business object for a dependent layout.

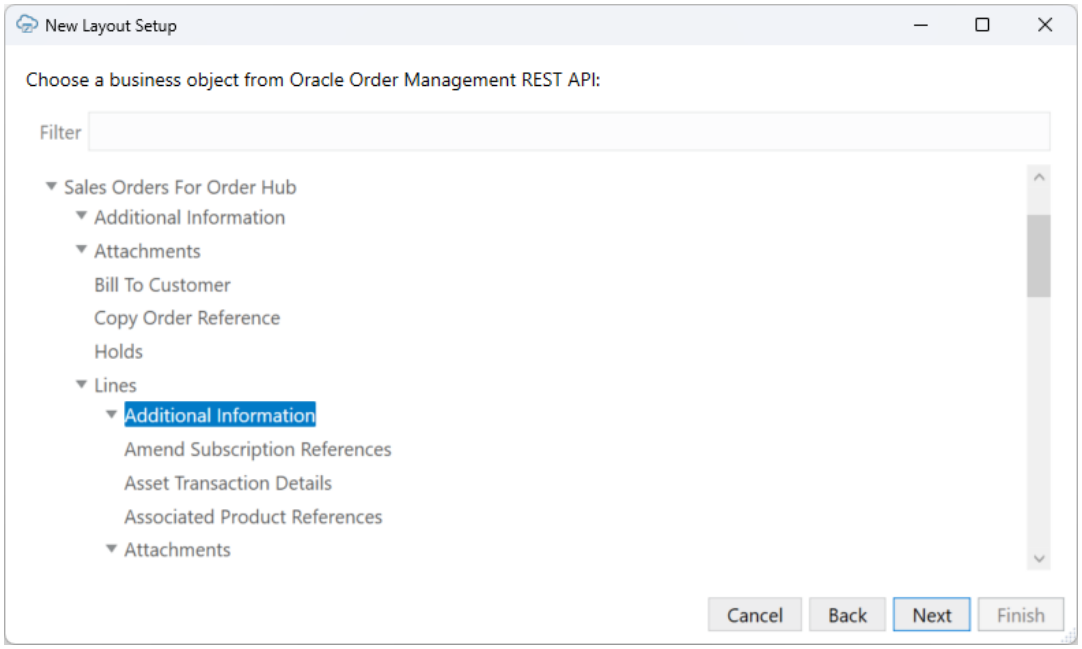
Like descriptive flexfields (DFF), EFF business objects have a one-to-one relationship with their parent business object. However, EFFs define additional extended attributes on child business objects. The extended attributes defined on any one-to-one children of the EFF business object are automatically expanded into the EFF layout. For more information about EFFs, see Overview of Extensible Flexfields.

Before you proceed, refer to Notes on Extensible Flexfields for information on feature interactions and limitations.

To create a layout with EFFs:

1. Create a new worksheet for your layout and click **Designer** to launch the **New Layout Setup** wizard.
2. Follow the instructions in the wizard, selecting a business object as the primary business object when prompted in the second screen..

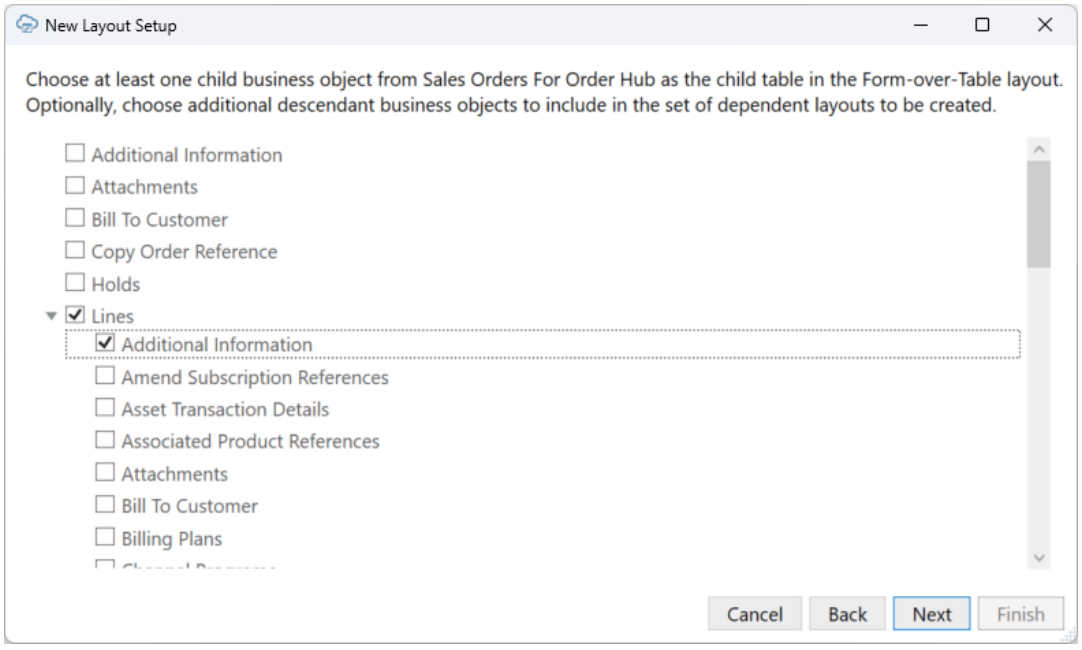
   If you want the layout to use an EFF business object, select it here. If you want to add the EFF to a dependent layout, you'll do this in the fourth screen.



In this example, the Additional Information business object is selected for the layout.

3. When prompted, select **Table Layout** or **Form-over-Table**.
4. When you reach the fourth screen of the wizard, you are prompted to select descendant business objects for your layout or layouts.

   If you want to add the EFF as a dependent layout, select it here. In this example, **Lines** as well as the EFF, **Additional Information**, are selected.



5. When you reach the final screen in the wizard, review the details of the new layout, then click **Finish**.

When you add an EFF to a layout, the EFF expands to include all global segments and the discriminator field defined on the base EFF business object as well as all the context-sensitive fields (extended attributes) defined on one-to-one children of the EFF business object.

Notes on Extensible Flexfields

Review this section for technical notes on extensible flexfields.

- Do not configure the expand and fields parameters for the layout in the Layout Designer. The EFF layout may not function as expected on download. EFF layouts use the expand parameter with a value of "all" to retrieve EFF records and child (extended attribute) records in a single request on download.
- Do not enable **Retrieve Descendant Rows in Single Payload** from the Advanced tab of the Layout Designer. EFF download is not compatible with this feature because of the layout's use of the expand parameter.
- An EFF business object must be bound to its own separate layout. It cannot be denormalized into its parent layout as descriptive flexfields (DFF) are.
- It is not possible to limit the EFF contexts (child business objects). All contexts will be denormalized in the EFF layout.
- The relationship between the EFF business object and the child extended attribute business objects must be one to one. One-to-many children will be ignored by the add-in.
- If your workbook was created using an earlier version of the add-in, refresh the metadata to retrieve the full set required to work with EFF business objects. See Refresh Polymorphic Business Object Metadata.

Show or Hide Context-Sensitive Columns in a Table Layout

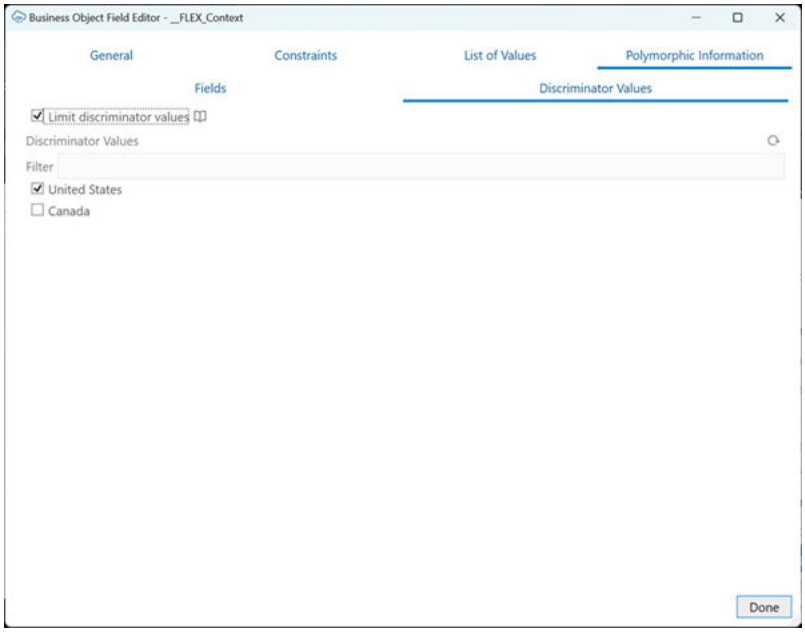
You can select which context-sensitive columns you want to display for a polymorphic business object, using the **Polymorphic Information** tab of the Business Object Field Editor.

All context-sensitive columns are shown by default. You may want to use this task to hide columns in a layout. For example, in the case of a Region polymorphic business object, you may choose to show regional information only for U.S. employees and hide it for all others.

To show or hide context-sensitive columns:

1. Open the worksheet with the layout that you want to modify.
2. From the Layout Designer, click the Edit icon () next to the Business Object field.
3. From the Business Object Editor, click the **Fields** tab and then select the business object's field.
4. Click the Edit icon () in the Business Object Editor to open the Business Object Field Editor.
5. From the **Polymorphic Information** tab, select **Limit discriminator values**.
6. To ensure you see all available discriminator values, click the Refresh icon () to fetch the latest polymorphic metadata from the service.

   This icon may be disabled if a refresh has already been performed during the current session.
7. From the Discriminator Values list, select the discriminator values for the context-sensitive segment columns that you want to display. For example, to show zip code and state columns in your Table layout, select the **United States** check box and deselect all others.



8. After you select or deselect discriminator values, close the open editors.

The layout displays the context-sensitive segment columns you selected.

F	G	H	I	J	K	L	M	N	O	P	Q
Job Title*	**Phone #**	**Email***	**Site**	**Time Zone**	**Region**	**Zip Code**	**State**	**Hire Date***	**Salary**	**Manager Id**	**Department**
President	515.123.4567	SKING	HQ	GMT-7	United States	12345	CA	6/17/2003 0:00	240,000.00		Executive
Administration VP	515.123.4568	NKOCHHAR	Burlington	GMT-4	United States	54321	MA	9/21/2005 0:00	170,000.00	100	Executive
Administration VP	515.123.4569	LDEHAAN	Toronto	GMT-4	Canada			1/13/2001 0:00	172,000.00	100	Executive
Programmer	590.423.4567	AHUNOLD	Toronto	GMT-4	Canada			1/3/2006 0:00	98,000.00	102	IT

Refresh Polymorphic Business Object Metadata

Clear polymorphic metadata when you publish a workbook to ensure the workbook gets the latest metadata. Oracle Visual Builder Add-in for Excel refreshes the polymorphic business object metadata during the first download operation performed after you open the published workbook.

Because polymorphic business object segments are configurable by customers and subject to change, metadata stored in the workbook may become stale.

To clear metadata when publishing a workbook, select **Clear all layouts** from the Publish Workbook window. See Publish an Integrated Excel Workbook.

Polymorphic Support Limitations

Before creating layouts for polymorphic business objects or adding descriptive flexfields to a layout, review the limitations here:

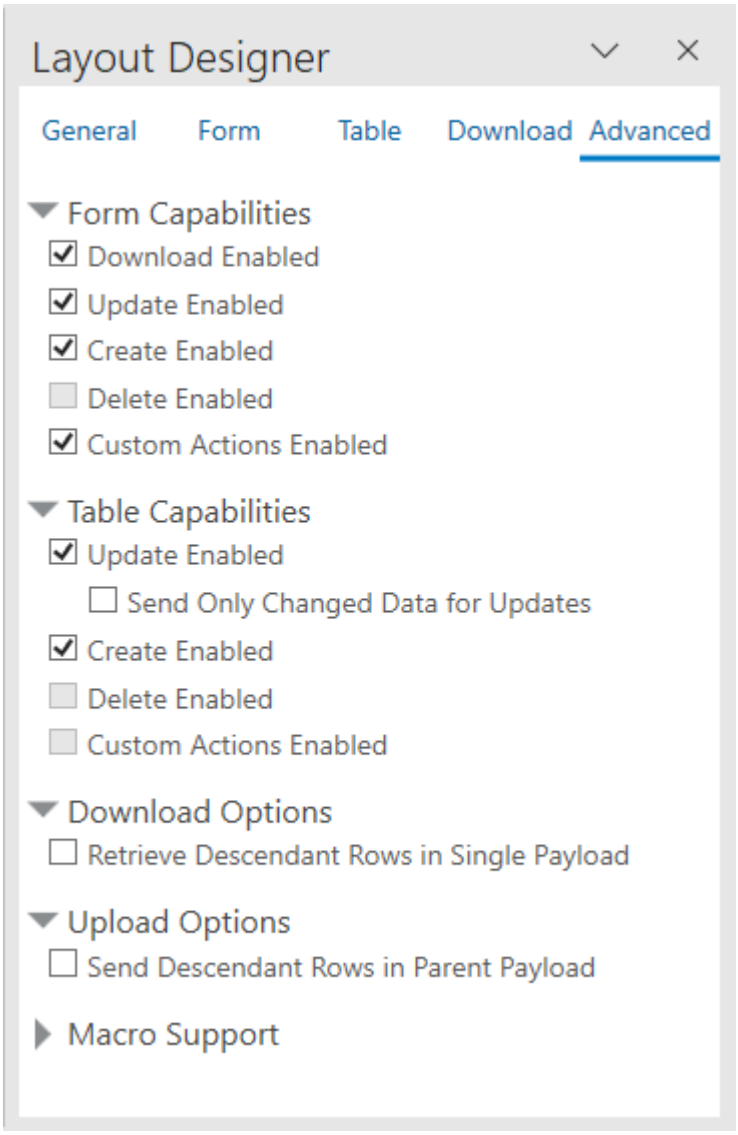
- Polymorphic business objects are only supported for ADF REST services.
- A polymorphic business object is assumed to only contain a single discriminator field. Multiple discriminators are not supported.

- The discriminator field must be a string.
- When a polymorphic column or form field expands, the order is global segments, the discriminator, and then context-sensitive segments. Note that:
  - Changing the order of the global segments or context-sensitive segments is not supported.
  - Hiding specific global or context-sensitive segments is not supported.
- In order to bind a polymorphic business object to a layout, it must expose polymorphic business object metadata. In an OpenAPI metadata document, this means that it must contain a "discriminator" and "oneOf" syntax in the schema for the business object. The "anyOf" polymorphic business object syntax is not supported.
- Hierarchical polymorphic business objects are not supported. All polymorphic segments must be defined directly on the polymorphic business object in the OpenAPI metadata document.
- Extensible Flexfields (EFFs) that define polymorphic segments on array-based subfields are not supported.
- When a layout contains a polymorphic business object, the "fields" and "expand" query parameters should not be manually configured in the **Download Parameters** area of the Layout Designer's **Download** tab.
- The case where a child business object's fields are determined by a parent business object's discriminator value is not yet supported.
- Limiting discriminator values for a polymorphic business object displayed in a Table layout does not limit:
  - The set of values that can be chosen in a list of values (LOV) for a discriminator field
  - The values users can provide for a discriminator field
- Polymorphic segments with a `DISPLAYHINT` value of `Hide` in the metadata are not displayed in the layout. If hidden segments are required, upload may fail.

Manage Layout Capabilities

Each layout in Oracle Visual Builder Add-in for Excel enables you to perform various standard and custom actions in an Excel workbook, so long as the operation is supported by your service. You can enable or disable these supported capabilities to control their availability in a layout.

1. In the Excel ribbon, click **Designer** to open a workbook's Layout Designer.
2. Click **Advanced** to see the layout's capabilities. Here's an example of a Form-over-Table layout, indicating form capabilities and table capabilities:



3. Select or deselect options as required. If the business object doesn't support an action, it won't be available for selection (like **Delete Enabled** under Table Capabilities as shown in this image). See REST Operations for more information about the REST support required for these options.

Layout Limitations

Here are some things to keep in mind when creating layouts for your integrated workbook using Oracle Visual Builder Add-in for Excel:

- Excel table objects, such as those created from the Excel ribbon using **Insert > Table**, are incompatible with the Table layouts used by the add-in.
- Do not save your workbook to the Excel 97-2003 workbook (`.XLS`) file format. Only the `.XLSX` and `.XLSM` file formats are supported. If you save to the `.XLS` format, the add-in disables commands in the **Oracle Visual Builder** ribbon.

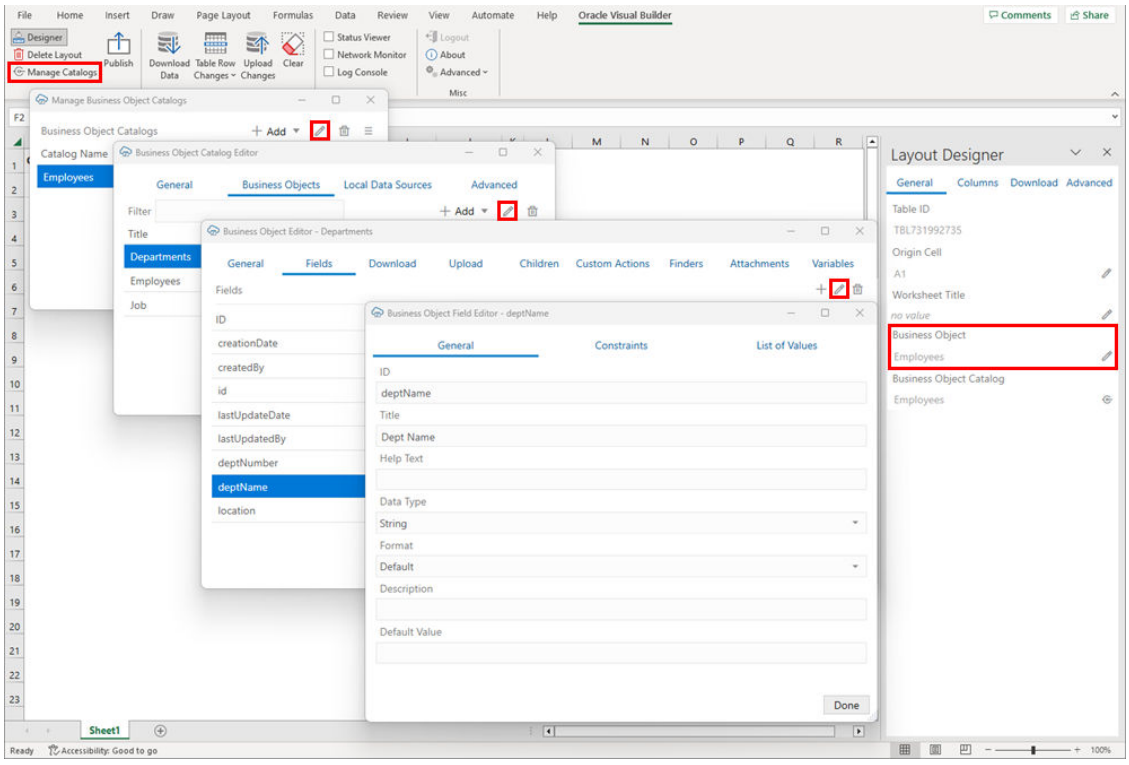
5

Manage Catalogs and Business Objects

When you create a layout, you provide the service metadata document for the REST service which Oracle Visual Builder Add-in for Excel uses to generate the Business Object catalog for your workbook. If required, you can use the available editors to modify the catalog in a variety of ways to enhance the overall user experience. For example, you can use the Business Object Field Editor to add help text to business object fields.
When you edit the details of business objects in your workbook, you're only telling the add-in how the service operates, you are not telling the service what it should do. Service behavior cannot be changed from the workbook. So any changes you make to the catalog in the workbook must be compatible with the service.

To view the editors progressively, start by clicking **Manage Catalogs** from the Oracle Visual Builder tab. This opens the Manage Business Object Catalogs editor. From here, you can choose a catalog and open the Business Object Catalog Editor, and so on.

You can also open the business object and business object field editors from the Layout Designer. Both options (using **Manage Catalogs** and the Layout Designer) are shown here:



Use these editors to perform tasks such as these:

- Manage Business Object Catalogs: Import, open, and refresh a catalog
- Business Object Catalog Editor: Configure the host and base paths for a catalog, as well as configure the business objects in the catalog. You can also configure more advanced settings such as authentication, GZIP, and the REST API Framework.

- Business Object Editor: Override the base path of the object, manage metadata path information, configure pagination, and configure row finders. You can also find **Download** and **Upload** tabs for configuring download and upload operations.
- Business Object Field Editor: Edit field titles, change when a field is editable, adjust the field data types (Advanced), and configure Help text. You can also configure the list of values for a field.

> **Note:**
>
> When reading through the topics in this section, you'll come across many terms such as OpenAPI, REST service, and path. Before you begin, refer to Key Concepts, Components, and Terms to familiarize yourself with some of these terms.

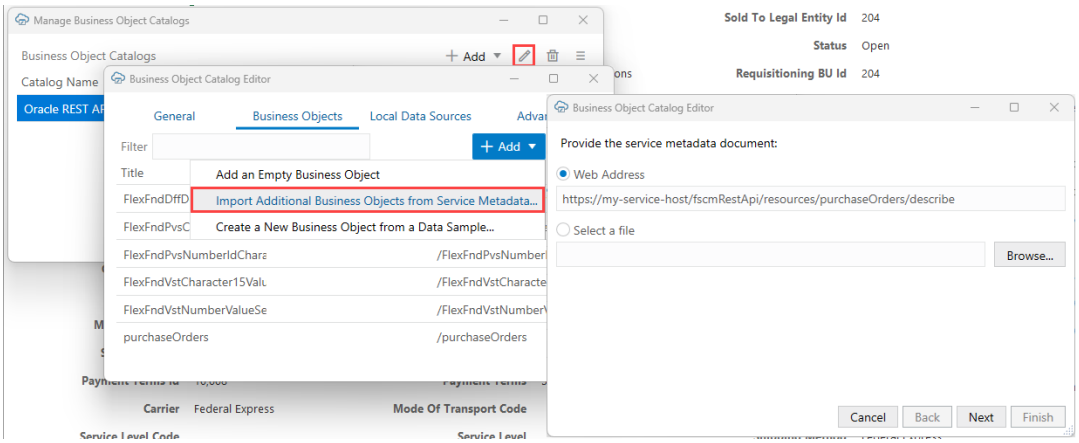
Add Business Objects to an Existing Catalog

If your catalog is missing one or more business objects, you can add new business objects to the catalog using the Business Object Catalog Editor. These new business objects must be available at the same host as the other business objects in that catalog.

> **Note:**
>
> The REST API type of the new business object must match the catalog. For example, you can't add an ORDS business object to an ADF REST catalog.

This is particularly useful when configuring a list of values.

1. Click **Manage Catalogs**.
2. In the **Manage Business Object Catalogs** window, select your existing catalog and click the Edit Business Object Catalog (✎) icon.
3. From the **Business Object Catalog Editor**, click **Business Objects**.
4. From the **Business Objects** tab, click **+ Add**, then select **Import Additional Business Objects from Service Metadata** from the menu.



5. Provide the URL or a file that contains the OpenAPI metadata of the new business objects and click **Next**.
6. Select the business objects you want to add, then click **Next**. If the list is large, type text in the Filter field to filter the list.

   The add-in only harvests the metadata for the selected business objects as well as any business objects referenced in any configured lists of values. The add-in also checks for and skips any duplicate business objects.

   When the add-in has imported the selected business objects, the wizard displays details of the action.
7. Review the details of the new business object, then click **Finish**.

   If there are any errors in the service metadata document, click **Save Report** to save the report to your local drive. Share this report with the service owner.
8. Click **Done** first in the **Business Object Catalog Editor**, then in the **Manage Business Object Catalogs** window.

Import a Business Object Catalog

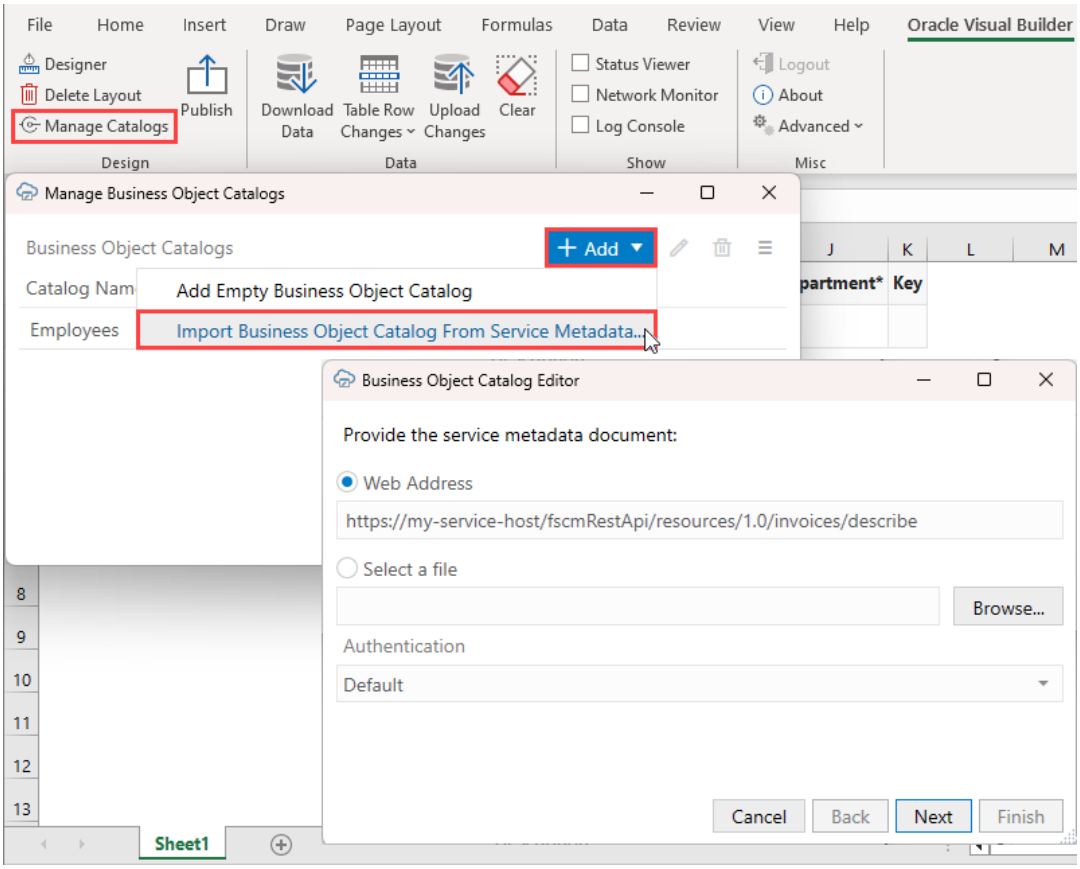
You can import a business object catalog by providing Oracle Visual Builder Add-in for Excel with the catalog's service metadata document.

You may also want to import an additional catalog if you need to use a different service at a different host or framework type.

> **Caution:**
>
> If you want to add an additional business object to your workbook, you should generally add it to your existing catalog. See Add Business Objects to an Existing Catalog for the steps.

To import a catalog:

1. Open the workbook where you want to import the catalog, then click **Manage Catalogs** from the Oracle Visual Builder ribbon.
2. From the **Manage Business Object Catalogs** editor, click **+ Add**, then select **Import Business Object Catalog From Service Metadata** from the menu.



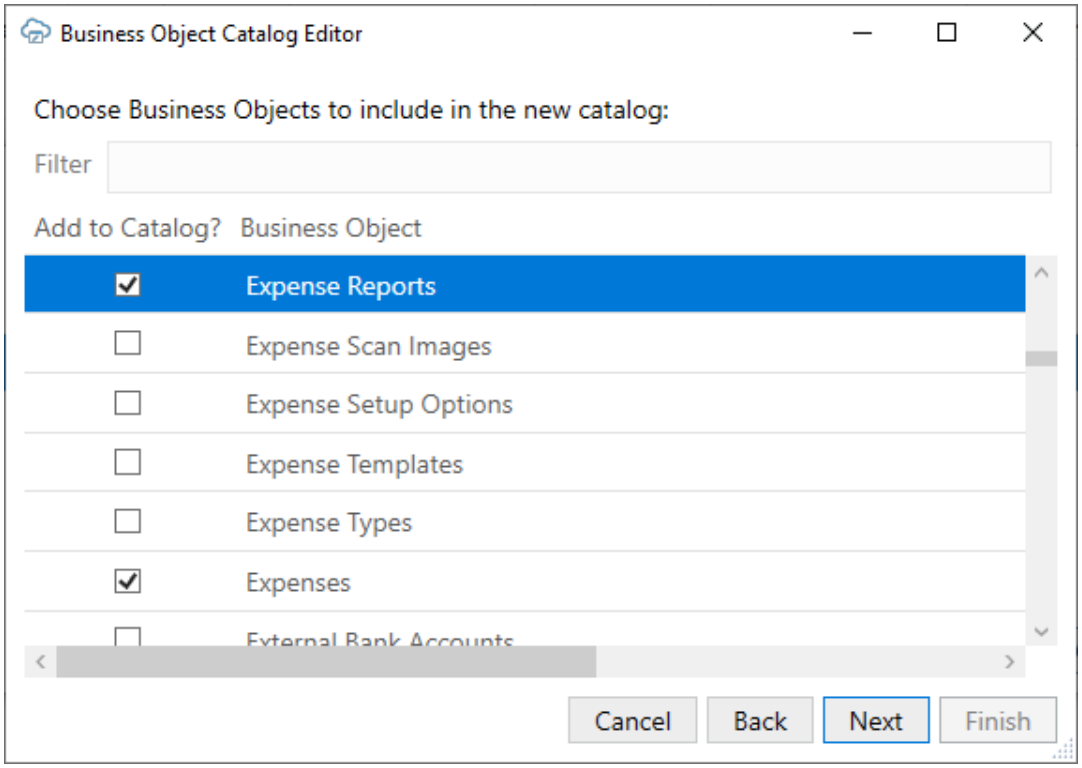
3. Provide the URL for the service metadata document.

> **Note:**
>
> If you are importing an Oracle ADF REST Resource catalog, it is recommended that you use minimal mode to limit the amount of metadata retrieved for an application.
>
> To use minimal mode, append the `metadataMode=minimal` query parameter to your URL like this: `.../fscmRestApi/resources/latest/describe?metadataMode=minimal`

4. If required, provide authentication details, then click **Next**.

   For more information about authentication settings, refer to the authentication configuration steps in Create a Table Layout in an Excel Workbook.

5. If the service includes five or more business objects, select the business objects you want to include in the catalog, then click **Next**.



The add-in only harvests the metadata for the selected business objects as well as any business objects referenced in any configured lists of values.

The wizard displays details of the newly-created catalog, such as the catalog name, service host and base path, and number of business objects.

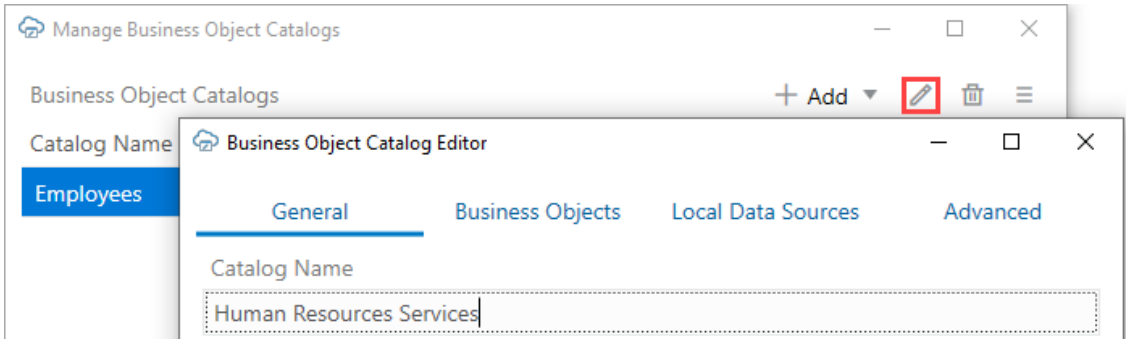
6. Review the new catalog details.

> **Note:**
>
> If there are any errors in the service metadata document, click **Save Report** to save the report to your local drive. Share this report with the service owner.

7. Click **Finish** to close the Business Object Catalog Editor wizard.

Once the catalog has been created, review other topics in this and other chapters to see how you can improve the catalog for your workbook's layouts. For example, you may want to add help text or lists of values to fields in the new catalog's business objects. See Add Help Text to Your Workbook or Use Lists of Values in an Excel Workbook.

You may also want to edit the name of the catalog in the Business Object Catalog Editor to make it easier to understand its purpose. Very often, the title that the add-in displays for the catalog in the Excel workbook is the name of one of the business objects that the service exposes. For example, this image shows a catalog in which the default value for the Title property is "Employees". This catalog exposes an Employees business object, so to avoid confusion, you can change the title that the catalog uses in the Excel workbook to something like `Human Resources Services`.



> **Note:**
>
> Certain OpenAPI document properties, such as Description, can contain formatting hints. The add-in displays the description text as is, with no interpretation of such hints.

Create a Business Object Catalog from a Data Sample

Use Oracle Visual Builder Add-in for Excel to create a business object catalog from a data sample when you don't have an OpenAPI-compliant service metadata document.

If you have the host, base path, and resource name for the service, the add-in can form the REST endpoint URL and send a GET request to the service to fetch a sample of data. The add-in can then use this sample to produce a list of fields available for that REST resource.

The add-in does this by first analyzing the JSON response to find the first JSON array available. It then selects the first JSON object in that array and, from this object, produces a list of business object fields. The add-in then creates a new business object in the catalog with that list of fields.

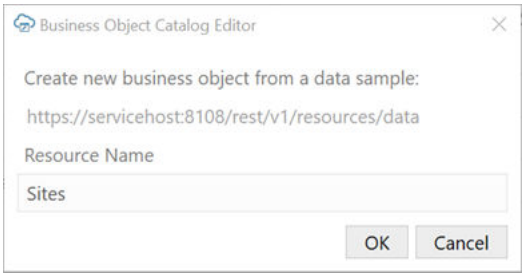
This new business object includes best guesses for the following:

- The Collection Path including GET and POST (create) operations
- Item Path including GET, PATCH (update), and DELETE operations
- Field properties such as data type

> **Note:**
>
> These guesses may not match the actual capabilities of the service. It is recommended that you review the business object definition and make sure it matches the service.

1. Click **Manage Catalogs** from the Oracle Visual Builder ribbon.
2. Open the desired catalog or create a new one.
   Make sure the Host and Base Path properties are set correctly.
3. From the Business Objects tab, click **+ Add**, then select **Create a New Business Object from a Data Sample** from the list.

4. Type in the name of the desired resource and click **OK**.



5. Review the business object definition to ensure it matches the service.
Here are some things to watch for:
   - If the service does not support delete, remove that operation from the item path *or* turn off delete in any layout that uses this business object.
   - If a given field is required or read-only, update the field settings from the General tab of the Business Object Field editor. Date fields may appear as string fields. Adjust the data type of fields as needed.
   - The add-in can't determine the data type of fields with null values. In these cases, the data type is set to "unsupported". These fields are not eligible for layouts, search, and so on. For these fields, correct the data type using the Business Object Field editor.
   - The add-in guesses that the primary key field is named "id". If the primary key field has a different name, use the Business Object Editor to update the item path as well as the parameter on each operation in the item path.

Configure Business Object Fields

Use the Business Object Field Editor to view and modify field settings as required.

Many of these settings control the appearance and behavior of the fields in your workbook and can be modified as desired. For example, you can enhance the usability of your workbook by:

- Changing a field title to something more intuitive.
- Overriding the default format for a field.
- Adding help text to help your business users understand the purpose of the field.
- Setting data validation behavior for the field.
- Defining custom field validation rules.
- Requiring the business user to provide a value for a field in the search prompt.
- Adding a list of values to ensure your business users can only enter a valid value for the field.

> **WARNING:**
>
> Some settings in the editor come from the service metadata and reflect how the REST service behaves. Changing these settings in the workbook cannot change the behavior of the service and may result in errors from your REST service. Refer to the notes in the table for recommendations for each field.

To modify the settings for a business object field:

1. Open the business object that includes the field in the Business Object Editor.

   You can navigate to the Business Object Editor from the Layout Designer or by clicking **Manage Catalogs**, then opening first the catalog and then the business object.
2. From the Business Object Editor, click the **Fields** tab.
3. Select the field you want to modify, then click the Edit icon (✎) to open the field in the Business Object Field Editor.
4. From the **General** tab, view or modify the settings as appropriate.

Setting	Description
ID	The ID for the field. **WARNING:** Do not modify the ID. The ID value must match the JSON member name expected by the service. JSON member names are case-sensitive.
Title	The name of this field. This value is used in various places, such as for the column header or form field label. This value can be localized. Provide a short value that is meaningful to your business users.
Help Text	A description of the field intended for business users. This value appears near the title where possible. This value can be localized. Provide a brief explanation of what values are expected for this field. See Add Help Text to Your Workbook
Data Type	The data type for the field, used for encoding and decoding data from the service, as well as data validation and cell formatting. Data types include values such as: • `Boolean` • `Date (no time)` • `Date-Time` • `Integer` • `Number` • `String` See Supported Data Types. **WARNING:** Do not modify the data type without consulting the service owner.

Setting	Description
Format	The cell formatting setting for the field. The default setting (**Default**) indicates that the field uses the standard formatting for the data type, for example, *mm/dd/yyyy* for date fields. To use a different format style, select an option from the list such as **Long Date** for a date field. This option displays "7/15/2022" as "Friday, July 15, 2022". See Choose Field Formats.
Description	An internal technical description for the field. This value only appears in the designer. It is not localizable.
Default Value	A default value for the field. The add-in populates the field for a new row with this value during row creation. If the field is editable, the business user can change the value before upload. You can use an expression in the default value field. Some functions are supported depending on the data type. For example, the add-in supports the `Now()` and `Today()` functions for the date data types. See About Expressions. **Note:** Reserved words are not supported.
Subfields	The subfields for a object-typed field such as those available in NetSuite SuiteTalk REST web services. To make changes to a subfield, select it from the Subfields table, then click the **Edit Field** (✎) to open the Business Object Field Editor for the subfield. See Object-typed Fields and Subfields.

5. From the **Constraints** tab, view or modify the settings as appropriate.

Setting	Description
Required for update and Required for create	Ensures a value is provided for the field during create or update: • If the check box is selected, the add-in checks that there is a value in the field cell and displays a data entry error if there is no value. The business user won't be able to upload the new or updated row until a value is provided. See Data Validation. • If the check box is not selected, the add-in doesn't require a value. The business user can upload the new or updated row without a value for this cell. **WARNING:** Don't deselect these check boxes without first consulting with the REST service owner. These settings come from the service metadata and reflect requirements of the REST service. If you deselect these check boxes for a field required by the service, the service may return an error, similar to a `(400) Bad Request` error, if no value is provided. See Required Fields.
Editable on update and Editable on create	Allows or prevents write operations on the field during create or update: • If the check box is selected, the business user can provide a value during create or update. • If the check box is not selected, the field cells are set to read-only. See Data Validation.
Searchable	Determines if a field can be used in a search. If selected, the field is available when creating a search for the workbook. See Use the Search Editor to Find Required Data. **WARNING:** The default value depends on the service metadata. Make sure that the service supports searching on the field before selecting this check box.

Setting	Description
Required for search	Determines if a value is required for the field during a search: • If the check box is selected, a value must be provided for the field on download. Required fields are indicated by an asterisk (*) in the Search Editor. • If the check box is not selected, no value is required. See Use the Search Editor to Find Required Data.
Omit from payload if value is empty	Determines if the field is omitted from the payload if the cell value is empty. If the check box is selected, empty values are omitted. See Omit Empty Values During Upload.
Minimum Length and Maximum Length	The minimum or maximum length of the value in characters for a string-based field. The value must be a number between 0 and 2147483647. If a value is set for one or both of these properties, the add-in checks that the number of characters in the field cell meets the set minimum or maximum value. If the value is less than the minimum or greater than the maximum, a data entry error is displayed. The business user won't be able to upload the new or updated row until a value meeting the minimum and/or maximum length is provided. See Data Validation. **Note:** Current versions of Excel limit the length of text for a single cell to 32,767 characters. See Excel specifications and limits.

6. Define a custom field validation rule using the **Validation Rule** and **Validation Failure Message** fields as described in Create Field Validation Rules.
7. To configure a list of values on the field, click the **List of Values** tab and configure it as described in Configure a List of Values with a Business Object.

Set an Authentication Method for a REST Service

Configure authentication for Oracle Visual Builder Add-in for Excel when connecting to the REST service.

When you create a new catalog from a URL, you can configure the authentication method depending on the service. The add-in provides five authentication options: Default, Basic Access Authentication, Oracle Fusion Applications Token Relay, OAuth 2.0 Authorization Code (PKCE), and No Authentication.

At log in, the add-in uses this setting to determine how to log in. If required, you can change the authentication method using the Advanced tab of the Business Object Catalog Editor.

See Authentication Options for more information.

1. Click **Manage Catalogs**.
2. In the **Manage Business Object Catalogs** window, select your existing catalog and click the Edit Business Object Catalog icon.
3. In the **Business Object Catalog Editor**, click **Advanced**.
4. Select an authentication method from the **Authentication** list:
   - **Default:** At login, the add-in pings an Oracle Cloud Application anti-CSRF servlet endpoint. If the ping succeeds, Token Relay is used. If the ping fails, Basic authentication is used instead.
   - **Basic Access Authentication:** See Basic Authentication.
   - **Oracle Fusion Applications Token Relay:** See Oracle Fusion Applications Token Relay Authentication.
   - **OAuth 2.0 Authorization Code (PKCE):** See OAuth 2.0 Authorization Code Flow with PKCE.
   - **No Authentication:** There is no prompt for credentials. No authentication-related headers are added to requests.

   > **Caution:**
   >
   > Be sure to choose an authentication method that is compatible with your catalog.

5. If the authentication method includes configuration properties, the **Edit Authentication Properties** button is enabled. Click the button to edit those properties.

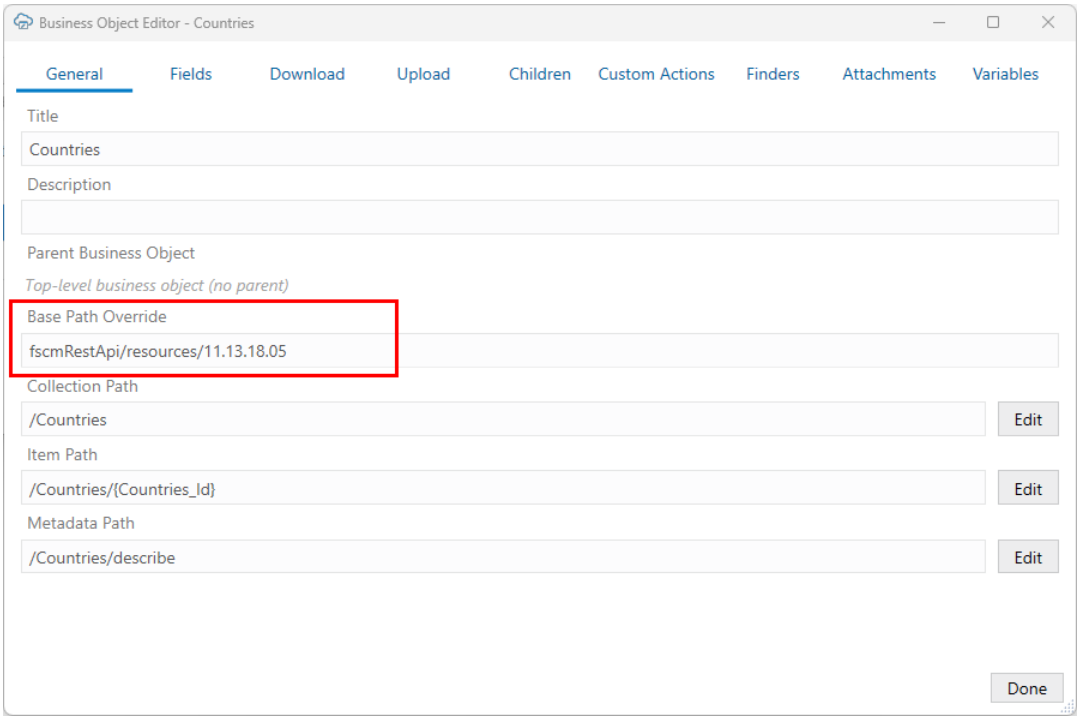
Override a Business Object's Base Path

You can configure a business object to use a different base path than the rest of the catalog. The Oracle Visual Builder Add-in for Excel then uses this base path when making REST requests for the business object.

Typically, a business object catalog holds business objects that share a base path (say, `/crmRestApi/resources/11.13.18.05`). But if a business object needs to use a different base path (for example, your list of values come from `fscmRestApi/resources/11.13.18.05`), you can provide a new base path for the business object through the Business Object Editor.

To configure a business object to use a base path different from the one shared by business objects in the Catalog:

1. Add a business object to your Business Object Catalog. See Add Business Objects to an Existing Catalog.
2. Click the **General** tab in the Business Object Editor, enter the different base path in **Base Path Override** to override the base path used by all business objects in the Catalog, and click **Done**.



The add-in assumes that no extra authentication is required by business objects with a base path override.

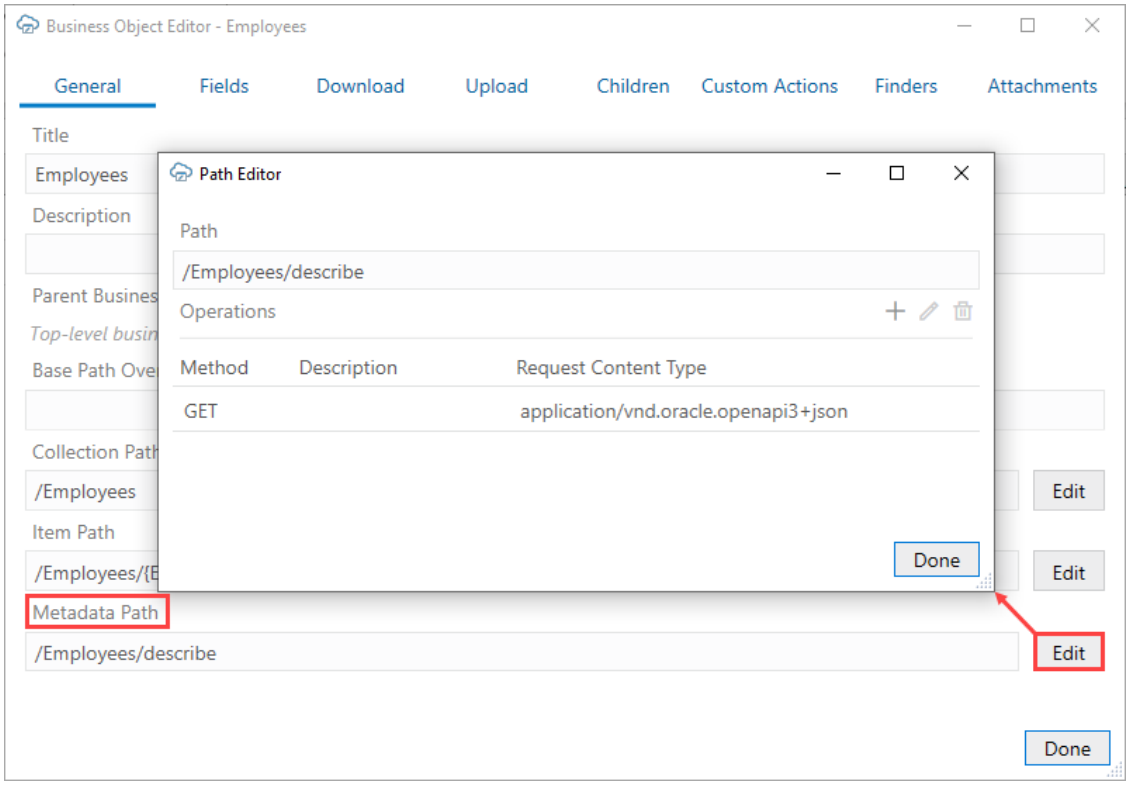
You can now use the newly created business object as the data source business object for a list of values if required. See Configure a List of Values with a Business Object.

Manage Metadata Path Information

When you provide a service metadata document, Oracle Visual Builder Add-in for Excel captures the path to the service metadata document relative to the service host and base path of the catalog. You can view this path and other path settings from the Business Object Editor.

This relative path is displayed in the **Metadata Path** field on the **General** page of the **Business Object Editor**. The add-in uses this path when you refresh the business object catalog. See Refresh a Business Object Catalog.

The add-in also captures path information such as REST API methods and parameters and displays these details in a couple of editors: the **Path Editor** and the **Operation Editor**. Open the **Path Editor** by clicking the **Edit** button next to the **Metadata Path** field.



The **Path Editor** lists all the available methods. To view settings for a method, including headers and parameters, select a method from the list and click the **Edit Operation** icon (✎) to open the **Operation Editor**.

The values seen in this image are correct for an ADF REST service. Other service frameworks may require other values. For example, ORDS might have a metadata path of "`/open-api-catalog/employees`" and a content type of "`application/json`". Refer to Oracle REST Data Services for more details.

> **Caution:**
>
> The metadata path should include a GET method that specifies the correct request content type that the service expects for a metadata request. If you need to modify these settings, be sure to provide values that will result in the service returning a proper OpenAPI service metadata document.
>
> Consult the REST API owner for the required values.

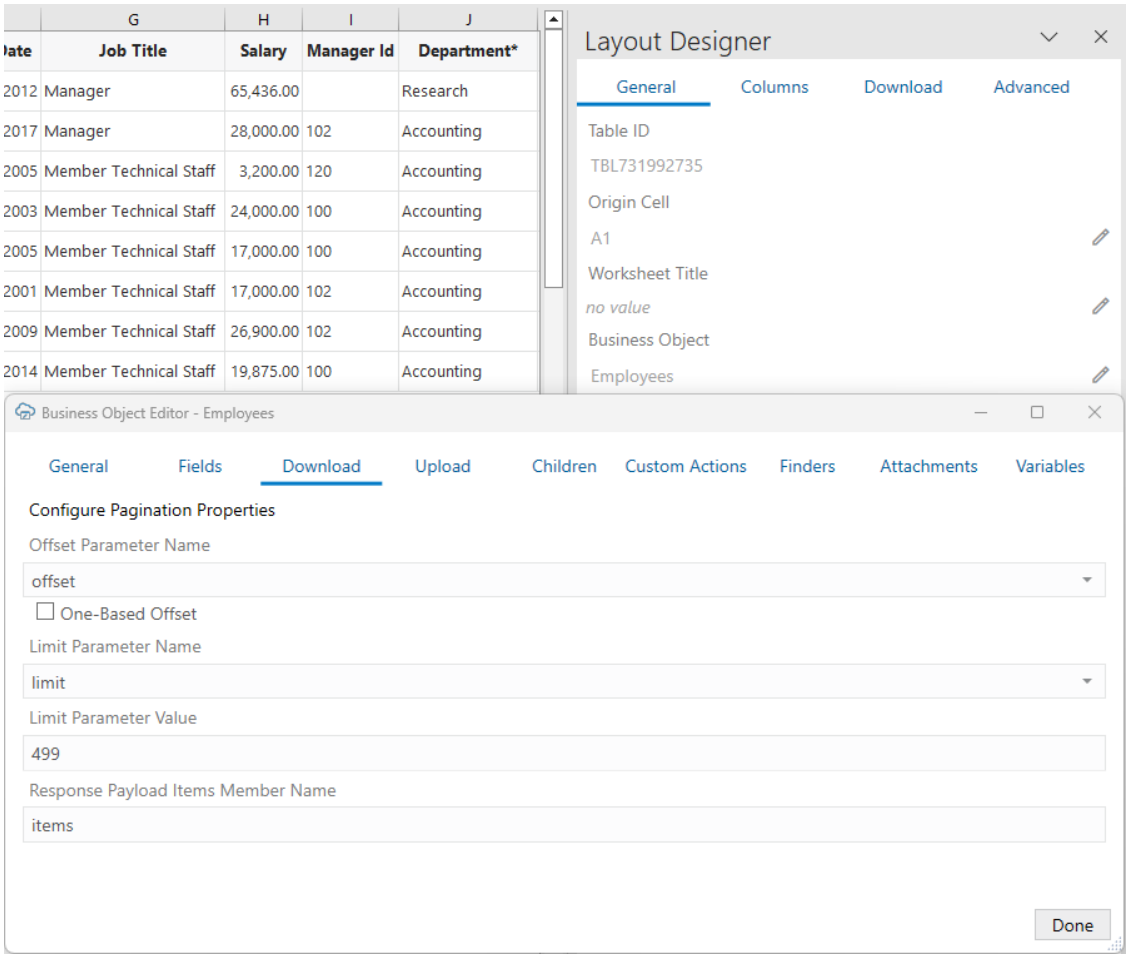
Configure Pagination for a Business Object

If the REST service supports pagination, you can download a large number of rows using multiple requests.

Imagine you need to download 10,000 rows of data. Downloading one row at a time is too time-consuming, and attempting to download all 10,000 rows in one request might result in a timeout error. Instead, download one page at a time where the page contains, for example, 500 rows.

> **Note:**
>
> Pagination does not limit the total number of rows downloaded. All available rows are downloaded with or without pagination. Pagination controls how many rows are downloaded *per request*. In the example here, there would be 20 requests of 500 rows to download all 10,000 rows.

You can configure the pagination behavior using the Download tab in the Business Object Editor.



Configure the following as required:

- **Offset Parameter Name**: The name of the URL parameter that controls where to start the next page. When fetching the first page, the add-in uses a value of zero. When fetching the second page, the add-in uses a value of 499, assuming the limit value is 499.
- **One-Based Offset**: Controls whether the service starts counting from one. If this check box is unselected, the service assumes the service starts counting from zero.
- **Limit Parameter Name**: The name of the URL parameter that controls how many rows to fetch for each page.
- **Limit Parameter Value**: Controls the page size (number of rows that the add-in downloads).

For example, using the defaults for an ADF REST service, the add-in appends `?offset=0&limit=499` for the first download request.

For other service types, pagination may or may not be supported. If supported, the service may use parameter names like `offset` and `limit` or it may use other parameter names for the same purpose.

Consult the service API documentation to determine which parameters to use.

Refer to Download Data for more information.

Use Row Variables for a Business Object

A row variable is a custom field you can add to a layout that allows business users to capture details for a row, such as a range start date for a date effective object. The row variable feature is intended to provide improved support for date effective objects.

When creating an integrated workbook for a date effective business object, you'll need to configure additional REST request headers of type `Effective-Of`. These headers may include a range attribute, such as `RangeStartDate`, that requires a date. See Support for Date Effective Objects.

The row variable feature provides a way for the business user to capture an appropriate date for a row that can be used in the header during upload.

Configure a Row Variable for a Layout

Create a row variable for a business object and add it to your layout. A row variable is similar to a business object field and allows business users to capture details for a row, such as a range start date for a date effective object.

Unlike a business object field, a row variable is not defined in the service metadata and the value of a row variable isn't included in upload request payloads.

This feature is intended to provide support for workbooks with date effective objects. If you create a layout for a date effective object, you may need to create a REST request header of type `Effective-Of` that includes a range start date. This header can use an expression to reference a date value stored in the row variable you create.

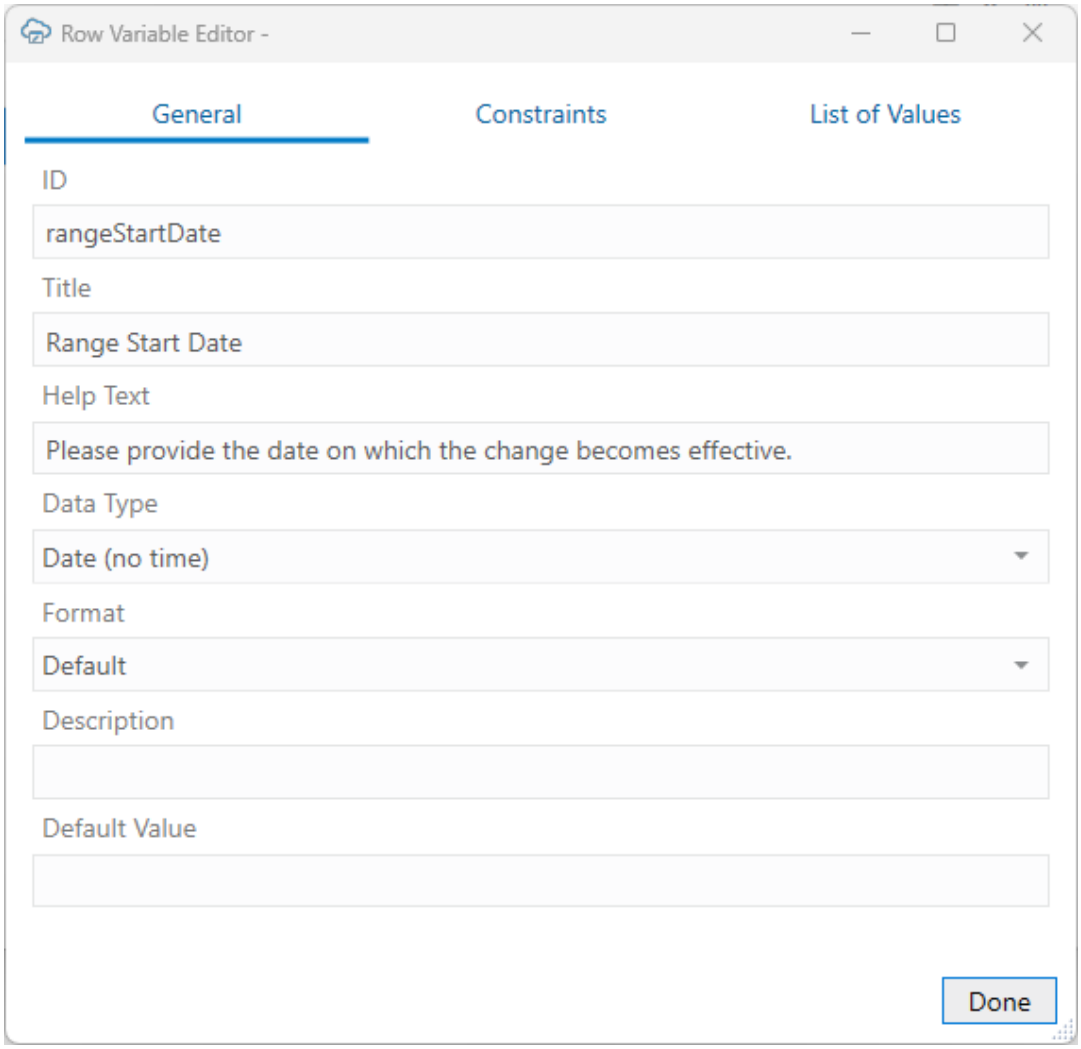
This task uses the example of a row variable to capture the `RangeStartDate` range attribute that will be referenced in an `Effective-Of` REST request header. See EffectiveOf Headers in Multi-Row Requests for more information.

To configure a row variable for a layout:

1. Open the Business Object Editor for the data effective object.
2. Click the **Variables** tab in the Business Object Editor.

   Row variables are not defined in service metadata so the **Row Variables** list is empty until you create one.
3. Click **Add Row Variable** (+).
4. Select the new variable, then click **Edit Row Variable** (✎) to open the Row Variable Editor.
5. Type an ID and title for the row variable, then select a data type from the **Data Type** list.

   In this case, choose **Date (no time)** for the date effective object's new row variable.



6. If required, configure the row variable with help text, a description, and a default value.

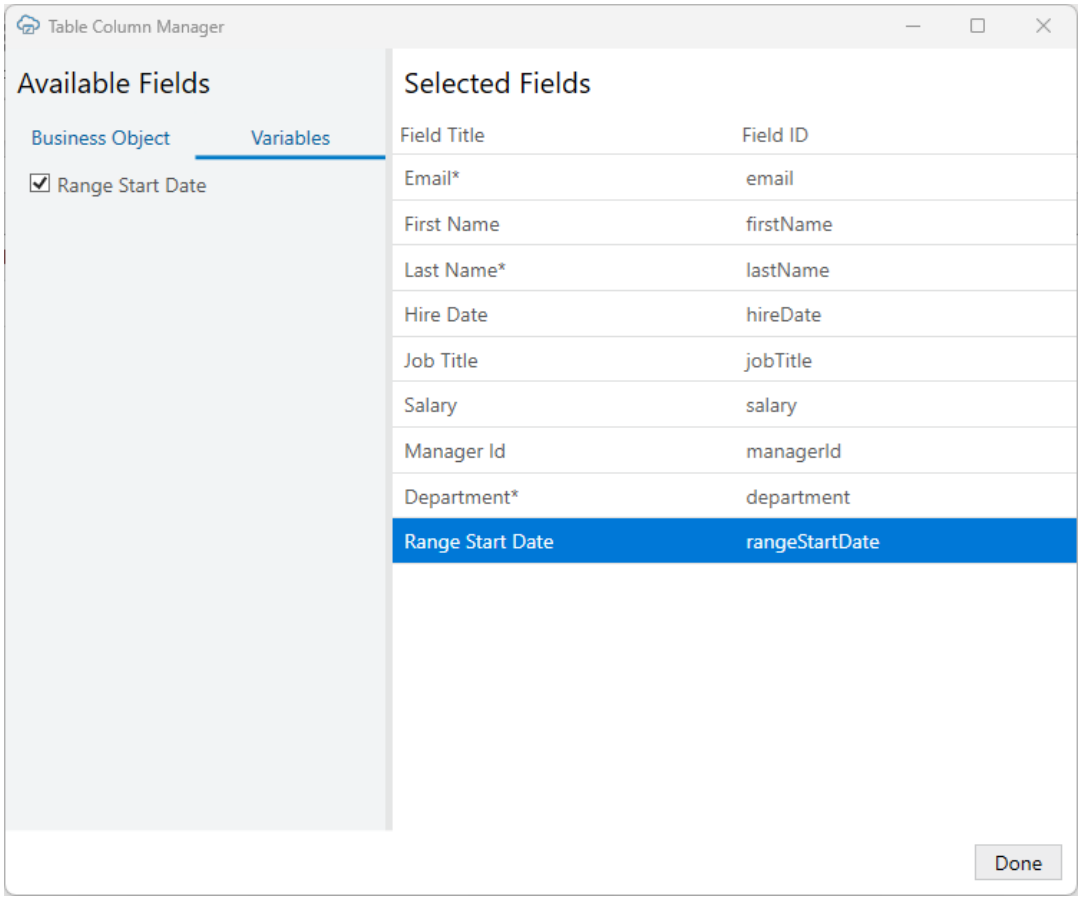
   The **Help Text** value is displayed in a popup next when the column header or form field label is selected. This value can be localized. The description is an internal technical description and only appears in the designer.

   The default value is used to populate the row variable for a new row during row creation. If the variable is editable, the business user can change the value before upload.

   You can use an expression in the default value field. Some functions are supported depending on the data type. For example, the add-in supports the `Now()` and `Today()` functions for the date data types. See About Expressions.

   > **Note:**
   >
   > Reserved words are not supported.

7. Click **Done** to close the Business Object Editor.
8. From the Layout Designer, click the **Columns** tab, then click **Manage Columns** (+).
9. From the Table Column Manager, click the **Variables** tab from the **Available Fields** pane, then select the new row variable—in this case, **Range Start Date**.



If required, drag the row variable to another position in the **Selected Fields** list.

Now that you've created a row variable and added it to your layout, your business users can use this row variable to select a range start date for an updated row.

If you've created the row variable for a date effective object, go ahead and configure a REST request header that references this row variable. See Configure a Request Header

Configure GZIP Compression for Request Payloads

For a POST, PUT, or PATCH request to the service that takes payloads, you can choose to compress the payload body using GZIP if the service accepts compressed request payloads.

When the **Supports gzip compression for request payloads** option is selected, the `Content-Encoding: gzip` header is added to the request. This option (found in the Business Object Catalog Editor's **Advanced** tab) is selected by default for ADF REST services.

Refresh a Business Object Catalog

If a service owner makes significant changes to the service metadata—especially its business object definition—after the workbook is integrated with the service, you can refresh the workbook's business object catalog to take advantage of the latest changes.

Refreshing a catalog is useful when changes to the service metadata are extensive—for example, when new fields, custom actions, or row finders are added to the service—and

updating the business objects to incorporate these changes would take considerable time and effort. You can also refresh the catalog when a new version of Oracle Visual Builder Add-in for Excel improves the metadata harvesting.

You have two options when you refresh a catalog. You can choose to refresh all property values in the catalog with the latest values from the service metadata. Or you can choose to preserve the existing property values and only add new values from the service.

If you decide to give priority to the new metadata property values, here's what happens. If the add-in finds an existing matching field, it replaces the old title with the new title from the service metadata. If the new title is blank, it does not clear the old title. Be aware this refresh overwrites most of the changes made locally to the business object catalog. If you've spent a lot of time customizing the business object (especially field titles), choose the other option.

If you decide to give priority to existing metadata property values, the add-in does not replace your old values with the new values from the service metadata. However, it does apply a new value if the old value is blank.

Your catalog might be based on a single service metadata document that defines one or more business objects. But a catalog can also include business objects that you have imported separately and that have their own service metadata documents. When you refresh a catalog, the add-in reads the metadata path for each business object and issues a request to fetch the service metadata for that business object.

You can only refresh a catalog with URL-based service metadata documents. Catalogs with file-based service metadata documents cannot be refreshed from a file.

> **Note:**
>
> It is recommended that you create a backup copy of the workbook before refreshing a catalog.

To refresh a business object catalog:

1. Click **Manage Catalogs** in the Oracle Visual Builder tab.
2. Select the catalog to refresh in the Manage Business Object Catalogs window.
3. Click the menu icon (☰, then **Select the Refresh Business Object Catalog**.
4. From the Refresh Business Object Catalog dialog, choose an option for the refresh:
   - **Give priority to the new metadata property values**: Use this option to overwrite existing values in the catalog with new values from the service metadata.
   - **Give priority to the existing catalog property values**: Use this option to preserve existing values in the catalog and only add new values from the service metadata.
5. Click **OK**, then **Yes** when prompted to confirm the overwrite.

   If a service host is missing or you have not logged in, you'll be prompted to provide the required details. If you cancel either of these prompts, the catalog refresh ends without completing.

   If, once the catalog refresh is complete, you see this warning, then the add-in was unable to refresh one or more business objects in the catalog.

To troubleshoot the error, open the log console and repeat the process to discover the cause. See Log Console.

If the catalog refresh ends successfully, the add-in displays a message that all business objects were successfully refreshed.

6. When the refresh is complete, click **OK** to close the message, then **Done** to close the **Manage Business Object Catalogs** window.

   During the catalog refresh:

   - Existing business objects are refreshed based on the response from the metadata path. New business objects are added to the catalog. No business objects are removed.
   - For existing business objects, new fields, finders, custom actions, and so on are added and existing items are refreshed according to the refresh option you choose. No existing items are removed.
     - If you choose to refresh all property values, new values replace the older ones. For example, a field's title, required, and editable properties are copied from the newer version to the older version, except when a new property is empty. If a field's newer version has a list of values configured, the newer list of values replaces the older list of values. If the newer version doesn't have a list of values, the previous list of values is left unchanged.
     - If you choose to preserve your existing values, new property values are added. Only existing properties with null values are updated during the refresh.
   - A refresh does not change any field formats or clear help text since these values are always set manually and do not come from the service metadata.

7. Following the refresh, check your workbook to make sure you are satisfied with the results.

   Here are some things to be aware of:

   - The add-in's metadata harvesting improves over time. You may see changes from a refresh even if the remote metadata hasn't changed.
   - If the ID of an item has changed, it will be treated as a new item.
   - If you delete an item such as a row finder variable, it may reappear on refresh.
   - If you delete a property value, it may be given a new value on refresh.

If you want to add a new field to your layout, see Manage Fields in a Form or Table.

Configure the REST-Framework-Version

You can change the value of the REST-Framework-Version request header for ADF REST and VBBO catalogs from the Business Object Catalog Editor. The REST framework version determines functionality for accessing business objects.

> **Note:**
>
> The REST-Framework-Version request header is set to version 6 by default for ADF REST services.

To configure a specific REST-Framework-Version that overrides the default framework version:

1. In the Oracle Visual Builder tab, click **Manage Catalogs**.
2. Select the business object catalog, then click the Edit Business Object Catalog icon.
3. In the Business Object Catalog Editor, click the **Advanced** tab.
4. Specify the framework version you want to use in the **REST API Framework Version** property. This property is available only for Oracle ADF REST Resource services (not ORDS or other REST services).

   The default value is version 6 (the only supported value). If the value is empty or lower than 6, it is set to the default. Framework versions higher than 6 are not certified.
5. Click **Done**.

Once you specify a framework version, it stays in effect until you manually change it again. Every request sent to any endpoint that belongs to this catalog includes the REST-Framework-Version header with the specified value. For details on framework versions, see About REST API Framework Versions.

6

Configure Search Options for Download

You can configure the search options for a layout, which are used when the user clicks **Download** in the Oracle Visual Builder tab.

Search Option	Available for	Benefits	Limitations
Search Editor	ADF REST, VBBO, ORDS, and NetSuite services	• Provides a graphical user interface for building a search • Allows business user to provide values and modify the search	Some limits on search syntax
Row Finders	ADF REST	• Provides access to powerful, complicated searches defined by the service • Allows the business user to provide values	Only available for ADF REST services
Download Parameters	Any service type	Developers can use any syntax the service supports	Business users cannot provide values or change the search at download time

Use the Search Editor to Find Required Data

You can configure each layout of the workbook to allow a user to specify search values for filtering the data Oracle Visual Builder Add-in for Excel downloads to the layout.

For example, you might have a Form-over-Table layout that displays a purchase order in the form and associated lines in the table. In this scenario, you'll want to create a search that allows business users to enter the order number for a specific purchase order that they want to view.

You can use the Search Editor to create a filter on two or more fields. So, you could create a search that returns employee records where the job title equals "software developer" and the salary is over 95000. When you configure more than one condition, the add-in uses the logical AND operator between each one.

The Search Editor is supported for workbooks that integrate with ADF REST, VBBO, ORDS, or NetSuite services.

About Searches

You can create search conditions on one or more fields in the layout using the Search Editor. A search condition consists of a business object field, an operator, and a value.

An operator determines how items are matched based on the given value and is based on the field's data type. For example, available operators for strings include "starts with", "contains", and "ends with". Those for numbers include "greater than" and "less than". For dates: "before" and "after".

To download data for employees making over $95,000, for example, you'd choose the "Salary" field, select the "greater than" operator, and type "95000" in the value field.

If you configure more than one condition, the add-in always inserts the logical AND operator between each one. So, if you have two conditions such as "`Job Title = Software Developer`" and "`Salary > 95000`", the add-in downloads only those items that match both these conditions.

You can also use expressions for the condition value. For example, you could use the `Today()` date function along with an integer (`{ Today() - 90 }`) to find employees with hire dates in the last 90 days. See About Expressions.

Configure a Search for a Layout

You can configure a search for a worksheet using the Search Editor available from the Layout Designer. Using the Search Editor you can create one or more search conditions.

Before you begin, be aware that there are two business object field settings in the Business Object Field Editor related to search: **Searchable** and **Required for Search**. If a field isn't searchable, it won't appear in the Available Business Object Fields window. If a field is required for a search, the business user must provide a value at download. Required fields are indicated by an asterisk (*) in the Search Editor. See Configure Business Object Fields.

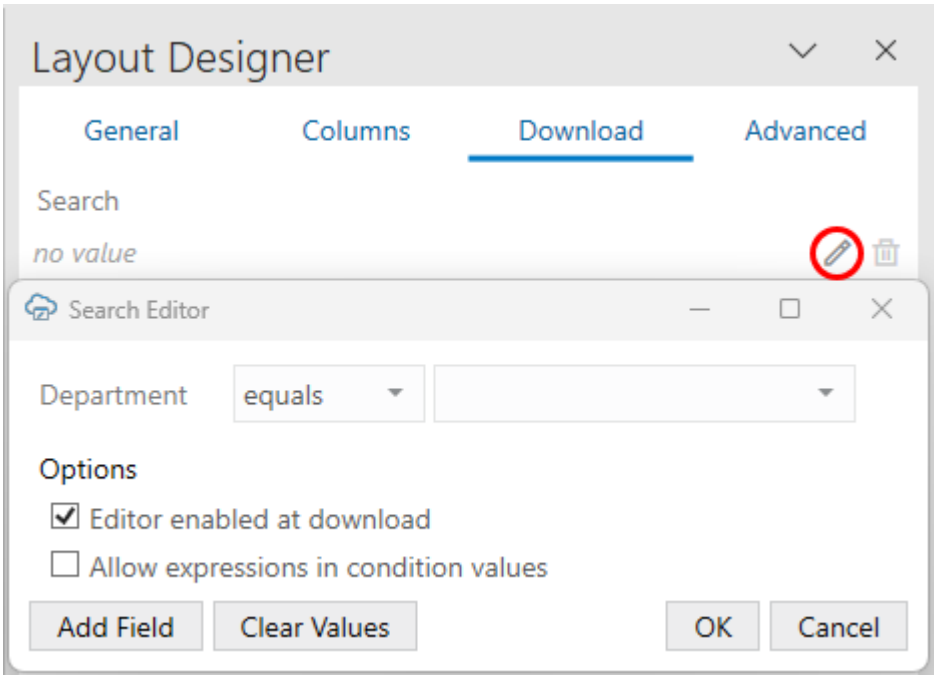
To create a search for a layout:

1. Select the layout you want to create the search for, then click **Designer** from the Excel ribbon.
2. From the Layout Designer's **Download** tab, click the Edit icon (✎) next to the **Search** property.

   If no search conditions have been defined, the add-in opens the **Available Business Object Fields** window.

   If you've already defined a search condition, the add-in opens the Search Editor instead. From the Search Editor, click **Add field** to add a new condition. This opens the **Available Business Object Fields** window.
3. From the **Available Business Object Fields** window, select the business object field that you want to enable users to enter search terms for.

   For example, select **Department Name** if you want to enable users to search on employees by department, as shown here:



4. Select an operator from the list beside the field name. For example, select "equals" to return employees from a department that matches the value in the value field or "not equals" to return only employees from other departments.

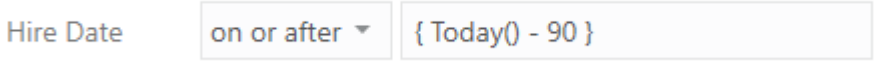
   The available operators depend on the data type. For example, string fields have filters such as "starts with", "contains", and "ends with".

   > **Note:**
   >
   > For Boolean fields, the only acceptable values are true and false, regardless of the current language preference. If desired, you can convert Boolean values to other labels by creating a list of values for the field that uses a local data source (LDS). See Use a Local Data Source for a List of Values.

5. If required, select or type a value in the value field you want to use to match items on download. For example, to download employees from the research department, select "Research". You can only specify a single value for this field.

   You can leave the value field blank if you want your business user to provide a value. If you do provide a value here, the business user can always change or clear the value when downloading.

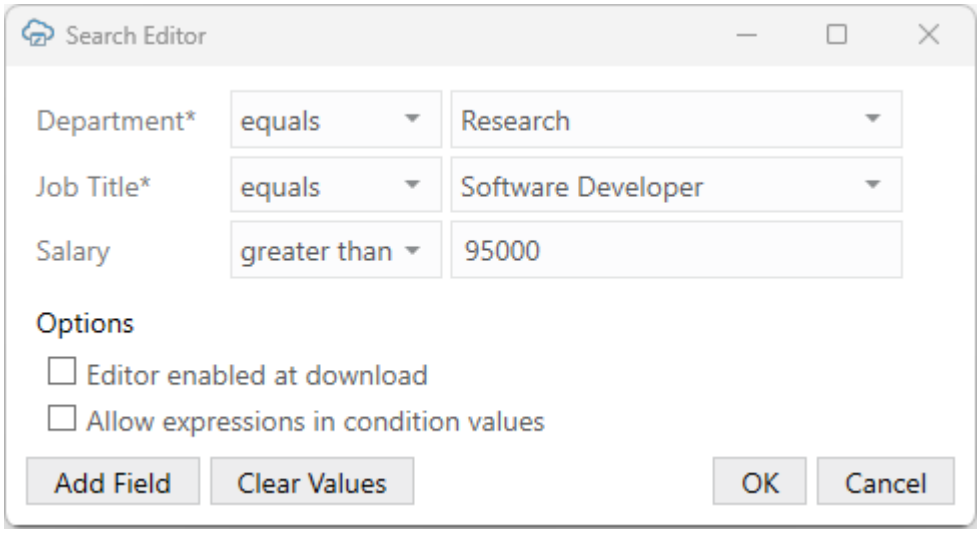
   The Search Editor also supports expressions in the value field with some restrictions. For example, you could use the "Today" function (`{ Today() }`) to return rows based on today's date. This sample condition for an Employees layout finds employees hired in the past 90 days:



> **Note:**
>
> Keep in mind that the add-in does not validate expressions in the Search Editor. You'll need to test your expression as described in step 10.

6. To add more fields for complex searches, click **Add Field**, then select a field from the **Available Business Object Fields** dialog.

   For example, you might want to search for employees with a job title of Software Developer, but only those who earn more than $95,000. In this case, first select **Job Title**, then click **Add Field** and select **Salary**. Finally, define the search parameters as follows:



> **Note:**
>
> When configuring your search, make sure the field your dynamic list of values is based on is added before the field with the dynamic list. So, to filter the Job Title list based on the Department value, add the Department field before the Job Title field in the Search Editor. See Configure a Filter with a Dynamic Parameter.

> **Tip:**
>
> To change the order of the conditions, right-click the label and choose the desired option.

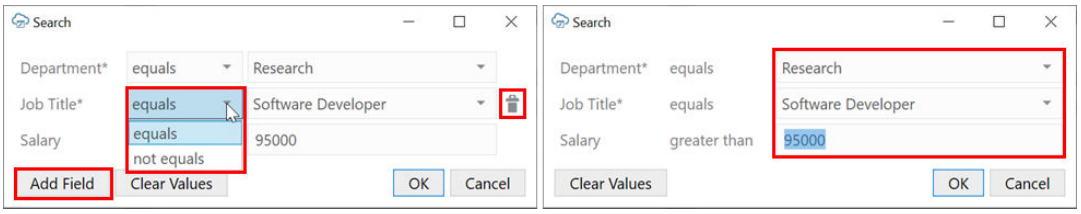
7. Select **Allow expressions in condition values** if you have used an expression in any of your configured conditions. Otherwise, the add-in treats the value as a constant and won't evaluate the expression.

   Expressions can reference workbook parameters. So, if the workbook parameter, `MinSal`, is set to "95000", the condition, `Salary > { Workbook.Parameters['MinSal'].Value }`, returns employees making over $95,000. See Use Workbook Parameters for Download.

> **Note:**
>
> Once you allow expressions in search conditions, the workbook can no longer be used with add-in versions prior to 4.0.

8. Select **Editor enabled at download** to allow business users to edit the search when downloading data. Deselect this check box to provide a simplified Search prompt instead.

   If this check box is selected, the business user can add and delete search fields as well as change search filters, as shown on the left:



   If the check box is unselected, only the values are editable, as shown on the right.

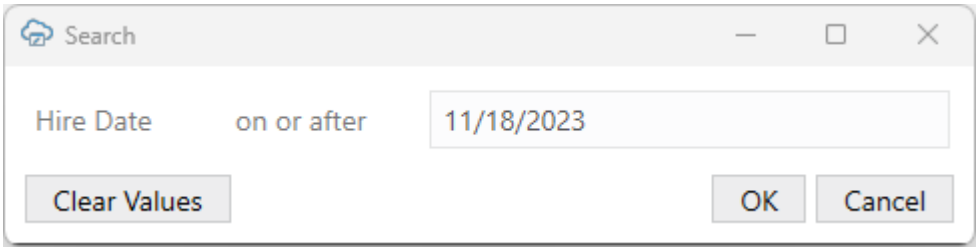
9. Click **OK**.

10. Test your configured search before releasing the workbook. Do this by performing a download and checking the Search prompt and download results.

    If the search value is unexpectedly empty at download time, review your expressions. For example, you may experience a problem if the expression result cannot be successfully converted to the appropriate data type for the condition. In this case, there is no error message. Make sure the expression matches the field's data type as configured in the Business Object Field Editor.

    If you need to change your search settings, make sure to click the Clear Layout icon before you re-test your search. The add-in stores the search definition with the layout and uses this cached definition for subsequent downloads. Clearing the layout ensures the add-in uses the latest search definition and evaluates any expressions anew during the next download.

When a search is configured, your business users are presented with the Search prompt during download. If a search condition contains an expression, this is evaluated at this time. Our sample expression, `{ Today() - 90 }`, is converted to a value 90 days before today's date, as shown here:



The add-in stores the details of the business user's search definition with the layout. This cached definition is used for subsequent downloads.

You can configure a list of values for a field so that, when the business user is presented with the Search prompt, they can select an appropriate value from a list. See Configure a List of Values with a Business Object.

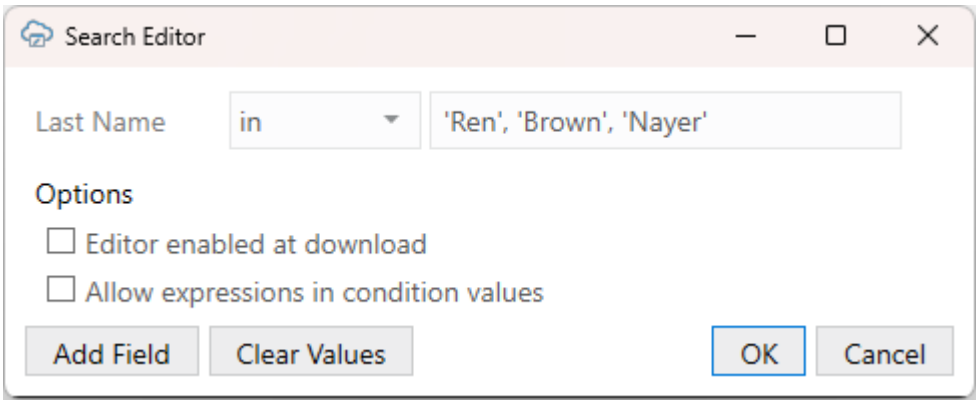
Searches even support cascading lists of values with dynamic filters. Let's suppose you want to download data for employees based on department and job title and have added the Department and Job Title fields to your search in order to do so. If the Department field has a LOV, you can choose the required department from the list. If the second field, Job Title, has a dynamic filter based on the Department field, the Job Title list shows only those job titles that are associated with the department you selected in the first field. See Configure a Cascading List of Values.

The IN Operator in Searches

Oracle Visual Builder Add-in for Excel supports the "in" comparison operator in searches for integer and string data-type fields. This operator is available for ADF REST, VBBO, and NetSuite (integer fields only) services .

The "in" operator allows you to return items with a field value that matches a value in a list you provide. The service treats the "in" condition like a series of `OR` conditions.

Suppose you have an Employees workbook and want to download employee rows for Sophie Ren, Dave Brown, and Julia Nayer. You could use the "in" operator on the Last Name field and include a list of the last names, like this:



The add-in returns all items matching any of the last names provided.

When entering your list of values, enclose string values in single quotes like this: `'Ren', 'Brown', 'Nayer'`.

You can also use the "in" operator on an integer-type field and provide a list of the integers you want to match. Do not enclose integer values in single quotes.

> **Note:**
>
> The "in" operator is not supported for fields configured with list of values.

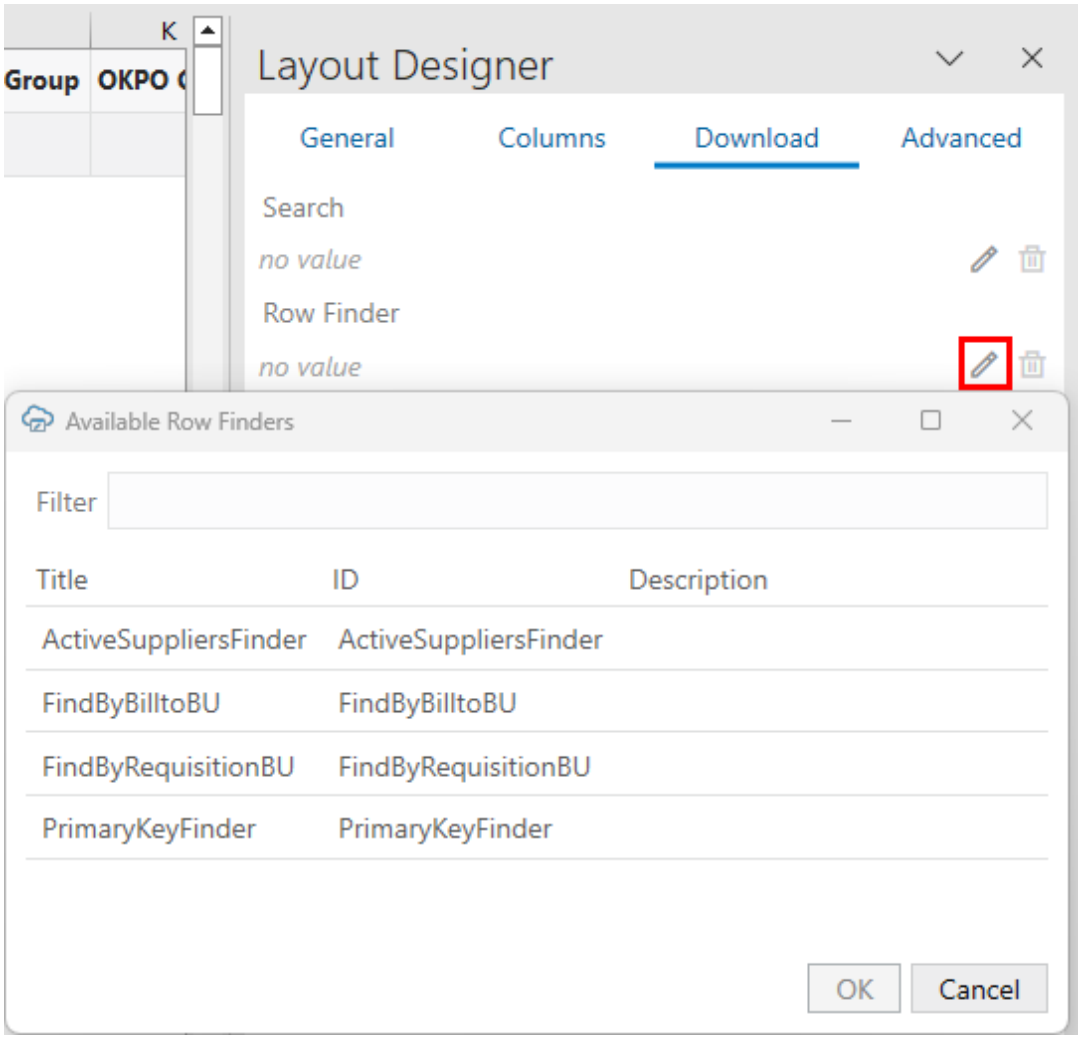
Keep these points in mind when configuring an "in" search:

- The list of search values must conform to the expectations of the service. For example, ADF REST services require that values for a string field have single quotes whereas values for an integer field must not have single quotes.
  Refer to the documentation for the target service for more information.
- The search editor does not impose any limit on the number of items you can enter. If you provide a very long list of items for an "in" search, you may run into errors. For example, the Oracle database supports a maximum of 1000 items for the "in" operator. If you run into an error, reduce the size of the list.
- For URL-based search, you may run into a problem with the total length of the URL used in the GET request for download.
- If you create an "in" search, the workbook can no longer be used with versions of the add-in prior to 4.1.

Use Row Finders to Limit Downloaded Data

For workbooks that integrate with ADF REST services, you may select one of the predefined row finders if any are associated with the layout's business object.

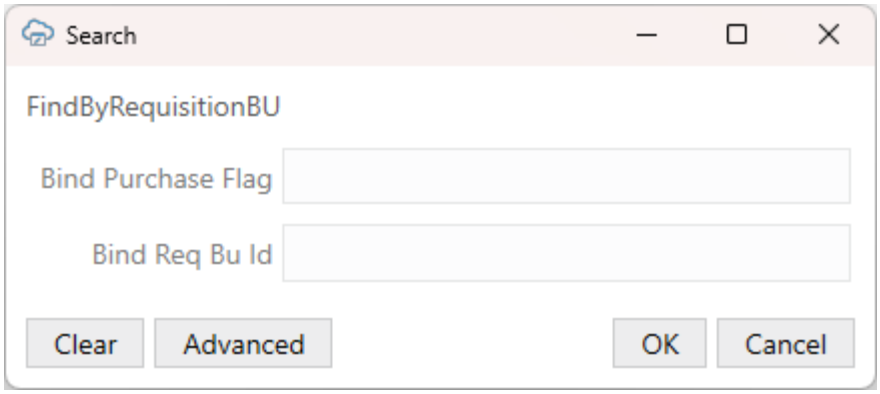
1. In the Excel ribbon, click **Designer**.
2. In the Layout Designer's **Download** tab, click the Edit icon (✎) next to the Row Finder property to see the row finders configured for the service in the Available Row Finders window:



For details on modifying how row finders appear in the add-in (including how to add titles and help text for row finders and variables), see Configure Row Finders for a Business Object.

3. Select a row finder and click **OK**.

During a download, the row finder configured here is used to filter data based on the finder's criteria. If the row finder requires input from the user, the business user is prompted to provide values in the Search prompt.



If a value is not provided, that parameter is not included in the REST request to the service.

See Download Data to the Workbook in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

For more information about configuring a Oracle ADF REST Resource service that supports finders, refer to these resources in *Developing Fusion Web Applications with Oracle Application Development Framework*:

- About RESTful Web Services and ADF Business Components
- Filtering a Resource Collection with a Row Finder

Configure Row Finders for a Business Object

If an ADF REST service includes row finders, you can view and configure them through the Finders tab of the Business Object Editor.

Row finders are predefined filters available through services that allow you to download a specific subset of records. For example, your service may include a row finder that filters expense reports by "unapproved" status and assigned to the "current user". In this case, a business user could use the row finder to download only those expense reports that require their approval. See Use Row Finders to Limit Downloaded Data.

From the Finders tab, you can open a row finder in the Row Finder Editor and configure how it appears in the add-in. For example, you can:

- Set row finder variables for either basic or advanced searches.
- Remove unwanted row finder variables so that they don't appear in the prompt at download time.
- Give row finders or row finder variables a more readable title.
- Add some help text to help your business users understand what a row finder does or what should go in a row finder variable.
- Configure a list of values for a row finder variable.
- Reorder how row finder variables appear in the Search prompt.
- Define default values for row finder variables.

> **Note:**
>
> Row finders are defined through the service. You can't add row finders or row finder variables through Oracle Visual Builder Add-in for Excel. Nor can you change properties like the finder's ID or a variable's data type on the service. Any changes you make must be compatible with the service. If your changes are incompatible, you will likely see errors during a download.

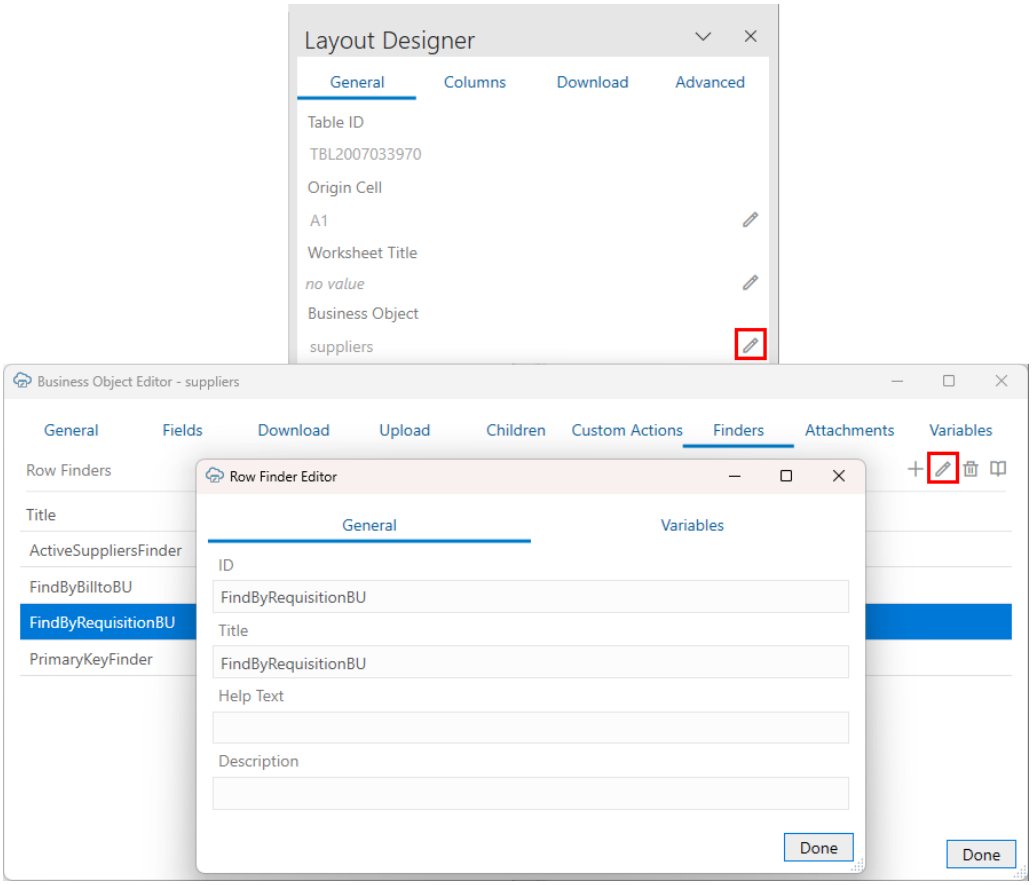
To configure row finders:

1. Navigate to the Row Finder Editor for the row finder you want to edit:
   a. In the Excel ribbon, click **Designer**.
   b. In the Layout Designer, click the Edit icon (✎) next to the Business Object.

c. Click the **Finders** tab in the Business Object Editor to view a list of available row finders. Remove row finders if required.

> **Note:**
>
> The Finders tab is only displayed for ADF REST services.

d. Select a row finder from the list and click the Edit icon ( ).



2. Edit the settings on the **General** tab as required.

> **Caution:**
>
> Take care not to change the row finder's ID.

For example, you can:

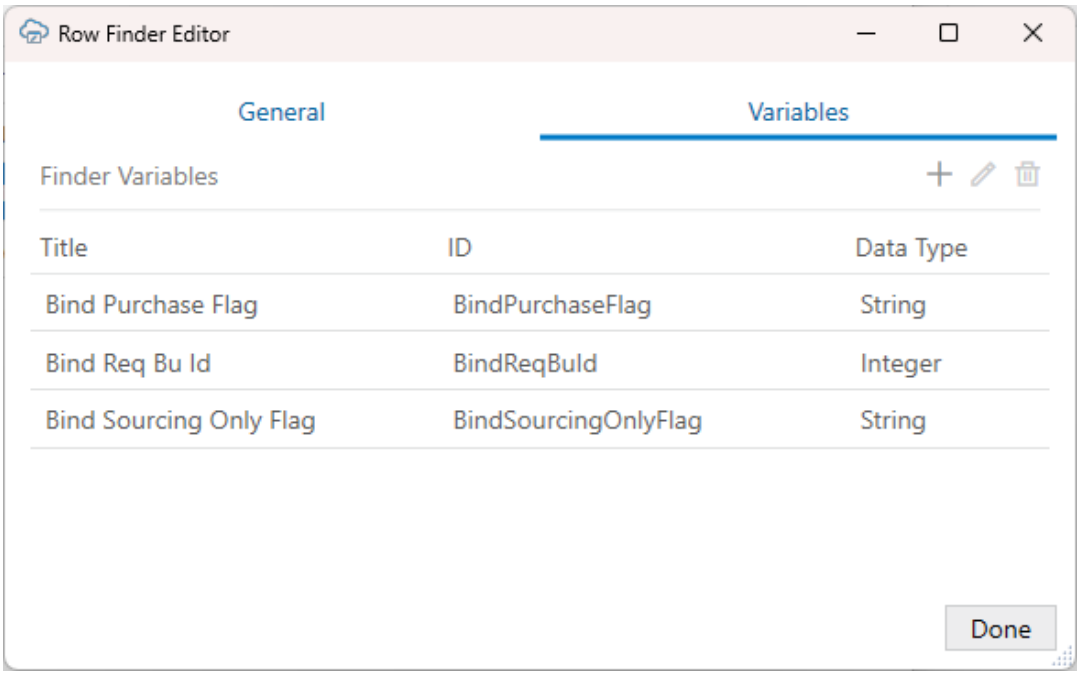
- Change the value for the **Title** property to be more readable.
- Add descriptions of the row finder using the **Help Text** and **Description** properties. The **Help Text** property is intended to give your business users more information about the row finder. This text is displayed in a popup next to the row finder title in the search prompt. The **Description** property is intended for workbook developers. This text is displayed only in the designer UI, like the Finders tab of the Business Object Editor.

You can use this property, for example, to provide technical details of the row finder to workbook developers.

> **Note:**
>
> The values for the **Title** and **Help Text** properties can be translated. See Manage Workbook Translations.

If a row finder supports variables, they'll be listed on the **Variables** tab. Variables act as arguments or parameters for the finder.

3. Click the **Variables** tab, to view a list of variables.

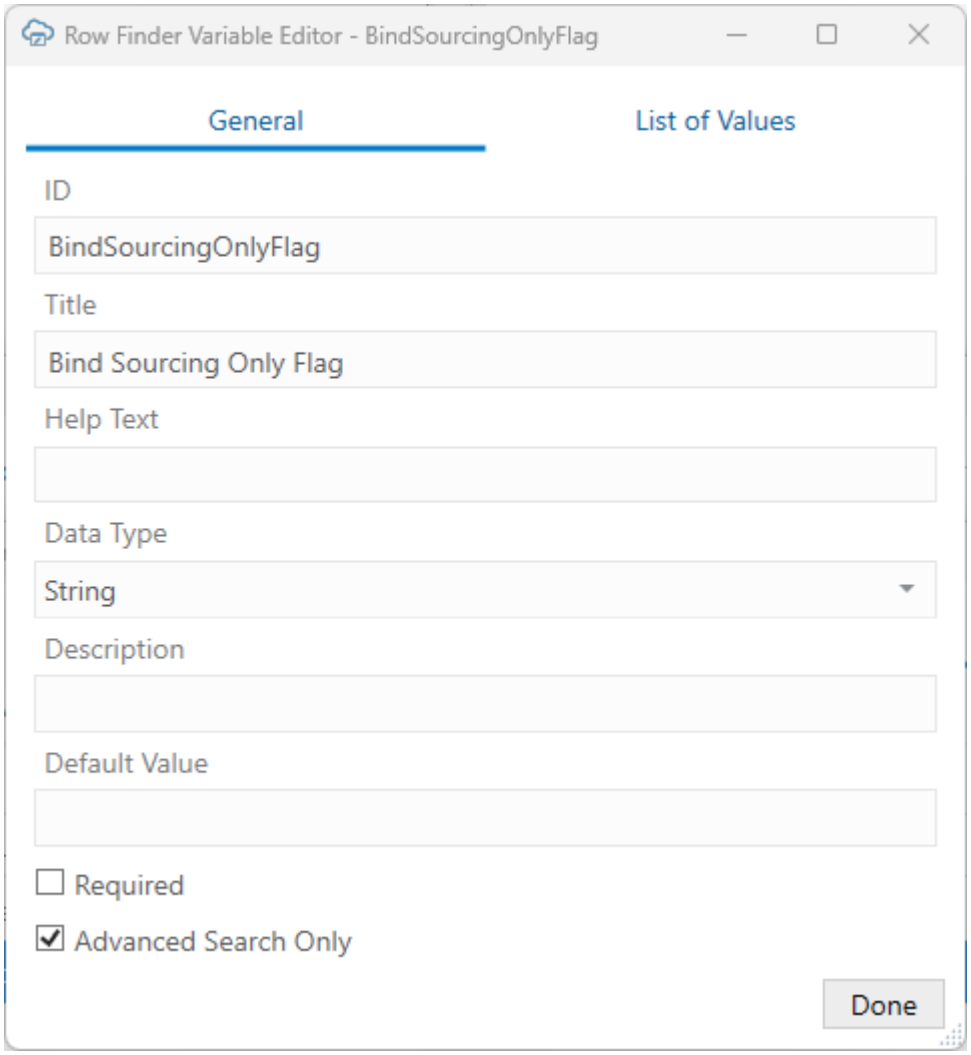


Variables are shown in the order they appear in the Search prompt. To change the order of the variables, drag a variable to another location in the list.

You may want to reorder variables if you are configuring lists of values with dynamic filters. Consider a row finder, "Employees By Country, State, and City" that includes three row finder variables in this order: "CityId", "CountryId", and "StateId". You'll want to move the CityId variable to the bottom so that the business user first selects a country, then a state, and finally a city.

> **Tip:**
>
> You can also right-click a variable and select an action (Move Up, Move Down, and so on) from the popup menu.

4. To open a row finder variable, select it from the list and click the Edit icon (✎).

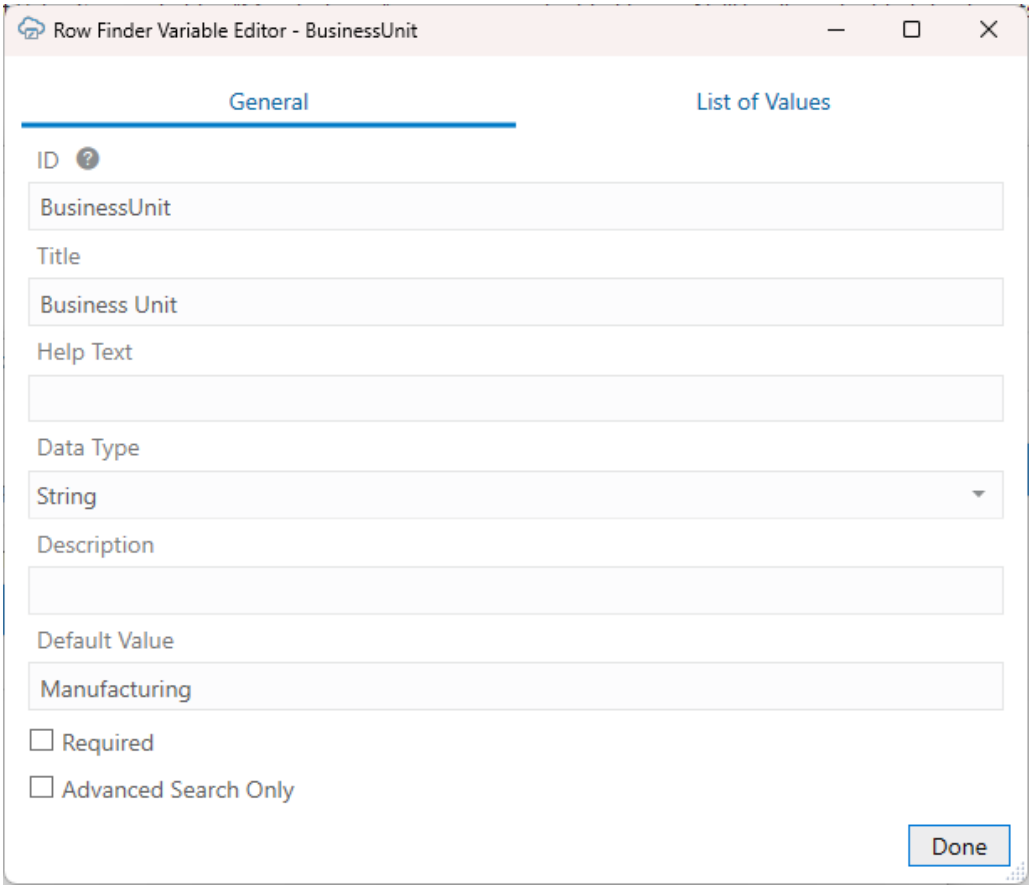


5. Edit the variable as required.

> **Caution:**
>
> Take care not to change the variable's ID.

For example, you can:

- Change the row finder variable's title.
- Use the **Help Text** and **Description** properties to provide useful descriptions of the variable. The **Help Text** value is displayed as a popup next to the row finder variable in the search prompt.
- Set a default value for the row finder variable. Let's say you have a workbook that needs to show expense reports based on the business unit using it, such as "Manufacturing". If the service has a row finder, "FindByBusinessUnit", and that finder includes a variable, "BusinessUnit", you can set the default value of the variable to "Manufacturing".



When a business user downloads data, the Search prompt displays the default value, "Manufacturing", for the Business Unit variable.

Default values are not supported if a list of values is configured for the row finder variable.

Default values can also use expressions such as `Today ()` for Date (no time) fields. See Use Expressions in an Integrated Workbook.

> **Note:**
>
> Reserved words are not supported with one exception. You can use expressions that reference workbook parameter values in the default value for row finder variables. See Use a Workbook Parameter Value for a Row Finder Variable.

- Select the **Required** check box to force the business user to provide a value at download.
- Select the **Advanced Search Only** check box to include this variable only in advanced searches. Deselect this check box to include this variable in the basic search. During a download, the add-in opens a search prompt that includes only basic search variables. If a business user wants to perform a more refined search, they can click the **Advanced** button to display both the basic and the advanced variables.

  If the business user proceeds with a basic search, the advanced variables are not included in the download request.

> **Note:**
>
> When editing a row finder's variables, make sure to leave at least one basic variable to display in the search prompt.

> **Note:**
>
> The values for the **Help Text** and **Title** properties can be translated. See Manage Workbook Translations.

6. If required, configure a list of values for a row finder variable from the List of Values page in the Row Finder Variable Editor. See Configure a List of Values with a Business Object.
7. Click **Done** until you return to the Layout Designer.

Once you are done making changes, the row finders become available to you as an option to limit downloaded data, as described in Use Row Finders to Limit Downloaded Data.

Use a Workbook Parameter Value for a Row Finder Variable

You can use an expression that references a workbook parameter to determine the default value for a row finder variable. This way, you can use a value stored in a workbook to control which rows are returned by a row finder during data download.

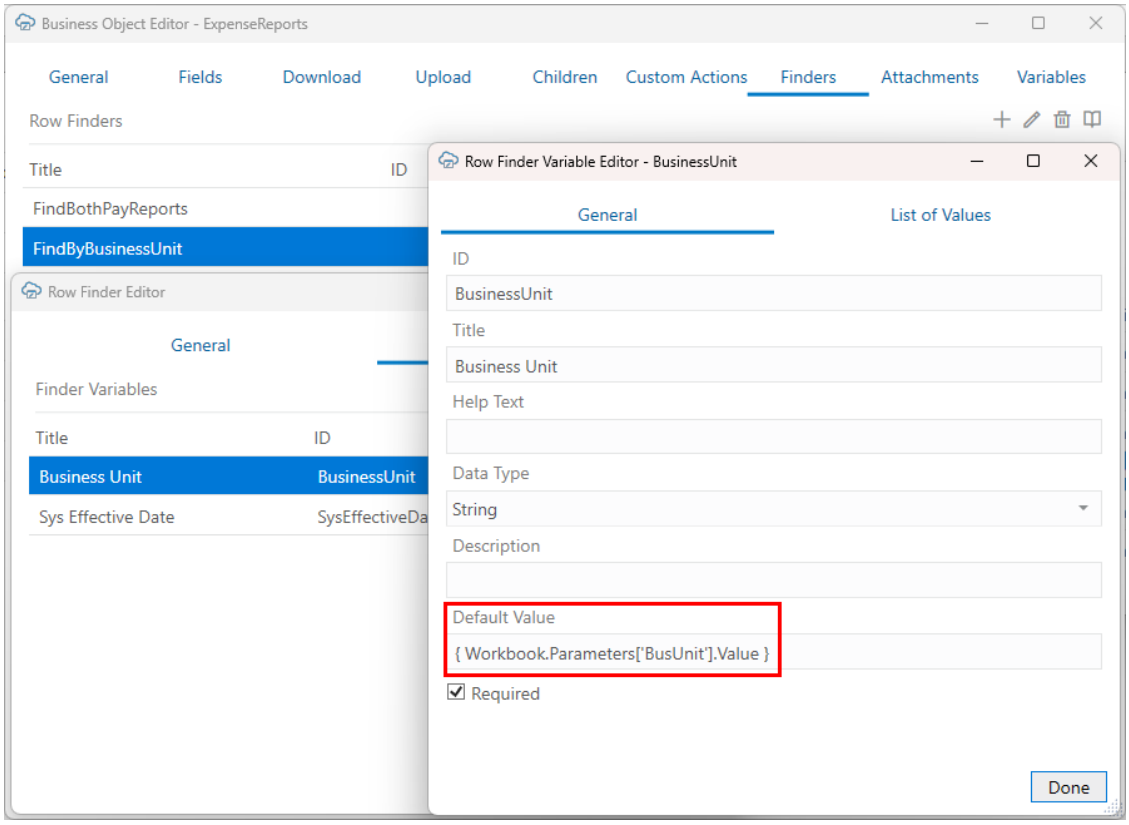
Let's say the service has a row finder, "FindByBusinessUnit", with a variable, "BusinessUnit". Instead of setting the variable's default value to a static value—like "Manufacturing"—you may want to write the user's business unit to a workbook parameter and use an expression that references this parameter.

When a business user downloads data, they are prompted to provide the business unit for the row finder, which then returns expense reports matching this value.

If you set the default value dynamically based on where the workbook is retrieved from, you would need to embed the value as a workbook parameter while it is being retrieved from the web application. Workbook parameters are name-value pairs that are embedded in your workbook programmatically, typically by a web developer. For help writing values to workbook parameters, see Embedding Workbook Parameters in a Workbook.

To use a workbook parameter value as the default value, enter an expression like this: `{ Workbook.Parameters['<para_name>'].Value }` in the **Default Value** field of the Row Finder Variable Editor. See Workbook Parameters in Expressions.

In this example, the expression references a workbook parameter, `BusUnit` which stores the name of the business unit.



Oracle Visual Builder Add-in for Excel does not validate expressions in the Row Finder Variable Editor. You should test your expression before distributing the workbook.

Keep in mind the following when testing:

- If the expression result cannot be successfully converted to the appropriate data type for the row finder variable, the row finder variable value will be blank/empty. There is no error message.
- Workbook parameters do not have a data type associated with them. Expressions that use them treat them as string data values.
  There are no conversions for different cultures within the expression.
- If you change the default value in the editor after downloading data, be sure to clear the layout to test the new default value.
- The expected format of the workbook parameter values is "invariant":
  - Dates: ISO 8601 "1992-06-15"
  - Numbers: decimal point; "3.14"
  - Date-time: ISO 8601 UTC: "2017-04-19T12:33:19Z"
  - Boolean: "True" / "False" (not localized)

After you complete your testing and distribute the workbook, your business users are presented with a prompt showing the value stored in the `BusUnit` workbook parameter when downloading data.

Use Download Parameters to Limit Downloaded Data

You can configure additional parameters to add to the query string at the time of download. These download parameters help limit the data that gets returned from the REST service when a business user invokes a download. For example, if a business user only needs to access records for a specific location, you can create a parameter that filters for that location.
You can configure parameters for each layout in a set of dependent layouts. See Filter Data for a Set of Dependent Layouts.

> **Note:**
>
> The specific GET URL query string syntax is governed by the REST service. Consult with your service owner if required.

How Download Parameters Work

Let's consider a couple of examples. Let's suppose you want to retrieve employee rows where the `managerId` field is empty. In this case, you would create a search on the **Employees** business object with a parameter of "managerId is null". The GET request would then include the following in the query string portion of the URL:

```
GET…/Employees?q=managerId+is+null
```

where:

- `q` is a query parameter supported by the Employees service
- `managerId` is the name of a field that supports the query parameter
- `null` is the search value

Download parameters can use expressions to filter data. For example, you could use the `Today()` date function along with an integer (`{ Today() - 90 }`) to find employees with hire dates in the last 90 days. See About Expressions.

An expression can reference a workbook parameter embedded in the workbook. A parameter for a 'q' query that references workbook parameters might look like this:

```
DepartmentId={ Workbook.Parameters['Dept'].Value } AND Salary >=
{ Workbook.Parameters['MinSal'].Value }
```

This parameter returns employee rows where the department Id matches the value of the `Dept` workbook parameter and the salary is greater than or equal to the value of `MinSal`.

During download, Oracle Visual Builder Add-in for Excel constructs the GET URL by resolving the expressions in the Parameter Value property. If the values of the `Dept` and `MinSal` parameters are "80" and "7000" respectively, then the resulting GET request URL would look like this:

```
.../Employees?q=DepartmentId=80 AND Salary >=7000
```

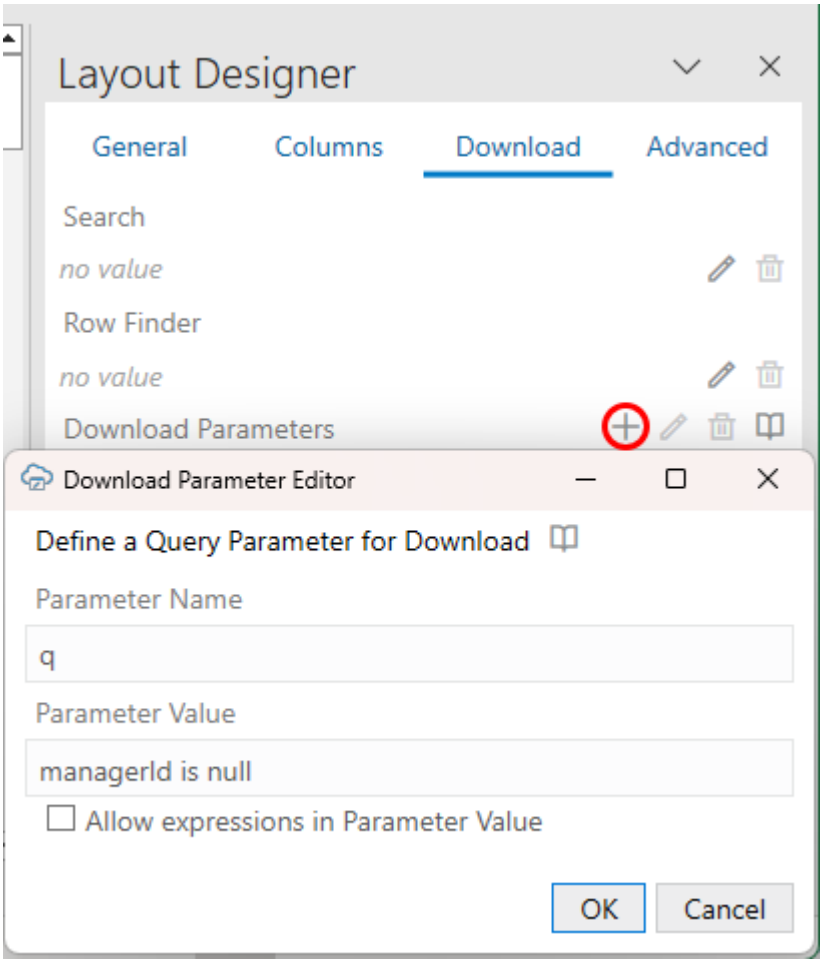
See Use Workbook Parameters for Download.

If the service supports complex searches, you can create complex searches in the layout, as shown in the following example:

```
q=((firstName LIKE '*es*') or ((hireDate< "2001-01-13") and (department = 10)))
```

To add download parameters to your layout:

1. In the Excel ribbon, click **Designer**.
2. In the Layout Designer's **Download** tab, click the Add or Edit Download Parameter icon next to the Download Parameters property to open the Download Parameter Editor where you add or edit download parameters.



3. Enter a name and value in the parameter properties.

> **Note:**
>
> Do not enter URL-encoded values in the parameter properties. The add-in applies URL encoding to the parameter value at download time. The parameter name is not encoded.

4. If you used an expression in the **Parameter Value** property, select **Allow expressions in Parameter Value**.
   Selecting this check box ensures the add-in evaluates the value as an expression.
5. Click **OK** to close the editor.
6. Test the parameter by downloading data and viewing the results.

> **Note:**
>
> There is no validation in this editor or at download time. The add-in cannot determine which parameters are useful for search. Likewise, the add-in can't determine the proper syntax for the parameter values. If you enter invalid information, you may get a bad request error. Consult the API documentation for the service you are using to identify whether to use "q" for the parameter name and how to formulate the search expression properly.

When a parameter is configured, the add-in appends the search query to the REST endpoint URL of the GET request when the business user clicks the **Download Data** button.

The Download Parameters property works in combination with the other two properties (where applicable). They are not mutually exclusive. If search or row finder settings have also been configured, the user is prompted to provide values for the configured parameters that will further filter the returned results.

> **WARNING:**
>
> If you configure multiple search options, make sure the service supports that combination. Some combinations may work where others may not.
>
> Some services, such as ADF REST and VBBO services, do not support using more than one "q" parameter in a REST call. If you are using one of these services, configure the add-in to merge all configured "q"-type queries into a single query for download. See Merge Searches for Download.

> **Tip:**
>
> To troubleshoot a particular combination of search settings, open the Network Monitor and download data. Then, inspect request details in the Network Monitor's window. This information may help you refine search settings.

Merge Searches for Download

If you have configured more than one 'q'-type query for a layout, Oracle Visual Builder Add-in for Excel merges all 'q'-type queries configured for a layout into a single query for download.

Let's take a look how the add-in constructs the URL for the GET request. Suppose you want to download rows for employees in the Sales department who were hired after June 22, 2022 and have no manager assigned. In this case, you could configure a search in the Search Editor for the Sales department and hire date and then a download parameter using the Download Parameter Editor for the manager. Both the search you configure through the Search Editor as well as those configured through the Download Parameter Editor use 'q' parameters.

During download, the URL for the GET request includes a single 'q'-type query using the AND operator. like this (URL parameters are shown without encoding for clarity):

```
. . ./employees?q=HireDate > '2022-06-22' AND Department='Sales' AND Manager
IS NULL&offset=0& ...
```

The add-in merges any download parameters where the Parameter Name value matches the 'q' parameter name using a case-sensitive comparison. In this case, this query returns rows for employees in Sales hired after June 22, 2022 who have no manager assigned.

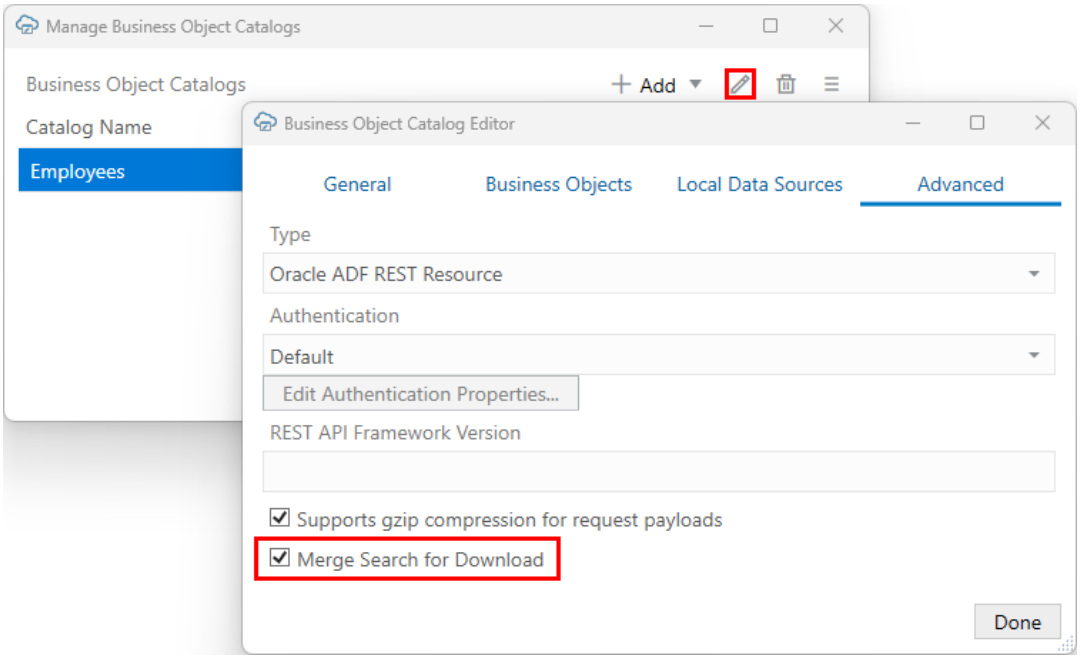
For more information about search and download parameters, see Use the Search Editor to Find Required Data and Use Download Parameters to Limit Downloaded Data.

By default, workbooks created before 4.0 use separate 'q'-type filter queries instead. This may not provide the expected results since some services, such as ADF REST and VBBO services, ignore all but the first 'q' sent.

If the service ignores the second and third 'q' parameters, all rows with hire dates after June 22, 2022 are downloaded which is not the intended result.

To ensure all the queries are used to filter the results, configure your older workbooks to use the merge option:

1. Click **Manage Catalogs**.
2. In the **Manage Business Object Catalogs** window, select your existing catalog and click the **Edit Business Object Catalog** icon ( ).
3. In the **Business Object Catalog Editor**, click **Advanced**.



The **Merge Search for Download** check box is selected by default for new catalogs.

4. To merge searches on download, make sure **Merge Search for Download** is selected.

> **Note:**
>
> Workbooks configured to use this feature may no longer be compatible with add-in versions earlier than 4.0.

Use Workbook Parameters for Download

You can use workbook parameters stored in your workbook to control which rows are retrieved during download. Workbook parameters are name-value pairs that are embedded in your workbook programmatically, typically by a web developer.
An organization may choose to embed different values in an integrated workbook—for example, when it is retrieved—based on the context. Let's consider an example. Suppose your organization has a web app that includes an Employees page and that this page includes a button for downloading an integrated workbook. The web developer could use a server-side mechanism to write the same filter values to the workbook used by the business user in the web app.

For example, a business user may filter the employees list in the web app to show only employees in the Sales department with a salary greater than or equal to $7000. If this user then clicks the download button on the page, the web server embeds the filter values—`Dept=Sales` and `MinSal=7000`—in the integrated workbook before sending it to the user.

If you configure a search, a row finder, or a download parameter that references these workbook parameters, then the user sees the same data as in the web page when they open the downloaded workbook and click **Download Data**. For more information about using expressions that reference workbook parameters, see Workbook Parameters in Expressions.

How these workbook parameters are written to the workbook is up to you. However, if your organization plans to take advantage of this technology, your web developers will need to know where to write these workbook parameters and in what format. You can point them to Embedding Workbook Parameters in a Workbook for these details.

See also:

- Use the Search Editor to Find Required Data
- Use Row Finders to Limit Downloaded Data
- Use Download Parameters to Limit Downloaded Data

Test a Download with Workbook Parameters

Before you distribute the workbook, it is recommended that you test the search behavior of a configured search, row finder, or download parameter that references a workbook parameter.

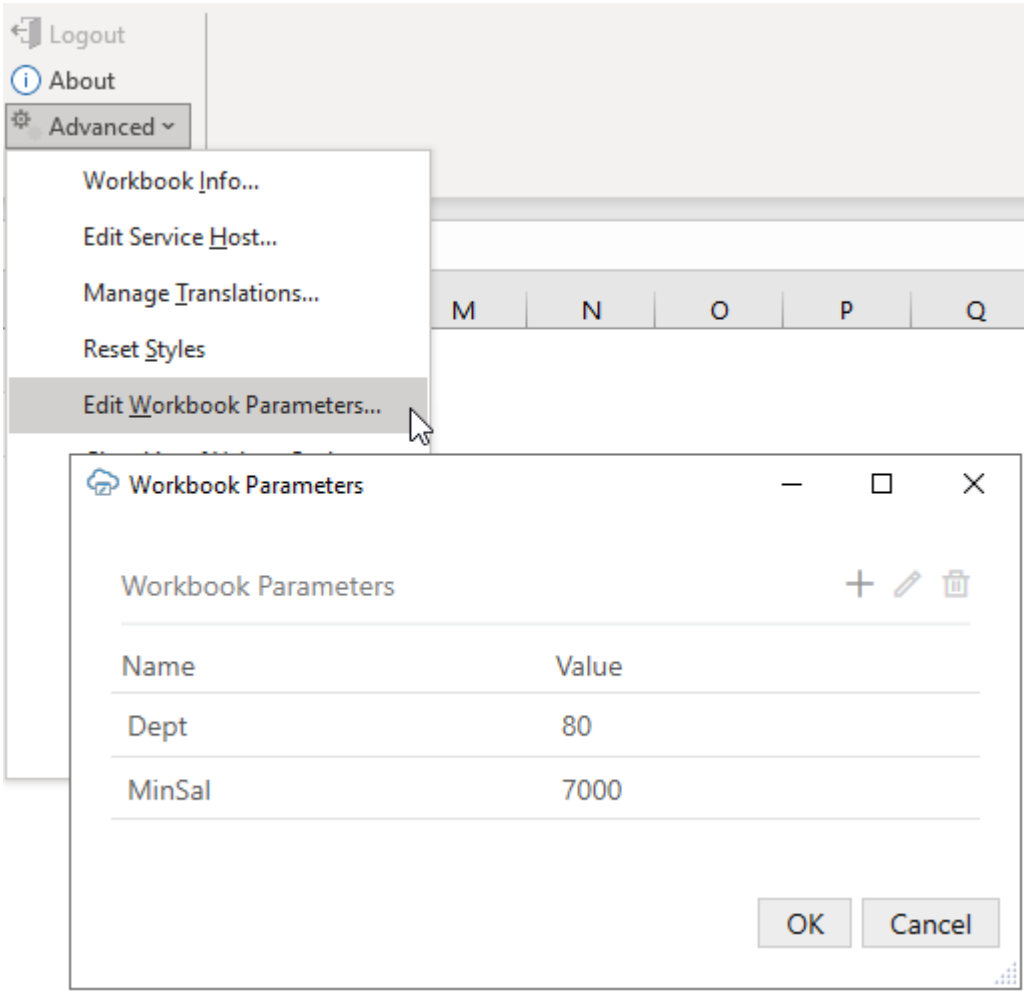
Use the Network Monitor to review the requests and responses to and from the REST service during a download. The Network Monitor shows the URLs that the add-in constructs using the search or row finder settings. See Network Monitor.

You can also view or modify workbook parameters while testing using the Workbook Parameters dialog.

> **Note:**
>
> The Workbook Parameters dialog should only be used for testing purposes. Any parameters added or edited using this dialog are removed when the workbook is published.

To test the search behavior:

1. Open the Network Monitor from the Oracle Visual Builder ribbon, then click the **Download Data** button.
2. Check the results of the download in the workbook. If you experience issues, review the requests and responses in the Network Monitor.
3. When troubleshooting issues, try these steps:
   - Check the syntax of the expression in the Search Editor, Row Finder Variable Editor, or Download Parameter Editor. Anything from a typo in the parameter name to a missing bracket may be causing the error. See About Expressions.
   - Modify the value of the workbook parameter and then click **Download Data** again. The next download will reflect any changes made to these parameters.
     Open the Workbook Parameters dialog from the **Advanced** menu.



Note:

The **Edit Workbook Parameters** menu command is only available when the designer tools are enabled.

To change a value, select the workbook parameter from the list and click the Edit icon (✎).

Embedding Workbook Parameters in a Workbook

Write workbook parameters to a workbook using a custom solution such as a J2EE servlet or browser-based JavaScript. Workbook parameters are name-value pairs that are stored in a hidden worksheet of the workbook using a packed string-value format. Workbook developers can create searches that reference these parameters.
Workbook parameters might be name-value pairs that correspond to URL arguments passed when the workbook is downloaded from a web app. They may also represent some state from a web app page. Or they might be arbitrary named values.

Oracle Visual Builder Add-in for Excel does not include functionality for embedding workbook parameters programmatically. An organization that wants to use this functionality must provide their own custom solution for embedding these parameters in a given workbook. This topic is intended for the programmer or web developer tasked with coding this solution.

Workbook parameters must be written to a specific location and in a specific format. Before you proceed, review the parameter requirements carefully to make sure you embed your workbook parameters correctly.

Sample Methods

Here are some possible methods to consider when planning a solution:

- A server-side mechanism using a J2EE servlet or filter to examine and update the workbook during download
- A client-side/browser-based (JavaScript) mechanism that can examine and update the contents of the workbook after it is retrieved from the server and before it is written to the local computer file system

Location

All workbook parameters are stored as a string on a hidden worksheet named `_VBCS_WorkbookInfo` at cell address `$B$15`.

Format

The format for the packed string is:

<paramName1>=*<paramValue1>*/*<paramName2>*=*<paramValue2>*/...

with these conditions:

- Each *paramName* and *paramValue* must be separately URL-encoded.
- Each name must be separated from its value using the equals character (=).
- When there are multiple name and value pairs in the string, each name value pair is separated from others by the forward slash (/) character.
- The equals (=) and forward slash (/) characters must not be encoded.

Notes

- The total size for the packed string format for the set of embedded parameters is limited to 32,759 characters.
- The specific worksheet cell (`_VBCS_WorkbookInfo!$B$15`) may:

  - Be empty (no parameters)
  - Contain a packed string of 1 or more parameter name and value pairs
  - Contain the special placeholder string, `$$VbafeWorkbookParameters$$`
    This value is set by default when a workbook is first integrated using the add-in, and also when the workbook is published.

    The add-in treats this value, if found, the same as an empty string (no parameters available)

- Workbook parameters are removed when the workbook is published. This applies to all workbook parameters embedded as described here or created using the Edit Workbook Parameters dialog.

7

Download Data

When a business user clicks **Download Data** from the Oracle Visual Builder tab, Oracle Visual Builder Add-in for Excel fetches data from the REST service and displays that data in the integrated workbook's layouts. This action generally involves one or more GET requests on the business object's collection path.

How the add-in manages a download depends on the type of layout, whether there are dependent layouts, and the add-in's download settings. For example, you can configure the add-in to retrieve descendant rows in a single payload. See Configure Download to Use a Single Payload.

To gain a better understanding of how the add-in behaves during a download, refer to the appropriate section:

- Table Download
- Form-over-Table Download
- Dependent Layout Download
- Notes on Download Behavior

> **Note:**
>
> The **Download Data** icon is disabled if the **Download Enabled** check box is not selected in the Layout Designer. See Manage Layout Capabilities. It may also be disabled if the GET operation is not configured on the collection path for the business object. See REST Operations.

Table Download

When a business user performs a download for a Table layout, Oracle Visual Builder Add-in for Excel performs a number of actions including clearing existing data from the table, sending a GET request on the business object collection path, processing the response, and writing values to the worksheet.

Here is a detailed breakdown of what happens when the business user clicks **Download Data** for a Table layout:

1. If the business user is not logged in, they are prompted to log in.
2. If polymorphic business objects are used in the layout(s), then the polymorphic metadata is refreshed.
3. The add-in may display one or more prompts in this order. If any of the prompts are canceled, the download operation is terminated.
    a. If the table has any pending changes, a warning is displayed and the business user can cancel to avoid losing changes.
    b. If path parameter values are required, the business user is prompted for these.

c. If a row finder is configured and the finder includes row finder variables, the business user is prompted to provide values for the variables.

d. If a search is configured, the business user is prompted to provide values for the search conditions.

4. When the final prompt has been accepted, a progress window is displayed. The business user may cancel the download at any time.

5. During download:

   a. If there are any worksheet filters, these are cleared.

   b. All existing data is cleared from the table.

   c. Table, including headers, is fully redrawn.

   d. All polymorphic field sets are expanded at this time to include the global segments, discriminator (context segment), and context-sensitive segments.

6. List of values (LOV) columns are initialized.
   If a given LOV already has choices cached, that cache is used at this point. If a given LOV has no cache, a REST request is sent for the first 300 choices.

7. The first page of items is requested from the REST service.
   The add-in issues a GET request on the business object collection path. This request includes various query parameters and request headers.

8. The add-in uses a single background thread to request the second page of items while processing the response from the first page.

9. For each row received, the add-in:

   a. Prepares the values to be written to the worksheet.

   b. Determines the appropriate cell styling.

   c. For LOVs, the identity values are exchanged for display values using the cache. Missing identity values are captured for later.

10. If there are more rows to download after the first page is processed, the add-in prompts the business user to continue.
    If the business user clicks **Stop Now**, the download operation stops immediately. If they click **Download All**, the operation continues to retrieve additional pages until no more pages are available.

11. For each page of rows, the add-in:

    a. Issues additional GET requests on the LOV data source for LOV cache misses, and completes the swap of identity values for display values.

    b. Writes blocks of values to the worksheet.

    c. Applies appropriate styling to blocks of cells.

12. The add-in continues to fetch page after page until there are no more items to fetch.

13. The columns are resized to fit the current content.

14. The table row height is reset to the add-in's standard height.

15. If the table is not visible, the upper-left corner is selected to make it visible.

16. The post-download macro is invoked, if configured.

During the download operation:

- The add-in populates a special column called the "Key" column. The content of the cells in the key column is not human-readable. This column contains "housekeeping" information

for each row. This information is essential to the proper working of the table features. You should never edit or attempt to remove the key column. If you sort the table, you must include the key column in the range of sorted cells. Otherwise, you may get data corruption.

- The status viewer is updated at various stages.

> **Note:**
>
> Problems communicating with the REST service may interrupt the download process. You can use the Network Monitor to see details of each request and response. See Network Monitor.

Form-over-Table Download

When a business user invokes a download for a Form-over-Table layout, Oracle Visual Builder Add-in for Excel downloads the form data first, then downloads the child rows for the current form row.

The download process is similar to the table download with these differences:

- For the form download:
  - The add-in sets the count/limit query parameter to "1" (one) since only one row is displayed in the form.
  - There is no pagination at the form level.
  - The add-in populates the form region with data from the first row in the response from the GET request on the collection path.
- For the table download, the path parameters for the child collection path are managed automatically by the add-in based on the current parent row.
- The post-download macro is invoked, if configured, when the form and table downloads have completed successfully.

Dependent Layout Download

When a business user invokes a download for a set of dependent layouts, Oracle Visual Builder Add-in for Excel starts with the primary layout and then progresses through the hierarchy.

The download process for a set of dependent layouts is similar to the other processes with these differences:

- When a download is invoked on any layout in a set of dependent layouts, the download process applies to all the layouts in the set.
- A download always starts with the primary layout even if a different worksheet is currently active.
- The check for pending changes applies to all layouts in the hierarchy.
- The entire hierarchy is cleared of current data.
- Query prompts are only available for the primary layout.

- Additional download parameters, if configured, are applied at each respective level. See Filter Data for a Set of Dependent Layouts.
- The prompt to continue after the first page appears once only for the primary layout. If the parent layout is a Form-over-Table layout, this prompt applies to the child table. There is no prompt for descendant layouts.
- Each layout in the hierarchy uses the specific pagination settings, such as Limit and Offset, from their respective business objects. See Configure Pagination for a Business Object.

Descendant Table Layout Download

For descendant table layouts, the rows the add-in downloads depend on the rows in the direct parent table layout. For each row in the direct parent table layout, the add-in retrieves and processes all relevant child rows. If there are download parameters defined on the descendant table layout, the search criteria may limit the number of child rows retrieved for each parent row. See Filter Data for a Set of Dependent Layouts.

By default, the add-in sends separate GET requests to retrieve descendant rows. So to retrieve child rows, the add-in starts by making a single GET request to retrieve the first page of child rows for a given parent.

Writing the child rows to the worksheet takes place in order: all child rows for parent row 1 are written, then the rows for parent 2, and so on.

Much like the standard download, the add-in processes the retrieved rows for the page, requests the next page in the background, and writes the page's rows to the worksheet. It keeps iterating over pages until all children for the parent row are retrieved and processed.

> **Note:**
>
> Up to 4 GET requests may be made on background threads, in parallel.

Instead of downloading with separate requests, you can configure the add-in download descendant rows in a single payload. See Configure Download to Use a Single Payload. In this scenario:

1. The add-in makes one or more GET requests to retrieve the business object items for the table rows.
   The number of parent rows per request is determined by the value of the **Limit Parameter Value** field on the Download tab of the Business Object Editor. See Configure Pagination for a Business Object.

   The GET request is configured to include all descendant rows.
2. The service sends a response payload that includes the set of parent rows for the table as well as all descendant items for those parent rows.
3. The add-in writes the returned parent items data into the table and captures the descendant rows.
4. The add-in repeats these steps until all parent items have been retrieved and written into the table.
5. The add-in writes the descendant rows for or each descendant layout in the hierarchy to the respective table without making any more REST requests.

Configure Download to Use a Single Payload

If you have a set of dependent layouts, you can configure Oracle Visual Builder Add-in for Excel to download all relevant descendant rows in a single payload. Use of a single payload can significantly improve performance.

This feature is only available for ADF REST services.

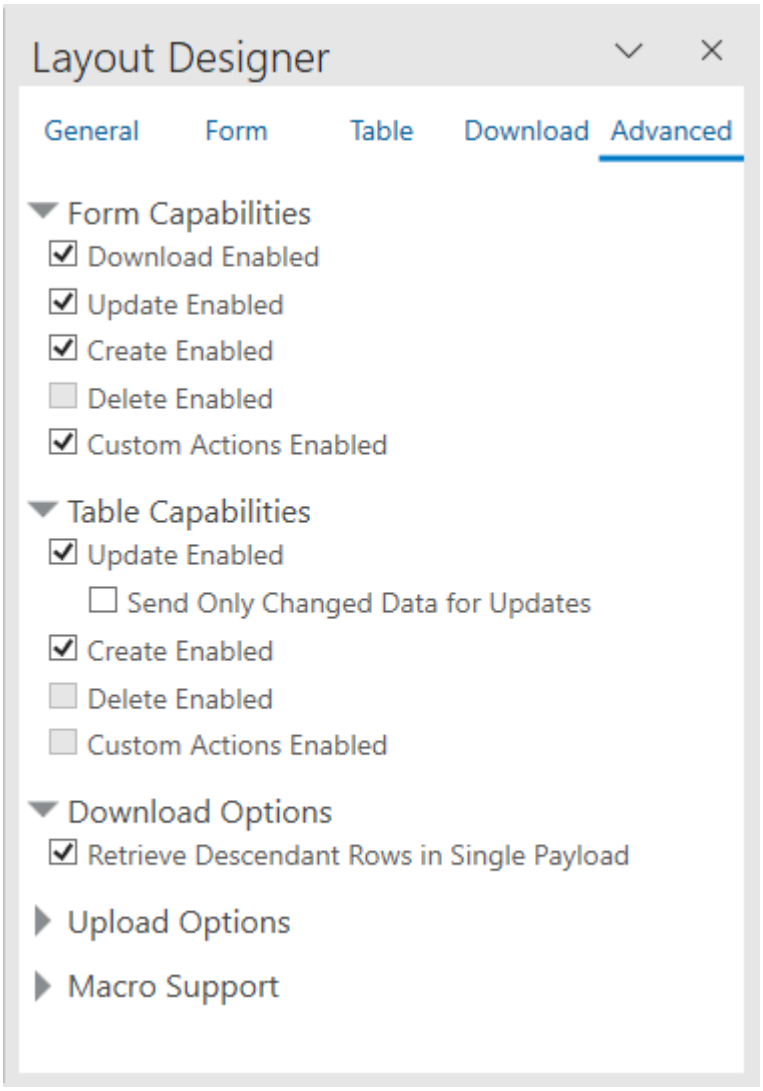
If your primary layout is a Form-over-Table layout, the add-in first makes a single GET request to retrieve field values for the form part of the layout. It then makes requests to retrieve blocks of rows for the table part of the layout as well as all relevant rows for the dependent layouts.

If the primary layout is a Table layout, the add-in makes one or more GET requests to retrieve the business object items for the table rows. All descendent items for those parent rows are also included in the response payload.

> **Note:**
>
> This feature will not scale to handle large, enterprise-level volumes. If you enable this feature, test the download behavior to ensure that all required use cases will succeed in all customer environments and scenarios. Review Notes and Limitations of Single Payload Downloads for more information about scalability.

To enable this feature:

1. Select the primary layout, then click **Designer** from the Oracle Visual Builder tab to open the Layout Designer.
2. Click the **Advanced** tab in the Layout Designer.
3. From the **Advanced** tab, expand **Download Options**, then select **Retrieve Descendant Rows in Single Payload**.



Notes and Limitations of Single Payload Downloads

Single payload downloads (also referred to as "nested" downloads) are only supported for ADF REST services. Review this section to see if the feature is suitable for your integrated workbook.

Scalability

Be aware that this feature will not scale to handle large, enterprise-level volumes. If you enable this feature for your workbook, you are responsible for any scalability issues that arise. Make sure to test your workbook to verify that all required use cases will succeed in all customer environments and scenarios.

Use of CPU, memory, and network bandwidth resources could exceed acceptable levels, depending on the business object hierarchy and the data volume at each level in the hierarchy. Potential issues include:

- Complex business logic or the processing of large numbers of nested descendant items could push the request duration beyond what some load balancers and other network components, such as Akamai, support. This could result in the request failing due to a timeout.

- Large response payload sizes will likely consume large amounts of memory and CPU. This could potentially affect other clients working against the same endpoint, virtual machine, or container. It could also affect Excel and other programs running on the business user's computer.

Limitations

- ADF REST services only
- This feature is not compatible with the use of the `fields` search parameter.
- The add-in uses the `expand` query parameter on the GET request for the top-most business object. The GET request includes all the descendant business objects (child, grandchild, and so on), by name for the `expand` query parameter value. The REST service implementation must support this.
- In the current implementation, there are limitations on the number of descendants retrieved for a given parent row. The limit is imposed by the ADF REST framework for a given GET on the parent collection that uses the `expand` query parameter to include descendant items in the response payload.
  In cases where the number of descendants for a given parent item exceeds the limit (`hasMore=true`), the add-in displays a message in the Status Viewer that not all rows were downloaded for the affected worksheet.

Notes on Download Behavior

Review these notes for more on Oracle Visual Builder Add-in for Excel download behavior:

- NetSuite services do not return full row details when responding to a GET request on the collection path. As a result, the add-in must issue individual GET requests on the item path for each item returned from the collection path.
- For ADF REST and VBBO services, the add-in sends additional GET requests to the LOV data source to get rows with missing identity values only. For other service types, the add-in may get all rows in the LOV data source to fix the cache miss.
  If the add-in still can't find the display value for an identity value, the identity value is written to the worksheet without the swap.
- REST services are stateless. As a result there are several issues to consider:
  - The query is re-executed for each page request
  - REST services cannot guarantee "read consistency" across multiple requests. If the data is changed during a multi-page download, it is possible for some rows to be missed entirely or downloaded more than once.

> **Note:**
>
> If pagination is not configured for a given business object, the add-in attempts to fetch and process all available rows in a single request. For larger volumes, such requests may time out.

- If **Send Only Changed Data During Upload** is enabled, the add-in caches a copy of the data during download to be used for comparisons later on during upload.

- The add-in harvests the self-links from the download response and uses them for subsequent item-level operations. For example:

```
"links": [
       {
        "rel": "self",
        "href": "http://myhost/myapp/rest/v1/Employees/101"
       }
     ]
```

For more information how some features affect the add-in's download behavior, see:

- Configure Search Options for Download
- Configure Pagination for a Business Object
- Work with Service Path Parameters in a Table Layout
- Use Polymorphic Business Objects and Fields
- Use Lists of Values in an Excel Workbook
- Use Macros in an Integrated Excel Workbook
- Use Multiple Layouts for Multi-level Business Objects

8
Custom Actions

Oracle Visual Builder Add-in for Excel supports custom actions defined by ADF REST services. A custom action is a function defined by the REST service that is callable by external systems using REST POST requests.

If a given REST API supports custom actions, they are described in the service's OpenAPI service metadata. The add-in can analyze this service metadata, harvest custom action definitions, and include them in the workbook's corresponding business objects. No further configuration is required. However, you may want to change the title and add help text for a better user experience. See Custom Action Configuration.

Types of Custom Actions

The add-in supports two types of custom actions: **business object (BO) level** and **BO item level** custom actions.

A BO level custom action is defined at the BO level and can operate on one or more rows in a layout. Such a custom action could, for example, reprocess all the project budget versions which are showing as failed and bring them back into working state. For more information about this particular custom action, see Reprocess all failed project budgets in *REST API for Oracle Fusion Cloud Project Management*.

A BO item level custom action operates on only one row in the layout. Examples would be a couple of custom actions, "Approve" and "Reject", that operate on the items of a expense report business object. A business user can use these custom actions to approve or reject each expense report individually before updating them all in a single upload.

These custom actions may require input from the user. If so, the business user is prompted to provide values for any custom action parameters. For example, the "Reject" custom action may require the business user to provide a reason code when rejecting an expense report. Likewise, a "Send to Auditor" BO level custom action may require the business user to provide a severity code and auditor email.

If a BO item level custom action includes fields, called **payload fields**, for capturing input from the business user, make sure these are added to your Table layout or the table part of your Form-over-Table layout. See Add Custom Action Fields to a Table Layout.

There are BO item level custom actions that report row status information. These are referred to as **row status custom actions**. See Row Status Custom Actions.

Performing Custom Actions

Business users perform custom actions using the buttons and menu commands from the Oracle Visual Builder ribbon tab. See Perform Custom Actions in Table and Form-over-Table Layouts in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

When a custom action is performed, the add-in sends a POST on the collection or item path that includes the action name, such as `/contextRoot/v1/myBusObj/{itemId}/action/doMyAction` for a BO item level custom action.

This post can include payload field values, known as "parameters", to the service. The service takes the passed parameters and performs the custom action. This action may act on a single row when defined at the row level. Or it may act on one or more rows when defined at the

business object level. In this case, the REST service determines which rows, if any are affected.

A custom action may accept zero or more parameters of simple types such as a String or Integer and returns a single value to the add-in, which is then displayed in the Status Viewer. This result returned by the service may be a simple scalar value like a string or a more complex non-scalar value like an array.

If a custom action supports multi-row mode, you can configure the add-in to process custom actions and other types of changes in batches. See Multi-Row Mode for Custom Actions.

BO item level custom actions can be performed on items in either the form or the table of a Form-over-Table layout, as well as on rows in a Table layout. BO level custom actions can be performed on BOs in a Table layout and the form in a Form-over-Table layout. They are not supported on the BO in the table of a Form-over Table layout. See Notes on Custom Actions for information on other custom action limitations.

Custom Action Configuration

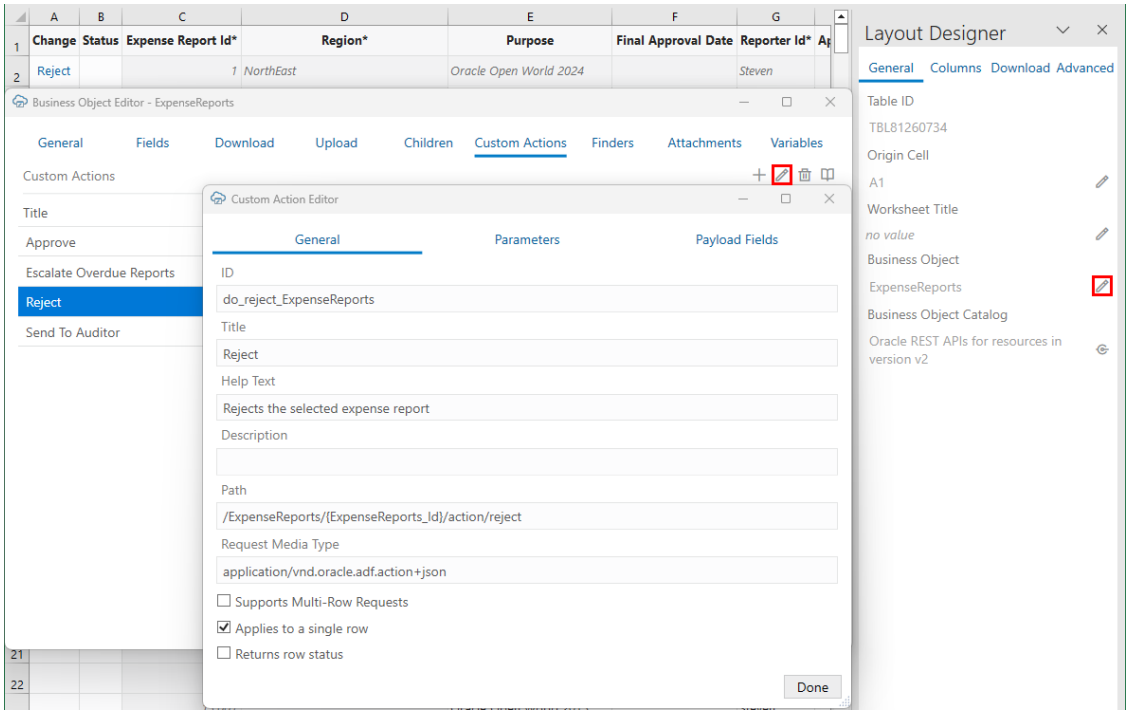
If the ADF REST service your workbook uses includes custom actions, Oracle Visual Builder Add-in for Excel can read the service's OpenAPI service metadata and display configured custom actions in the **Custom Actions** tab of the Business Object Editor.

Oracle Visual Builder Add-in for Excel includes two editors for viewing and editing custom action settings:

- **The Custom Action Editor**
- **The Custom Action Field Editor**

The Custom Action Editor

Open the Custom Action Editor for a custom action from the **Custom Actions** tab of the Business Object Editor. This image shows the Custom Action Editor for a "Reject" custom action on an `ExpenseReports` business object (BO) item path (`/ExpenseReports/{ExpenseReports_Id}/action/reject`).



The editor includes three tabs: **General**, **Parameters**, and **Payload Fields**.

The General tab includes a number of properties including ID, Title, Description, and Path. The Title, Help Text, and Description properties can be edited as needed in the Custom Action Editor. The other properties, ID, Path, and Request Media Type, reflect the service configuration and should be left as is.

The Payload Fields tab lists the custom action's configured payload fields.

Here are the editable settings on the General tab:

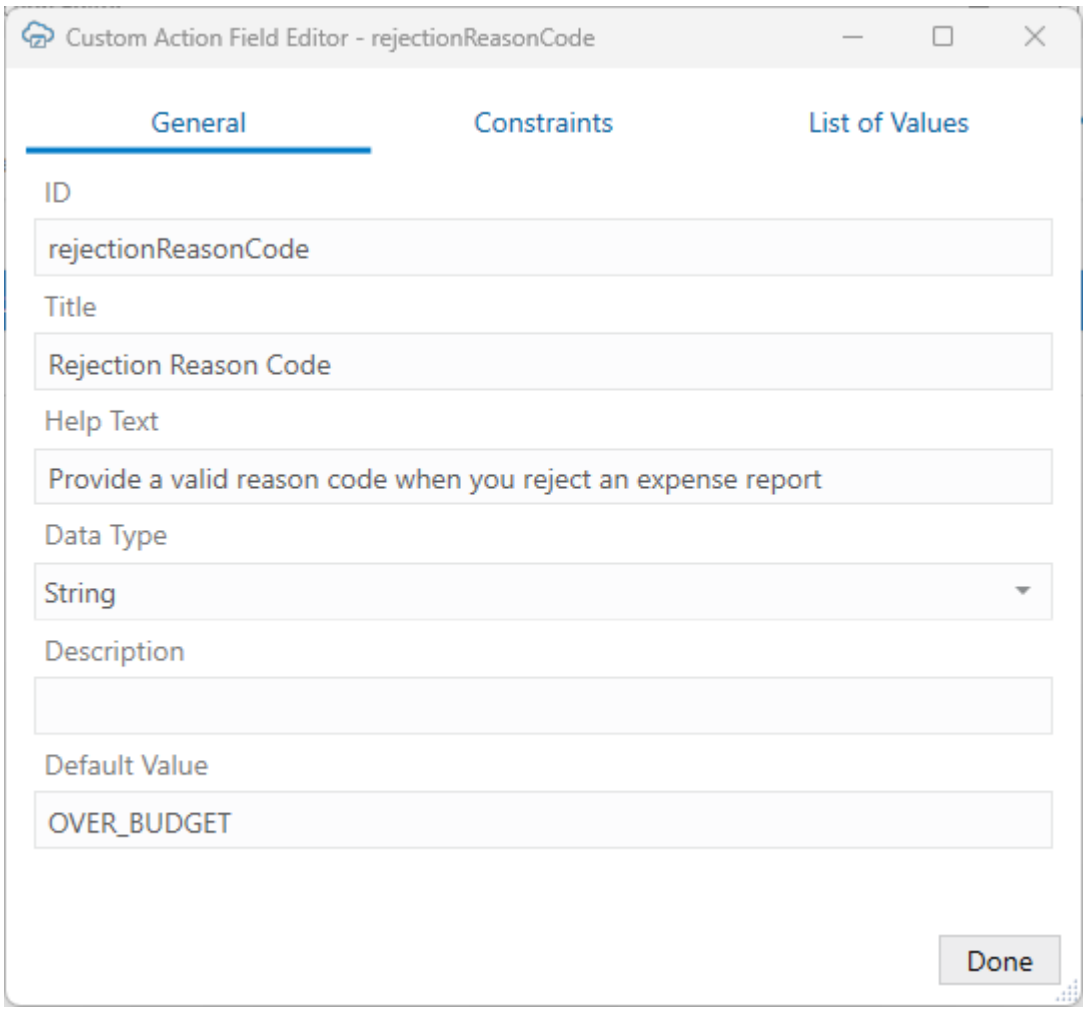
Control	Description
Title	The name of this custom action. Provide a short value that is meaningful to your business users. This value appears wherever business users can invoke the action. This value can be localized.
Help Text	Provide a brief explanation of what the custom action does. This value appears near the title when possible. This value can be localized.
Description	This is an internal technical description. This value only appears in the designer. It is not localizable.

> **Note:**
>
> When the **Help Text** value is displayed to your business users, the label next to that value is "Description".

Control	Description
Supports Multi-Row Requests	Select this check box if the custom action supports multi-row mode. See Multi-Row Mode for Custom Actions.
Applies to a single row	Select this check box if the custom action operates at the business object (BO) item level. If this check box is not selected, the custom action operates at the BO level.
Returns row status	Select this check box if the custom action is a row status custom action. See Row Status Custom Actions.

The Custom Action Field Editor

Open the Custom Action Field Editor for a payload field from the **Payload Fields** tab of the Custom Action Editor. This image shows the Custom Action Field Editor for a "Rejection Reason Code" payload field. This field is part of the "Reject" custom action.



The editor includes three tabs: **General**, **Constraints**, and **List of Values**.

The Title, Help Text, Description, and Default Value properties on the General tab can be edited as needed. The ID and Data Type properties reflect the service configuration and should be left as is.

Control	Tab	Description
Title	General	The name of this custom action field. Provide a short value that is meaningful to your business users. This value can be localized.
Help Text	General	Provide a brief description of the custom action field. This value can be localized. For single row custom actions, this help text appears in a popup when the business user selects the column header for a payload field in a Table layout. For BO-level custom actions, the help text appears in a popup when you hover over the help icon ( ) next to a payload field in the Perform Action wizard.
Description	General	This is an internal technical description. This value only appears in the designer. It is not localizable.
Default Value	General	If desired, you can set a default value for the custom action field. A default value can be an integer, decimal number, Boolean value, string, or Date(no time). Default values must match the data type of the custom action field. Default values can be constants or expressions. For example, you could enter `{ Today () }` to set a `Date(no time)` field to today's date. Reserved words are not supported. See Use Expressions in an Integrated Workbook. Lists of values are not supported.
Required	Constraints	Select this check box to mark the payload field as required.
Validation Rule and **Validation Failure Message**	Constraints	See Create Field Validation Rules.
List of Values fields	List of Values	See Configure a List of Values with a Business Object.

Add Custom Action Fields to a Table Layout

If the service your workbook uses includes custom actions on individual items of a business object, you can add defined custom action payload fields to your Table layout or the table part of your Form-over-Table layout.

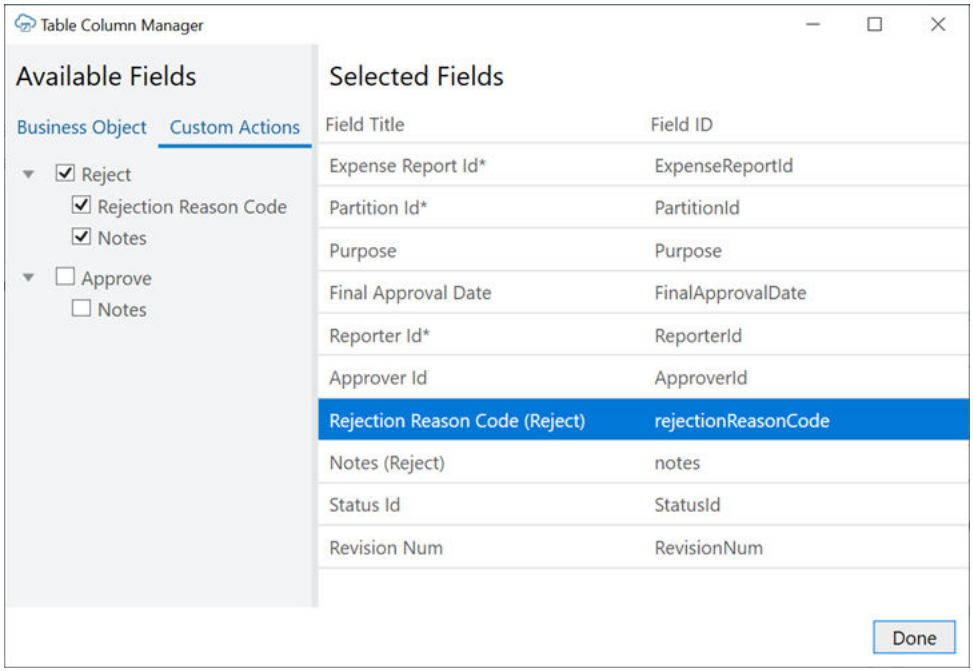
Let's take as an example a "`reject`" custom action that's exposed by an expense report business object to reject expenses.

This task assumes you have a Table layout for an expense report business object that includes a "`reject`" custom action. You can also add custom actions to the table of a Form-over-Table layout.

To add custom action table columns to a layout:

1. Select the layout, then click **Designer** from the Oracle Visual Builder tab to open the Layout Designer.
2. Click the **Columns** tab in the Layout Designer.
3. Click Manage Columns (+) to open the **Table Column Manager**.
4. From the **Table Column Manager**, click **Custom Actions** in the **Available Fields** pane.
5. In the **Selected Fields** pane, select the location where you want the payload field columns to be added. The columns will be inserted before the selected field.
6. In the **Available Fields** pane, select the payload fields you want to add.

   In this example, we'll select **Reason** and **Notes** to add these columns to the layout.



7. Click **Done**.

Once you add custom action fields to a layout in your integrated workbook, your business users can mark rows for the custom action by entering values in the payload field columns or

by using the ribbon command. See Perform Custom Actions in Table and Form-over-Table Layouts in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

When they click **Upload Changes**, Oracle Visual Builder Add-in for Excel invokes the custom action on the marked rows along with any relevant payload field values.

Row Status Custom Actions

Row status custom actions are BO item level custom actions that report row status information without changing the row state or field values.

Row status custom actions may flag an error or issue a warning when a business user enters an invalid value—say a duplicate invoice number or an expense amount that exceeds a certain threshold.

They can be performed manually as well as triggered automatically, for example, when a business user first selects a cell. You can configure a row status custom action to trigger automatically from the Layout Designer. Only row status custom actions *without* payload fields can be configured for automatic use. See Automate a Row Status Custom Action.

Configure a Row Status Custom Action

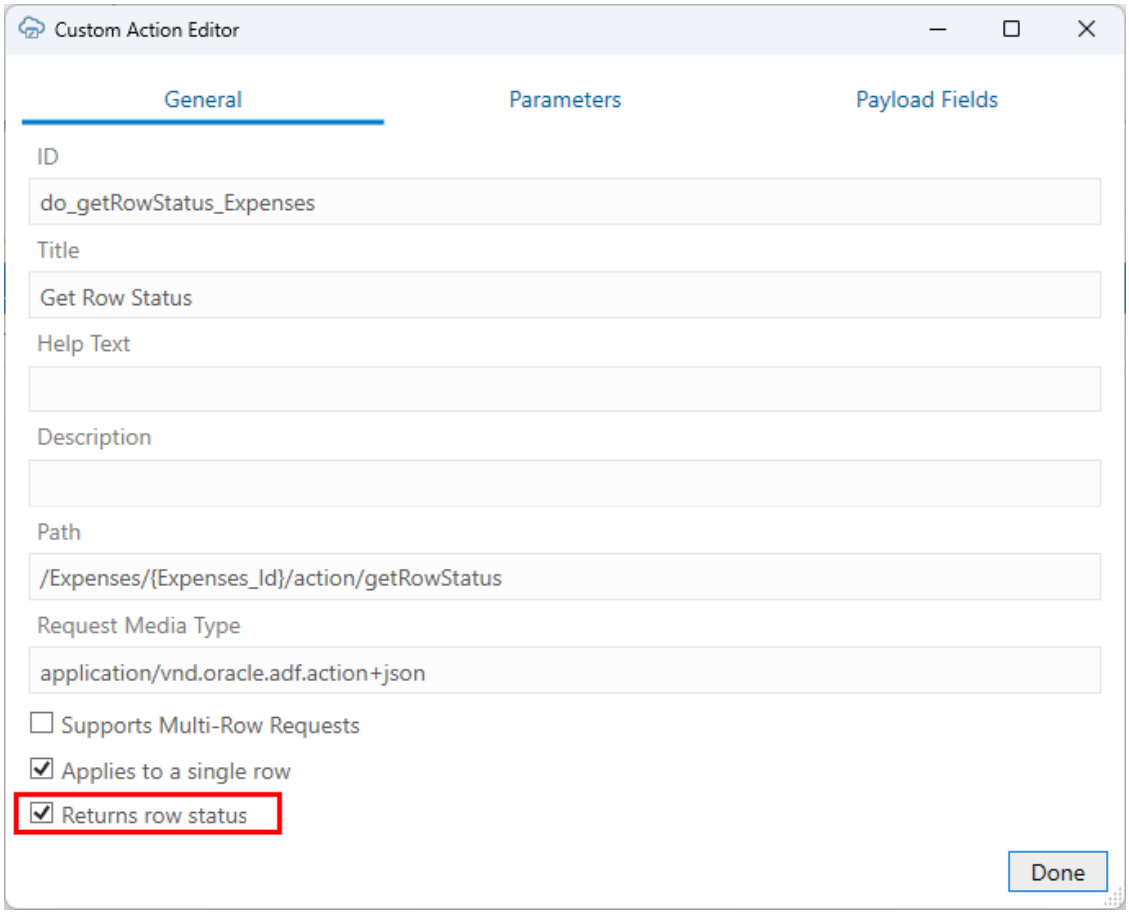
Set a custom action as a row status custom action from the Custom Action Editor.

A row status custom action *must*:

- Be a row-level custom action.
  This check box is unavailable if **Applies to a single row** is not selected.
- Report the current state of the row without changing the row state, field values, and so on.
- Return row status information following the expected row status custom action schema. See Row Status Custom Action Response Schema.

> **Caution:**
>
> Selecting this check box for a custom action that is not a row status one will result in unexpected behavior by the add-in. You could also experience data corruption if the custom action makes changes to the row.

To designate a custom action as a Row Status Custom Action, open the Custom Action Editor for the custom action, then select **Returns row status** from the General tab.



Automate a Row Status Custom Action

If a row status custom action doesn't include payload fields, you can configure it in the Layout Designer to trigger automatically when a business user first selects a cell in a Table layout.

A row status custom action operates on a single row and reports the current state without changing the row state or field values. It may be used to flag an error or issue a warning such as when it detects a duplicate invoice number or an expense amount over a set threshold.

If you decide not to configure a row status custom action for automatic invocation, it can still be performed manually by the business user, like other row-level custom actions.

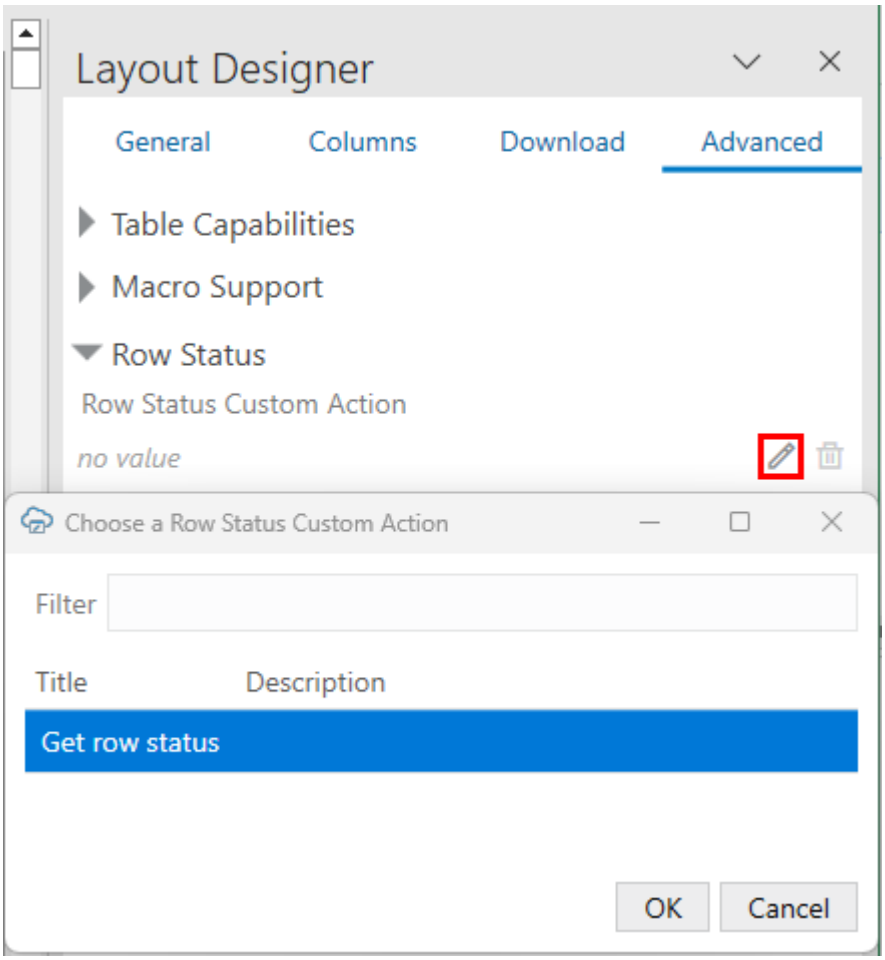
Before you begin, make sure the custom action is configured as a row status custom action in the Custom Action Editor. The **Returns row status** check box should be selected. See Row Status Custom Action Response Schema.

> **Note:**
>
> Automatic invocation is not supported for Form-over-Table layouts.

To configure a row status custom action for automatic use:

1. Open the **Advanced** tab of the Layout Designer.

2. Expand the **Row Status** section, then click the Edit icon (✎) to open the Choose a Row Status Custom Action window.



> **Note:**
>
> If the custom action you are looking for doesn't appear in the list, it is either not set as a row status custom action in the Custom Action Editor or it includes payload fields.

Once configured, the row status custom action is triggered automatically when a business user first selects a cell. As with other custom actions, results are displayed in the Status column and Status Viewer. See Perform Custom Actions in Table and Form-over-Table Layouts in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Since the row state and field values are not changed, the business user can continue to edit the row following a row status custom action without having to upload or download data. Oracle Visual Builder Add-in for Excel does not consider the row "stale".

The result of the row status custom action is cached, and the custom action will not be performed on selection again until:

- The layout is cleared
- Data is downloaded

- Row updates are successfully uploaded

> **Caution:**
>
> Technical limitations in Microsoft Excel require the spreadsheet to be disabled while the custom action request is executed. If the response time is slower than 100 ms, the data entry experience may be negatively affected.

Service Metadata and Response Schemas for Custom Actions

If a given REST API supports custom actions, they are described in the OpenAPI service metadata generated by the ADF REST service. For example, a `reject` BO item level custom action would appear in the `paths` collection:

```
// Note: some JSON content has been omitted for brevity/clarity

"/ExpenseReports/{ExpenseReports_Id}/action/reject": {
    "parameters": [
      {
        "$ref": "#/components/parameters/ExpenseReports_Id"
      }
    ],
    "post": {
      "summary": "reject",
      "description": "reject",
      "operationId": "do_reject_ExpenseReports",
      "responses": {
        "default": {
          "description": "The following table describes the default response
for this task.",
          "content": {
            "application/vnd.oracle.adf.actionresult+json": {
              "schema": {
                "type": "object",
                "properties": {
                  "result": {
                    "type": "string"
                  }
                },
                "required": [
                  "result"
                ],
                "additionalProperties": false
              }
            }
          }
        }
      },
      "requestBody": {
        "description": "The following table describes the body parameters in
the request for this task.",
        "content": {
          "application/vnd.oracle.adf.action+json": {
```

```
            "schema": {
              "type": "object",
              "properties": {
                "rejectionReasonCode": {
                  "type": "string",
                  "nullable": true
                },
                "notes": {
                  "type": "string",
                  "nullable": true
                }
              },
              "additionalProperties": false
            }
          }
        }
      }
    }
  }
```

Note the following:

- For a BO item level custom action, the path entry contains a path parameter for the row/item ID, for example: `/ExpenseReports/{ExpenseReports_Id}/action/reject"`

  In the case of a BO level custom action, the path (such as `/ExpenseReports/action/sendToAuditor`) doesn't include a path parameter for the row/item.
- The end of the path entry (`reject`) matches the name of the custom method defined in the service. See Publishing Custom Service Methods to UI Clients.
- The presence of a POST operation for the action path entry is required.
- In the `requestBody` schema, there are properties that match the parameters defined in the custom method signature from the service. In this document, these properties are referred to as custom action payload fields.

Row Status Custom Action Response Schema

This sample JSON file shows the response schema for a row status custom action:

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "properties": {
        "result": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "path": {
                        "type": "string",
                        "description": "A valid field ID of a field defined
on the business object that is bound to the layout. If a value is omitted,
the message is considered relevant to the entire row instead of a specific
field."
                    },
```

```
                    "summary": {
                        "type": "string",
                        "description": "A localized summary message, no
markup allowed."
                    },
                    "detail": {
                        "type": "string",
                        "description": "A localized detail message, no markup
allowed."
                    },
                    "type": {
                        "type": "string",
                        "enum": [
                            "info",
                            "warning",
                            "error"
                        ],
                        "default": "info"
                    }
                },
                "required": ["summary"]
            }
        }
    }
}
```

Custom actions return a JSON object with a "result" property. This property for a row status custom action must contain an array of Row Status Item objects.

The Row Status Item objects contain these properties:

Property Name	Required?	Default Value	Allowed Values
`path`	No	–	A valid field ID of a field defined on the business object that is bound to the layout. If a value is omitted, the message is considered relevant for the entire row instead of for a specific field.
`summary`	Yes	–	A localized string; no markup allowed
`detail`	No	–	A localized string; no markup allowed
`type`	No	info	info | error | warning

Sample Response - Rejection Messages:

```
{
  "result": [
    {
      "path": "invoiceNumber",
      "summary": "Duplicate Invoice Number",
      "detail": "Invoice numbers must be unique.",
```

```
      "type": "error"
    },
    {
      "path": "amount",
      "summary": "Invalid Amount",
      "type": "error"
    }
  ]
}
```

Sample Response - Info/Suggestion:

```
{
  "result": [
    {
      "summary": "A similar item is available from Acme Corp at a lower
price",
      "type": "info"
    }
  ]
}
```

Sample Response - Warning:

```
{
  "result": [
    {
      "path": "amount",
      "summary": "The expense report exceeds the standard limit and requires
special approval",
      "type": "warning"
    }
  ]
}
```

Multi-Row Mode for Custom Actions

If a custom action supports multi-row mode, Oracle Visual Builder Add-in for Excel can send multiple custom actions in a single REST request for better performance.

Multi-row mode is only relevant for custom actions defined on the item path since custom actions on individual items are sent as separate requests. A custom action on an entire business object may update multiple items in the business object but is accomplished using a single request.

To enable multi-row mode, select the **Supports Multi-Row Requests** check box in the Custom Action Editor. If the check box is not selected, the add-in invokes the custom action one row at a time.

If you attempt to use multi-row processing on a custom action that does not support it, you may get an error from the service. If the custom action doesn't work properly in multi-row mode, deselect the check box.

For more information, see Upload Changes Using Multi-Row Requests.

> **Note:**
>
> The `"batchEnabled" : false` property in the ADF REST service metadata determines the initial value of the **Supports Multi-Row Requests** check box in the Custom Action Editor. If the property is not configured, the check box is selected by default.
>
> After the catalog is created, the check box setting controls how the add-in behaves.

Notes on Custom Actions

Here are some things to keep in mind when using custom actions:

- Custom actions are not supported for pending Create rows or form in Create mode.
- Custom actions defined in the OpenAPI service metadata document that have request payload schema members that match business object fields are unlikely to function properly.
- ADF REST service OpenAPI service metadata documents do not indicate which custom action request payload fields are required. You can use the Business Object Editor to adjust the **Required** property on payload fields. See Custom Action Configuration.
- ADF REST services do not return the updated row. As a result, you need to download before making further changes.
- Custom action payload fields must be simple scalar types. Complex objects or maps are not supported.
- The add-in handles custom action results automatically. Here is how it processes the "result" member of the REST response for custom actions that aren't row status custom actions:
  - Simple scalar values are displayed verbatim
  - The first level of JSON objects and arrays are unpacked
  - Deeper levels of JSON are displayed verbatim
  - If there is a null value in an object or array, a "null" string is shown in the results. If the value of the result member is null, then the action result is suppressed.

  > **Note:**
  >
  > Row status custom actions have their own handling based on the response schema in Row Status Custom Action Response Schema.

- To use custom actions with the VBBO REST framework, you will need to define an Object Function for each custom action in the business object. In those Object functions, be sure to enable `Callable by External System`. See Object Functions for Business Objects.
- Row status custom actions must meet these requirements:
  - Must be a row-level custom action
  - Must not change row state, field values, and so on.
  - Must return row status information following the expected row status custom action schema. See Row Status Custom Action Response Schema.

- Row status custom actions can only be triggered automatically in Table layouts.

9

Use Lists of Values in an Excel Workbook

You can configure a **list of values** (LOV) for a field in your workbook to allow business users to select a valid value from a drop-down list. You can also allow users to enter a search term in a search box to filter this list to find the value they want.

Oracle Visual Builder Add-in for Excel supports LOVs on business object fields including custom action payload fields and descriptive flexfields with parameter and segment type client binds. The add-in also supports lists of values for search fields and row finder variables.

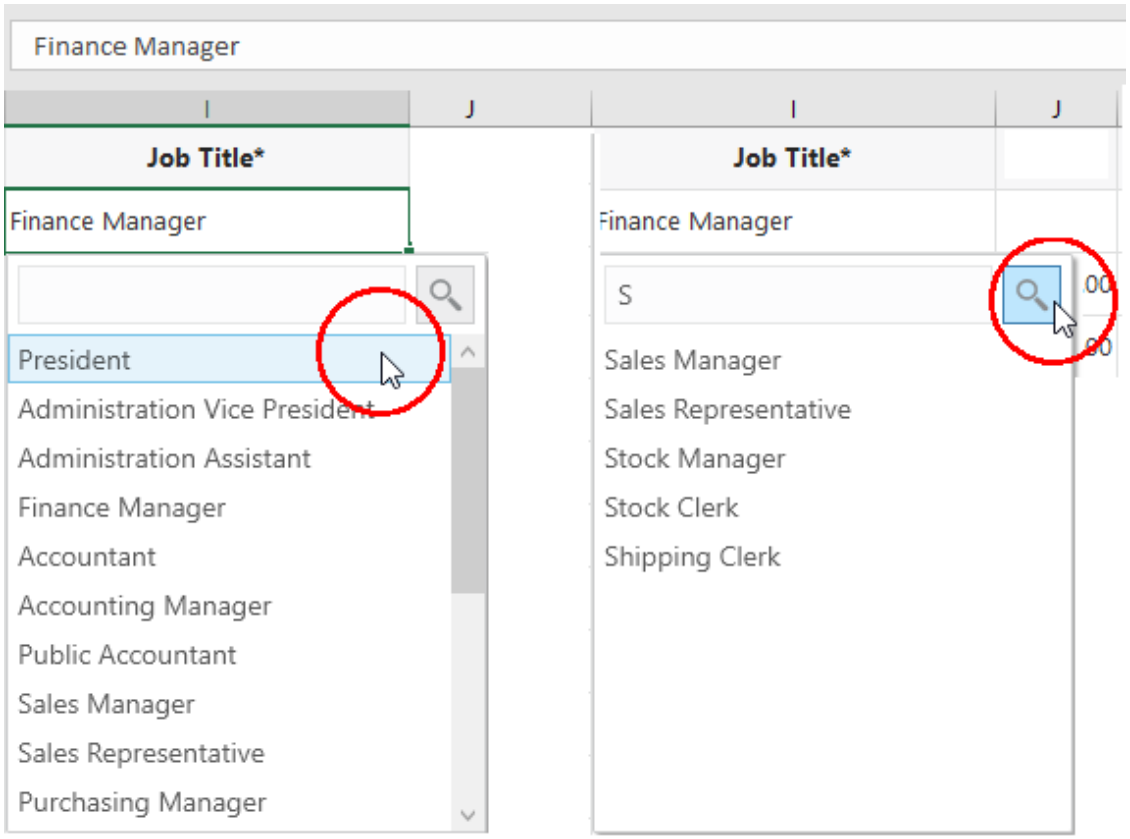
Lists of values also manage the conversion of internal ID or code values to and from user-friendly display values.

About Lists of Values

When you configure a list of values in your workbook, Oracle Visual Builder Add-in for Excel displays a drop down list of values when a business user selects the field.

If a filter with a search term parameter is configured, a business user can also search for values in the list by typing a search term in the search-and-select window and clicking the **Search** icon.

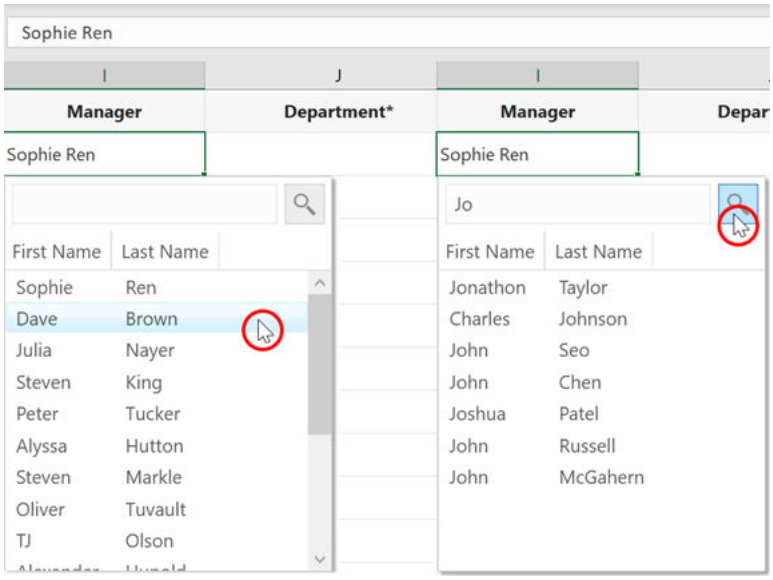
In this image, the search-and-select window on the left is populated with a list of values drawn from a Jobs business object. The search-and-select window on the right shows the result of entering `S` in the search box and clicking the **Search** icon.



When a business user clicks the **Search** icon, the list of values displays only job titles that begin with `S` such as "Sales Manager", "Sales Representative", and so on. Business users can also enter the full value in the search box, assuming it's a valid value such as "Sales Manager".

Lists of values can also be configured to show more than one display field in the search-and-select window. The add-in shows all the display fields, in separate columns, in the search-and-select window but concatenates those values configured for display when shown in the Excel cell.

In this image, the search-and-select window on the left shows separate columns for first and last names. The search-and-select window on the right shows the result of entering `Jo` in the search box and clicking the **Search** icon.



The list of values displays all first or last names that match the string. In this example, the add-in matches the search string "Jo" to the first name "Joshua" but also the last name "Johnson".

Each item in the list of values has a **display** value (which appears in the Excel workbook) and an **identity** value that is retrieved and posted to the business object field. For example, the Jobs business object used for the list of values might contain these display and identity values:

Display value (jobTitle)	Identity value (jobId)
President	PRES
Finance Manager	FIN_MGR
Sales Manager	SAL_MGR

On download, the identity values are replaced by the display values; On upload, the display values are replaced by the identity values.

Configure a List of Values with a Business Object

Configure a list of values that uses values from another business object. Lists of values are supported for business object fields, custom action payload fields, and row finder variables.

When you create a list of values, you can associate the selected field with the values from another business object. For example, you may have two business objects: Employees and Jobs. If the Employees layout includes a JobId column, you may want to add a list of values that references the `jobId` field from the Jobs business object.

When a business user selects a cell from the Job Title column in the Employees layout, a search-and-select window shows a list of values drawn from the Jobs business object for the user to choose from.

If you have a hierarchy of business objects, you can configure a list of values that uses a child or lower descendant business object as your data source. See Use a Child Business Object as a Data Source.

You can configure a list of values to show more than one display field in the search-and-select window. The add-in shows all the display fields, in separate columns, in the search-and-select window but only those you configure are shown in the Excel cell. If you choose to show more than one field in the cell, the values are concatenated.

For business object sources, you can configure a filter to restrict the results to a given subset of values. You can also configure it to let business users filter the list based on a search term they type in a search box. See Configure a Filter for a List of Values.

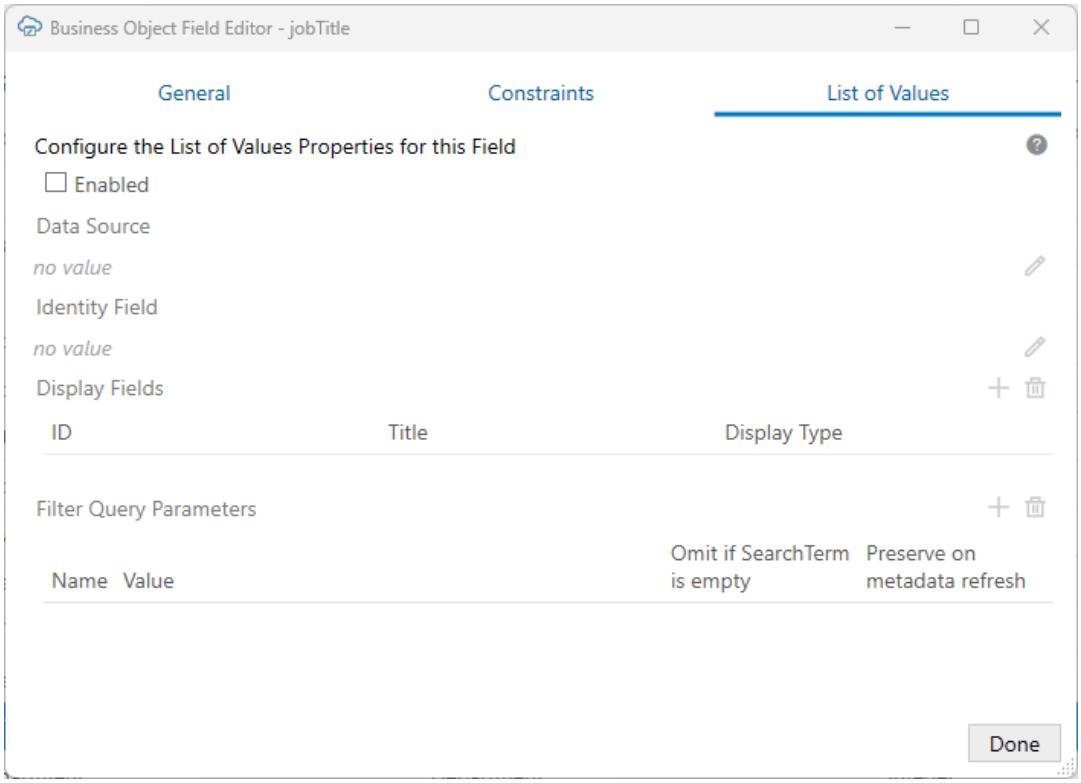
If the catalog is missing the desired business object, you can add a new business object to an existing catalog (see Add Business Objects to an Existing Catalog).

If you want to reference a business object with a different base path, you can create or import a business object into the current catalog and then override the base path (see Override a Business Object's Base Path).

> **Note:**
>
> This procedure takes you through the steps to create a list of values for a business object field used in a form, table, or search. The steps are the same for custom action payload fields and row finder variables except for the navigation. To open the List of Values page for a custom action payload field or row finder variable, open the Business Object Editor and go to the **Custom Action** or **Row Finder** page. From there, select the appropriate field or variable.
>
> To create a list of values based on a local data source instead, see Configure a List of Values with a Local Data Source.

To create a list of values:

1. From the Layout Designer, click the **Edit** (✎) icon next to the Business Object field.
2. From the Business Object Editor, click the **Fields** tab, then select the business object's field.
3. Click the **Edit** (✎) icon to open the Business Object Field Editor, then click the **List of Values** tab.



4. Select the **Enabled** check box on the **List of Values** page.
5. Click the **Edit** (✎) icon next to the **Data Source** field, then pick an appropriate data source. Only business objects from the current catalog are displayed in the Choose a Data Source window.
   This data source provides the display values for the corresponding identity values.
6. Click the **Edit** (✎) icon next to the **Identity Field** field, then choose the appropriate identity field from the data source.
   This is the field used to look up the display values for the identity values in the current field.
7. Click the **Add Field** (+) icon to open the **Available Business Object Fields Editor**, then choose the desired display field.
   These fields come from the data source and are shown instead of the identity values where this field is used in a layout.

   You can choose multiple display fields for one list of values. Repeat this step to add additional display values.
8. For each display field, select either **Picker and cell** or **Picker only** from the **Display Type** list.
   If you configured only one display field, use **Picker and cell** to display the value in both the Excel cell and the search-and-select window. For additional display fields, use **Picker only** if you don't want to display the value in the Excel cell.

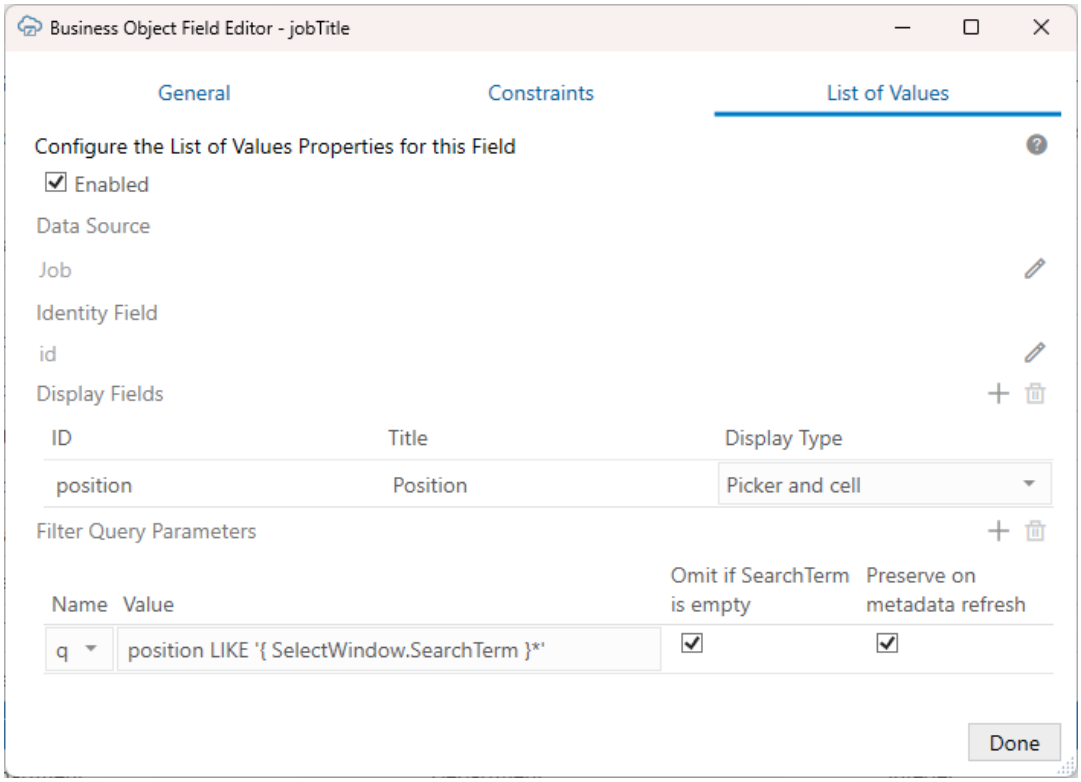
> **Note:**
>
> When configuring the display values, make sure the information in the cell is unique and meaningful for your business users. Take for example a Contact field in your layout. To ensure your business users have enough information to determine the right contact for a purchase order, you may want to include the contact name, company, and email as display fields in the Excel cell. In this case, ensure that the display type for these display fields are set to **Picker and cell**.

9. To configure a filter, click the **Add Query Parameter** (+) icon next to Filter Query Parameters, then set a name and parameter value.
If the parameter requires a search term, select **Omit if SearchTerm is empty**.

   If you want the add-in to retrieve the latest query parameter definitions from the service metadata on refresh, deselect **Preserve on metadata refresh**. See Refresh Parameter Definitions for a Lists of Values.

   Repeat this step to create additional parameters.

   In this image, a list of values is configured for the jobTitle field that is populated with values from the `position` field of the Job business object. A filter query parameter has been configured that allows a business user to return positions that start with the business user's search term.



10. Click **Done**.

Once you define a list of values, the choice list will appear wherever that business object or payload field appears. For the row finder variable, the choice list will appear wherever that row finder variable appears during download.

On first download and before showing all or filtered values in the search-and-select window, the add-in may send requests to download data from the data source. This downloaded data is cached.

After you modify the configuration of any list of values, clear the cached data by clicking **Clear List of Values Cache** from the **Advanced** menu. See Clear Cache for a List of Values.

Use a Child Business Object as a Data Source

Oracle Visual Builder Add-in for Excel supports using child or lower descendant business objects as data sources for your lists of values. The only extra step you'll need to do is configure the required path parameters for the child object. To populate the list of values, the add-in needs to send a GET request that include the full path—including path parameters—to the child business object.

Let's take a look at an example. Suppose you have a hierarchy of business objects for purchase requisitions, where **Purchase Requisitions** is the parent, **Lines** is the child, **Distributions** is the grandchild.

In your integrated workbook, you have a Table layout for **Purchase Order Line Distributions** that includes a `RequisitionDistributionId` field the business user uses to enter an appropriate requisition distribution Id. To make this easier, you'd like to create a list of values for this field that displays the available distributions from the **Distributions** grandchild business object.

When you create this list of values, you'll need to provide the full path to the **Distributions** business object. You'll do this by referencing the appropriate field in the layout using expressions. See About Expressions.

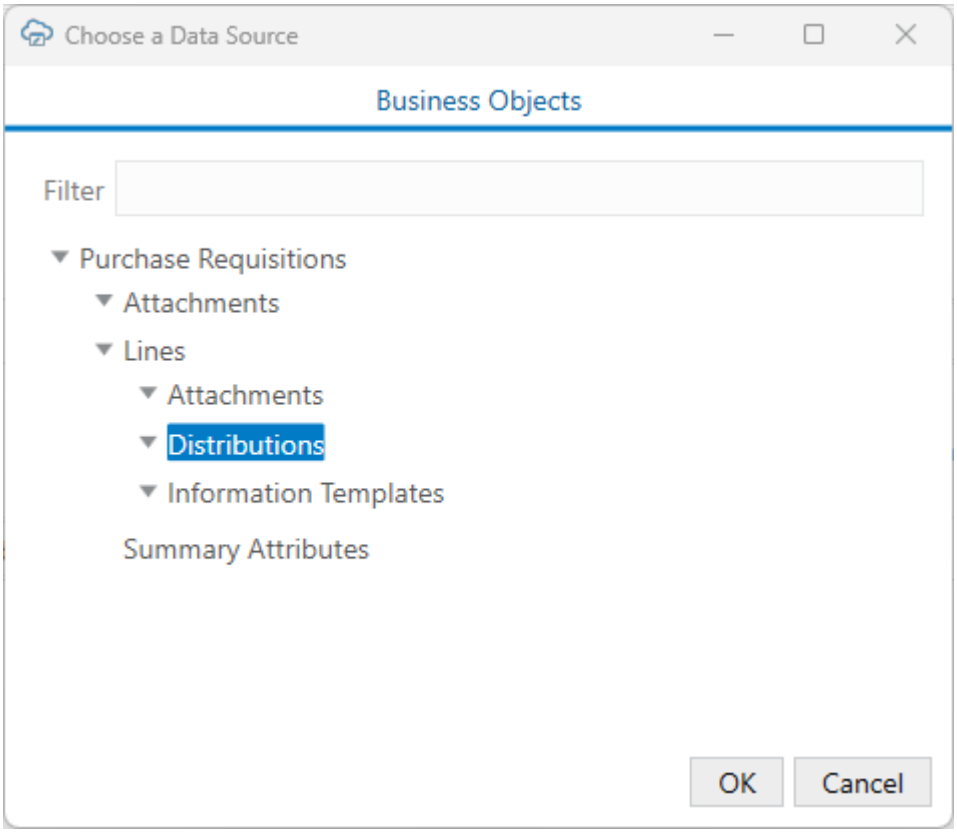
The collection path for this business object is:

```
/purchaseRequisitions/{purchaseRequisitions_Id}/child/lines/
{purchaseRequisitions_lines_Id}/child/distributions
```

> **Note:**
>
> The add-in does not support child business objects that use complex keys for the path parameter values since these keys cannot be written using the add-in expression language.

To create this list of values:

1. Navigate to the Business Object Field Editor for the `RequisitionDistributionId` field, then open the **List of Values** tab.
2. Enable the list of values, then click the **Edit** (✎) icon next to the **Data Source** field to open the **Choose a Data Source** window.



In this image, the available business objects are displayed in a nested tree. To find the business object you want, use the arrows to expand or collapse a node or type in the **Filter** box to find matching entries.

3. Pick an appropriate data source—in this case, **Distributions**—then click **OK**.

   The List of Values tab now includes a new Path Parameters section below the Data Source area:

4. In the **Value** fields, enter expressions for the required paths that reference the appropriate fields.

   The appropriate fields are:

Required Path	Referenced Field
`{purchaseRequisitions_Id}`	`RequisitionHeaderId`
`{purchaseRequisitions_lines_Id}`	`RequisitionLineId`

Assuming these fields are all in the same layout, your expressions would look like this:

Path Parameters

Name	Value
purchaseRequisitions_Id	{ this.BusinessObject.Fields['RequisitionHeaderId'].Value }
purchaseRequisitions_lines_Id	{ this.BusinessObject.Fields['RequisitionLineId'].Value }

> **Note:**
>
> These expressions must reference field values in the current business object or any ancestor business object. Those fields must be present in the layouts.

5. Complete the configuration the way you would for a list of values using a top-level business object.

Configure a Filter for a List of Values

Configure a filter for your list of values to determine which items from the business object used as the data source are included in the list. A filter is a set of one or more URL query parameters that are appended to a REST request to the referenced business object.

Filters are not available for lists of values that use a local data source.

The filter query parameters are added to all requests to the referenced business object. These include the request to fetch the initial set of values as well as the one sent when the business user clicks the **Search** icon after entering a search term.

You should always configure a filter with a search term parameter since the search option is always available to the business user.

There are a few basic filter scenarios you may want to consider when configuring a list of values:

- **Search term parameter only**: You don't need to use a filter parameter for the initial set of values if you want your business users to access all values from the referenced business object. In this case, just configure a search term parameter and let your business user filter the list based on a search term. See Configure a Filter for a Search Term Only.
- **Filter and search term parameters**: To limit the choices available from the referenced business object, consider using a filter parameter such as a finder. Also, add a search term parameter to let your business user search from within the list. See Configure a Filter to Limit Available Choices.
- **Dynamic and search term parameters**: A dynamic filter parameter limits the number of values in a list of values based on a field value in the current layout or search box. For example, you may filter a Job Title list of values based on the selected employee's department so that only job titles for this department are displayed. See Configure a Filter with a Dynamic Parameter.
- **Cascading lists of values**: A cascading list of values uses dynamic filters where the value selected in one list determines the range of values that users can select from subsequent lists. See Configure a Cascading List of Values.

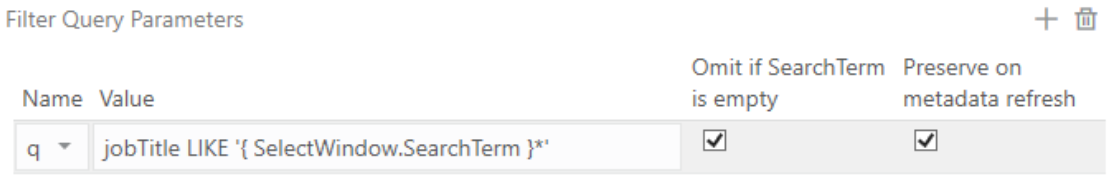
See About Expressions for more information about expressions in lists of values.

Consult the API documentation for your service to determine the appropriate search syntax to use in the Value column. For example, if you use an Oracle ADF REST Resource service, consult Understanding Framework Support for Query Syntax in *Accessing Business Objects Using REST APIs*.

Configure a Filter for a Search Term Only

Configure a filter with a search term parameter to allow your business users to search for values using a search term.

For example, let's say you have a list of values that displays job titles for your company. To allow the business user to filter this list, you can use a "q" parameter value to return job titles that start with the business user's search term:

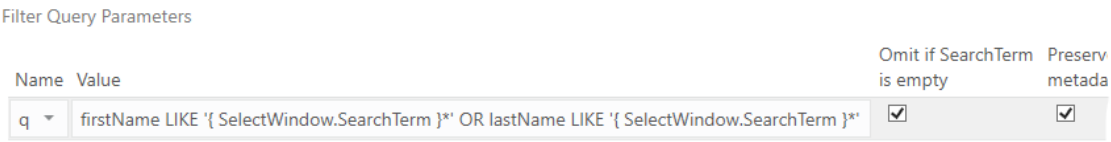


> **Note:**
>
> Select **Omit if SearchTerm is empty** to ensure this parameter is only applied when there is a value in the search box.

In this example, a business user can type "ma" in the search box and click the **Search** icon to display job titles such as "Manager" and "Marketing Specialist".

You should always configure a search term parameter for your filter since the search option is always available to your business users.

If you have more than one display field in your list of values, you can use the "OR" operand to search on each of the fields. For example, to match text to either the employee's first or last name, create a parameter like this:



When a user types in the search box and clicks the **Search** icon, the add-in displays all first or last names that match the string. In this example, the add-in would match the search string "Jo" to the first name "Joshua" but also the last name "Johnson".

> **Note:**
>
> The syntax for the filter query varies based on the REST service type and expected query syntax for a business object.

Configure a Filter to Limit Available Choices

Configure a filter with multiple query parameters to limit the choices that are fetched from the referenced business object.

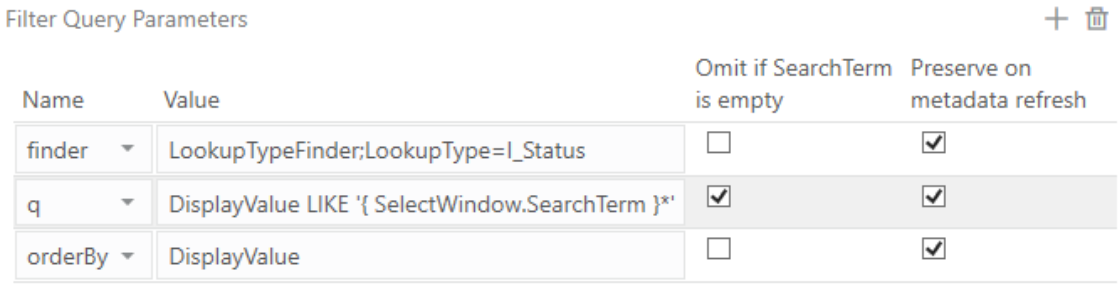
For example, you may have a referenced business object that stores values for a number of different lists of values and you need a way to retrieve only the values for the current field.

Let's consider a business object, `StandardLookupsLOV`, that stores immigration details for a company's employees such as immigration status (`I_Status`) and type (`I_Type`).

You may want to use a finder, for example, to return just the immigration status (`I_Status`) values, such as "Not Applicable", "Pending", "Accepted" and so on, for your list of values.

You'll also want to create a search term parameter to allow your business users to find the value they are looking for.

To do this, configure your filter as shown here:



In this case, the initial set of values is determined by the finder configuration. If the business user types a search term and clicks the **Search** icon, the list is further limited to status values whose `DisplayValue` starts with the search term.

Configure a Filter with a Dynamic Parameter

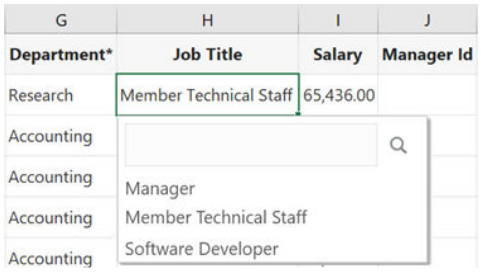
You can configure a list of values that is dynamically filtered based on another value in a layout, search field, or row finder variable. To do this, configure a filter for your list of values that uses an expression that references another business object field or row finder variable.

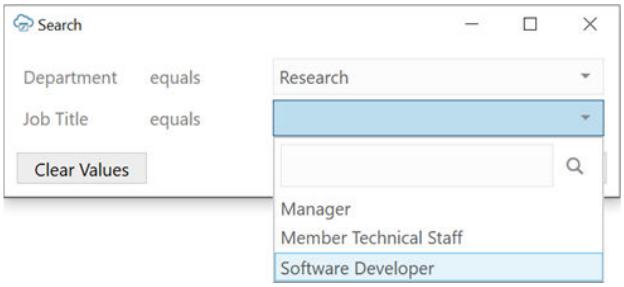
Referencing Business Object Fields in a Dynamic Filter

Let's say you have a Table layout for an Employees business object that includes two columns: "Department" and" Job Title". You may want to configure a list of values on the Job Title column that includes a filter that filters the list based on the value in the Department cell in the same row. When configured, your business users will see only those job titles that are relevant for the department that they have selected.

Of course, you can also use these fields in a search. Again, when a business user selects a department, Oracle Visual Builder Add-in for Excel shows only relevant job titles in the Job Title field.

As you can see on the left in this image, the list of values (in this example, "Job Title") is based on the value of another field ("Department"). When used in a search as shown on the right, the Job Title list is filtered based on the value in the first search field ("Department").





To configure a dynamic filter for the Job Title list, use a parameter that filters based on the department Id. If you have a row finder that filters job titles by department, you can use an expression like this:

The expression in the form, `{ this.BusinessObject.Fields['FieldId'].Value }`, where *FieldId* is the ID property of another field in the current business object or search—in this case, `DepartmentId`.

Since this expression refers to another field in the currently selected row or search, make sure this field ("DepartmentId", in this case) has been added to your layout or added before the field with the dynamic filter when it is used in a search. See Use the Search Editor to Find Required Data for information about how to use these lists of values in a search.

The Oracle Visual Builder Add-in for Excel replaces this expression with the corresponding field value (specifically, the identity value) when sending the request for the referenced business object's list of values. When business users select the Job Title field for an employee in the marketing department, the list includes only job titles associated with Marketing.

Likewise when using a dynamic list of values in a search. When business users select the Job Title field, the list includes only job titles associated with the department they first chose from the Department field.

If the corresponding field value is missing or invalid, the expression evaluation fails and an error is reported.

> **Tip:**
>
> Remember to include a search term parameter, such as `jobTitle LIKE '{ SelectWindow.SearchTerm }*'`, to allow your business users to filter the list to jobs based on a search term they provide.

Referencing Ancestor Business Object Fields in a Dynamic Filter

For a list of values that you plan to use in a layout, you can also refer to a parent or higher ("ancestor") business object in your expression using one or more `Parent` terms. For example, to create a list of values that refers to a field in the parent business object, you would use `ByDeptFinder;DepartmentId={ this.BusinessObject.Parent.Fields['DepartmentId'].Value }`.

To refer to a grandparent business object, use two `Parent` terms instead: `this.BusinessObject.Parent.Parent.Fields['DepartmentId'].Value`.

> **Note:**
>
> Do not refer to an ancestor business object if you plan to use your list of values in a search. Such a configuration will result in an error at runtime.

> **Tip:**
>
> When using a dynamic filter in a layout, it is recommended that you add an ancestor column for the field ("DepartmentId", in this case) to the current layout. The ancestor field must be positioned *before the column ("Job Title") that references it*. This makes it easier for the add-in to find the correct value during runtime and improves performance.
>
> If there is no matching ancestor column in the current layout, the add-in uses other ancestor columns to map the current row to its parent row. It repeats this process until it finds the ancestor row and reads the field value. This can take some time if there are a large number of parent rows. See Add a Parent Column to Support Row Creation.

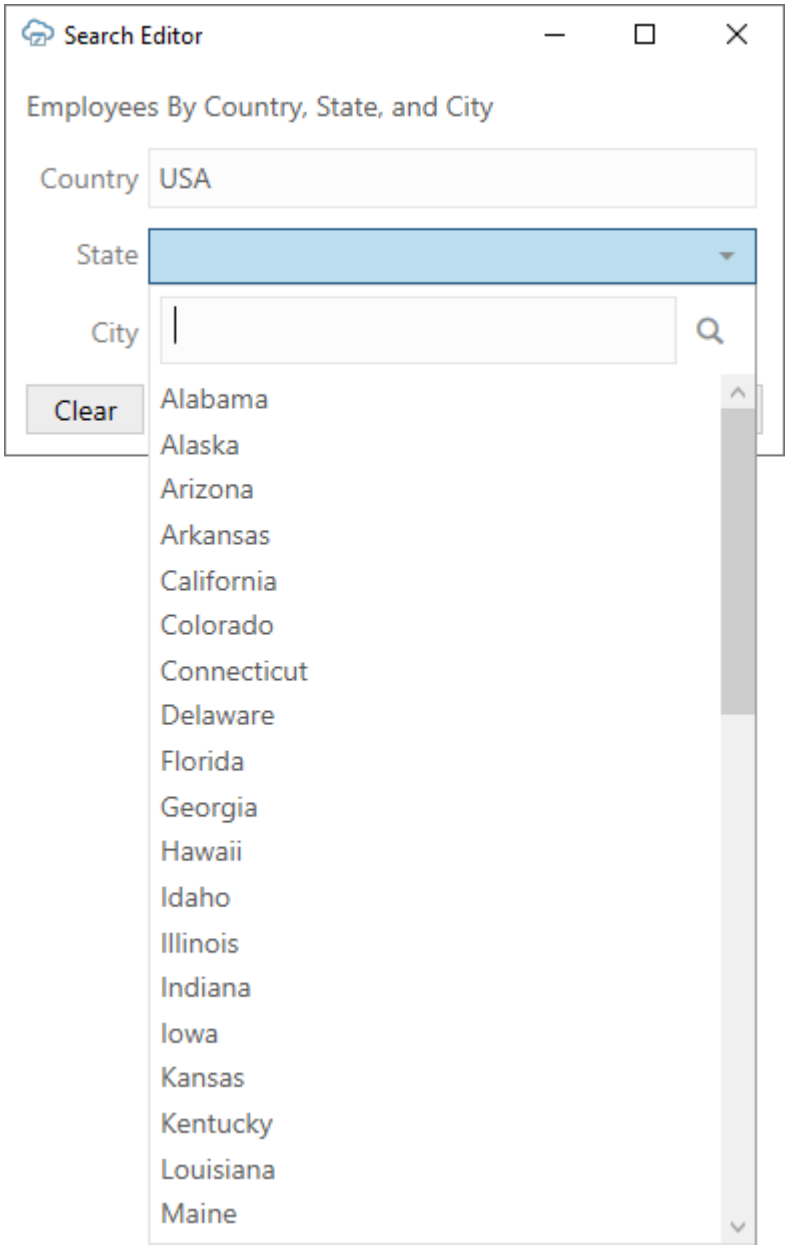
Referencing Row Finder Variables in a Dynamic Filter

Just as you do for a business object field, you can also define a list of values on a row finder variable and use it in a search. Like our business object field example, you can also define a parameter that filters the list of values based on the value selected from another variable in the same row finder.

Consider a row finder, "Employees By Country, State, and City", that filters employees by country, state, and city. This row finder includes three row finder variables: **CountryId**, **StateId**, and **CityId**.

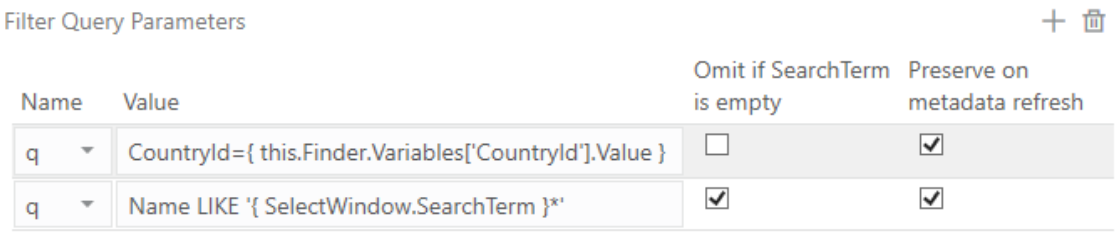
You may want to configure lists of values on the each of these variables. For the first one, **CountryId**, you don't need a dynamic parameter. But for the other two, you would want to configure a parameter that filters the list of values based on the value selected in the previous variable. So, the values in the **StateId** variable would be based on the value of **CountryId** and those of the **CityId** variable would be based on the value of **StateId**.

Here's how this would look when configured:



In this image, the Search Editor uses our sample row finder and includes three fields for the row finder variables. The list of values in the **State** field is filtered based on the value entered in the **Country** field. In this case, the business user has selected "USA" from the **Country** list, and so the add-in is displaying only US states in the **State** list.

To configure a list of values on the **StateId** row finder variable, use a States business object as your data source, then define a "q" query that filters based on the value in **CountryId** variable, like this:



The first filter parameter uses the expression, `CountryId={ this.Finder.Variables['CountryId'].Value }`, where:

- `this` represents the currently selected row finder variable (**StateId**, in our example);
- `Finder` is the row finder to which this finder variable belongs (**EmployeesByCountryStateCity**);
- `Variables['CountryId']` is the finder variable (**CountryId**) whose value determines the available choices of the current finder variable; and
- `Value` is the value of the field.

This expression sets the value of the variable, `CountryId`, used in the q query to the value of the **CountryId** row finder variable—"USA" in our example. See About Expressions.

This value is then used to filter values from the States business object and populate the list of values with just US states.

> **Note:**
>
> A finder variable can only depend on the values of variables from the same finder. It cannot depend on values from the Form-over-Table or Table layout.

To use the row finder in a download filter, use the row finder as part of a search as described in Use Row Finders to Limit Downloaded Data. When configured, the business user will see the Search Editor when they click **Download Data** where they can select values for each finder variable.

> **Tip:**
>
> By default, row finder variables are added to the Search Editor in the alphabetical order (**City**, **Country**, and **State** in our example). For a better user experience, consider changing the order so that the dependent ones appear after those they depend on (**Country**, **State**, and **City**).

Configure a Cascading List of Values

Just as you configure a single list of values with a dynamic parameter, you can also configure **cascading lists of values** for two or more lists using dynamic parameters. In cascading lists of values, the value selected in one list determines the range of values in the next list, and so on.
Let's suppose you have an Employees layout with three columns to capture an employee's location: Country, State, and City. To make sure only appropriate values are entered for an employee, you decide to use dynamic parameters to filter the State list of values based on the Country value, and the City list of values based on the Country and State.

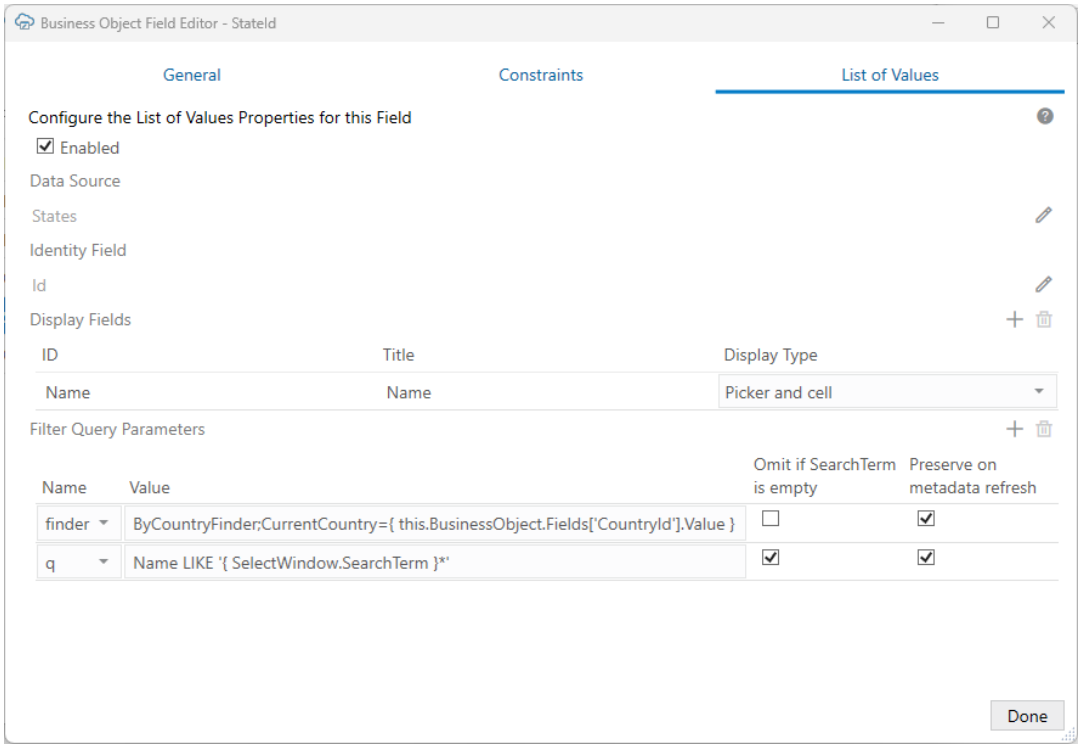
If the service is an ADF REST service and has complete metadata for the cascading lists of values, the filter query parameters of the corresponding field's list of values are automatically configured.

If the service does not have complete metadata for a list of values but you know it supports search syntax that makes for cascading lists of values, you can manually configure the lists of values by adding filter query parameters, as described here.

> **Note:**
>
> You can configure cascading lists of values for row finder variables as well as for business object fields used in a layout or search. This task uses the Business Object Field Editor but the steps are the same for the Row Finder Variable Editor.

To configure a cascading lists of values:

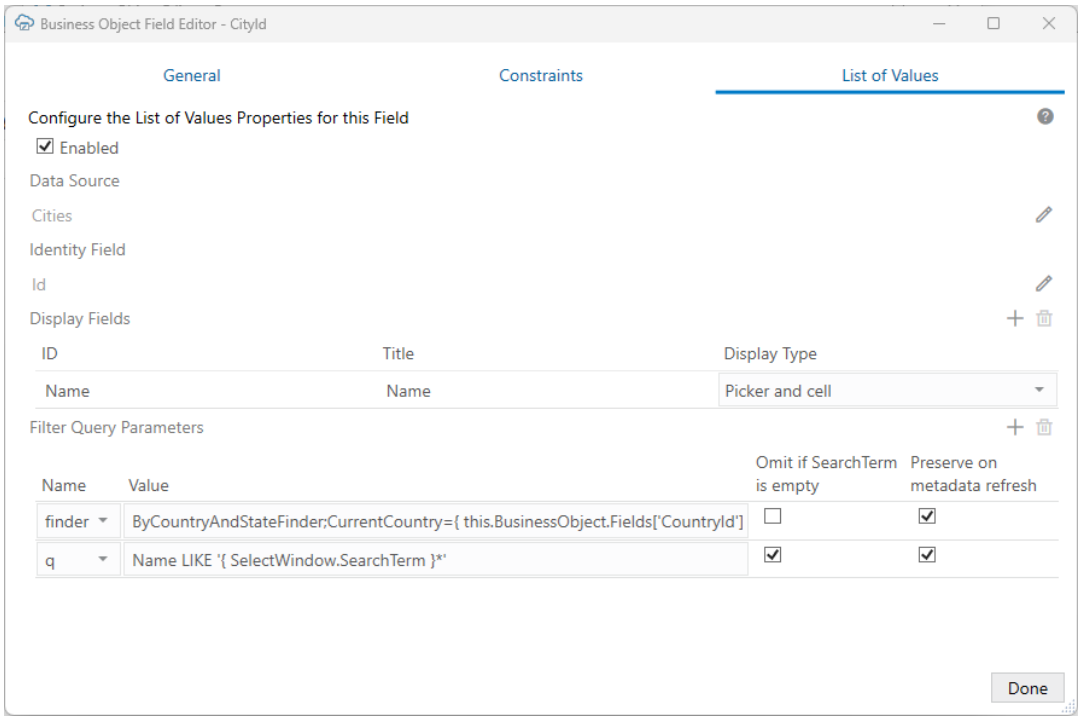
1. Start by creating a list of values for the first field, "CountryId", with just a search term filter (`Name LIKE '{ SelectWindow.SearchTerm }*'`). See Configure a Filter for a Search Term Only.
   This filter allows the business user to type in the CountryId field to find the country they want.
2. Then, create a list of values for the second field, "StateId", that filters the list of values by the current country, as shown in this image:



This example uses a finder query, "ByCountryFinder", that filters based on the value in the **CountryId** field in the row. If the business user selects "USA" from the **CountryId** list, Oracle Visual Builder Add-in for Excel displays only US states in the **StateId** list.

As with the first list of values, this list also includes a search term filter so that the business user can type in the field to find the state they want.

3. Finally, create a list of values on the third field, "CityId", and filter the list of values by CountryId and StateId, as shown in this image:



This example uses another finder query, "ByCountryAndStateFinder", that filters based on the values in both the **CountryId** and **StateId** field in the same row. Now, if the business user selects "California" from the **StateId** list, the add-in displays only cities in California in the **CityId** list.

In a scenario like this with three fields ("fieldA", "fieldB", and "fieldC"), remember that fieldC depends on both fieldB and fieldA. So when you create the filter for fieldC's list of values, also include fieldA's value expression. In general, all of fieldB's dependencies must also be fieldC's dependencies.

Note the following name-value parameters under Filter Query Parameters:

Table 9-1 Example Name-Value Parameters in Filter

Field	Name	Value
StateId	**finder**	`ByCountryFinder;CurrentCountry={ this.BusinessObject.Fields['CountryId'].Value }`
CityId	**finder**	`ByCountryAndStateFinder;CurrentCountry={ this.BusinessObject.Fields['CountryId'].Value },CurrentState={ this.BusinessObject.Fields['StateId'].Value }`

Now you are ready to use the fields or row finder variables with the cascading lists of values in your integrated workbook.

To use the fields in a layout, add and order them using the Form or Table tab of the Layout Designer. See Manage Fields in a Form or Table. To use them in a search, add and order them in the Search Editor. See Configure a Search for a Layout.

To use the row finder variables in a row finder search, first order the row finder variables in the Variables tab of the Row Finder Editor. See Configure Row Finders for a Business Object.

Then, select the row finder from the Download tab of the Layout Designer as described in Use Row Finders to Limit Downloaded Data.

> **Note:**
>
> Remember to order the cascading lists of values in the layout or search so that they appear in a meaningful order—Country, State, and City in our example.

Notes on Filters

Refer to these notes when configuring filters for lists of values:

- Oracle Visual Builder Add-in for Excel does not validate your filter configuration. If you provide invalid information, the add-in will make invalid requests that return errors.
- The query parameters are applied in the order that they are configured.
- The add-in applies URL encoding to the final resolved value of each filter query parameter before sending the request.
- Expressions like `{ this.BusinessObject.Fields['FieldId'].Value }` can't be used in the filter for:
  - Row finder variable's list of values
  - Custom action payload fields if the field referred to is not added to the layout
- If your list of values includes more than one display field, you can configure a search query that returns matches from any of the display fields. See Configure a Filter for a Search Term Only for an example of a search term parameter that includes the OR operand.
- If you configure multiple values with the same parameter name, the last value that is not "omitted" (**Omit if SearchTerm is empty** is enabled) is used by default.
  For example, in this filter configuration, the first query parameter is used *if there is no search term*; the second parameter is used *if there is a search term*.

- Filters are not available for list of values based on local data sources

Configure a List of Values to Populate Related Fields

You can configure a list of values to populate additional related fields (referred to as "driven fields") when a business user selects a value.

Let's suppose you have an Invoices table in your workbook with three related columns for the supplier: Supplier Id, Supplier, and Supplier Number. You may want to configure a list of values on the Supplier Id field that uses a Suppliers business object as the list of value's data source. When you do this, you may also want to map other fields from the source—such as `Supplier` and `SupplierNumber`—to the appropriate columns in the layout so that the supplier's name and number are automatically populated when a supplier Id is selected.

> **Note:**
>
> This feature is not supported on row finder variables or custom action payload fields.

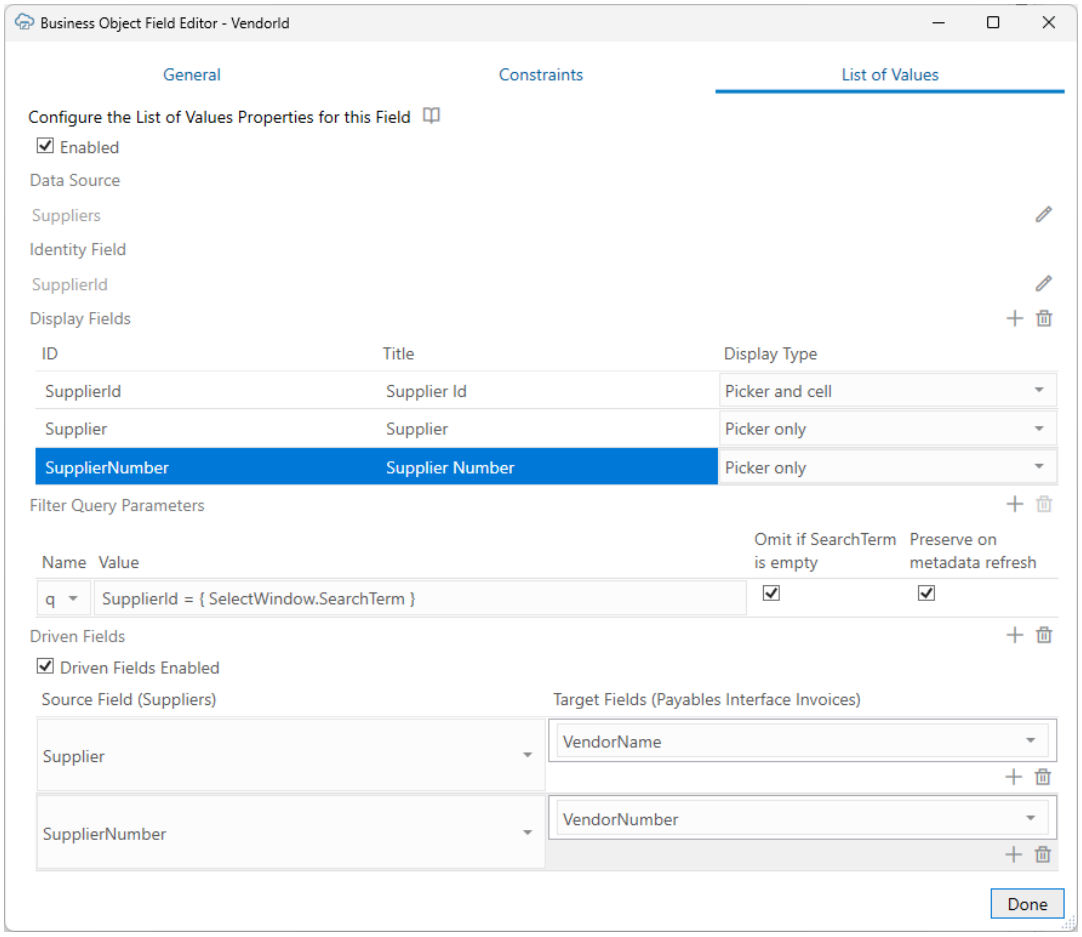
To configure related fields for a list of values:

1. Open the Business Object Field Editor of the field for which you want to configure the list of values—in this example, the `VendorId` field (the Id for the Supplier Id column in the layout).
2. Open the List of Values tab and configure the list of values as required.

   In this example, you are going to use:

   - The Suppliers business object for the data source
   - The `SupplierId` field from the data source as the Identity Field
   - The `SupplierId`, `Supplier`, and `SupplierNumber` fields as display fields. Select **Picker and cell** for `SupplierId` and `Picker only` for `Supplier` and `SupplierNumber`.

     > **Note:**
     >
     > If you include the additional source fields as display fields (with a display type of **Picker only**), the business user can preview their values before selecting a supplier Id.



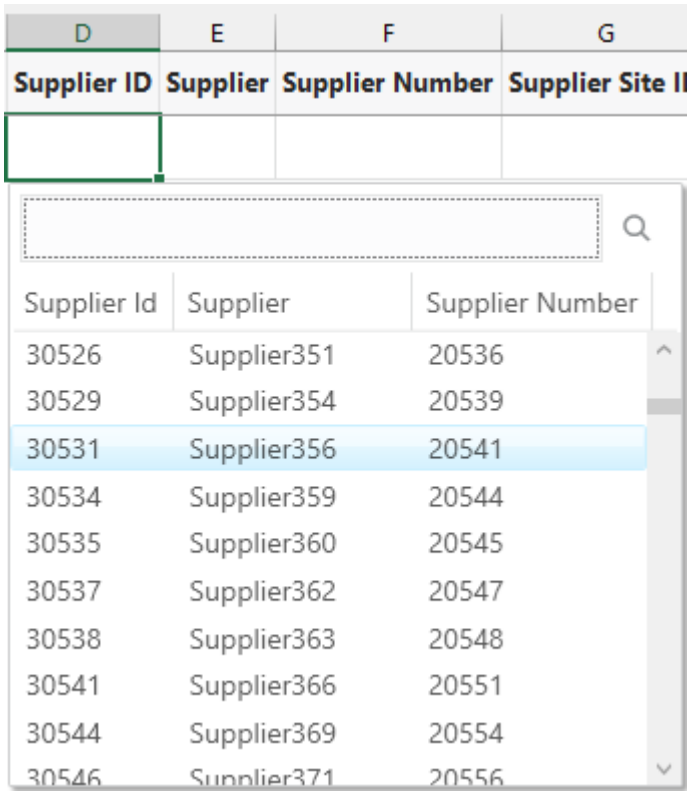
3. To map the source fields in the Suppliers business object to target fields in your Invoices layout:
   a. Select **Driven Fields Enabled**.
   b. Click the **Add Field** icon (+) next to Driven Fields to add a row to the table.
   c. Select a source field from the row's **Source Field** drop-down list.

      The value in this field will be written to the target fields. In this example, select the `Supplier` field.
   d. Select a target field in the layout from the **Target Fields** drop-down list.

      In this example, select `VendorName` to map it to the `Supplier` source field.
   e. Click the **Add Target** icon (+) below the drop-down list to add an additional target.
   f. Repeat steps 3.b to 3.e for any additional fields.

      In this example, map the `SupplierNumber` source field to the `VendorNumber` target.
4. Click **Done**.

When a business user selects the Supplier Id field from the table, the add-in displays the search-and-select window with values for the three supplier fields.



If the user selects a supplier Id from the list, the add-in populates the Supplier and Supplier Number fields with the related values. Note these values are not populated if the business user pastes a value into the Supplier ID cell instead of selecting one from the list.

If required, the user can change these values from the row.

Use a Local Data Source for a List of Values

You can configure a list of values that uses a "local data source" for its values. A local data source is a set of name-value pairs stored in the integrated workbook.

These local data sources can be created by Oracle Visual Builder Add-in for Excel to store `enums`, if the service includes these. You can also create your own local data sources if required.

- Create a Local Data Source for a List of Values
- Configure a List of Values with a Local Data Source

Create a Local Data Source for a List of Values

You can create a local data source that provides the values for the list of values (LOV). This data source stores these values in your integrated workbook as a set of name-value pairs.

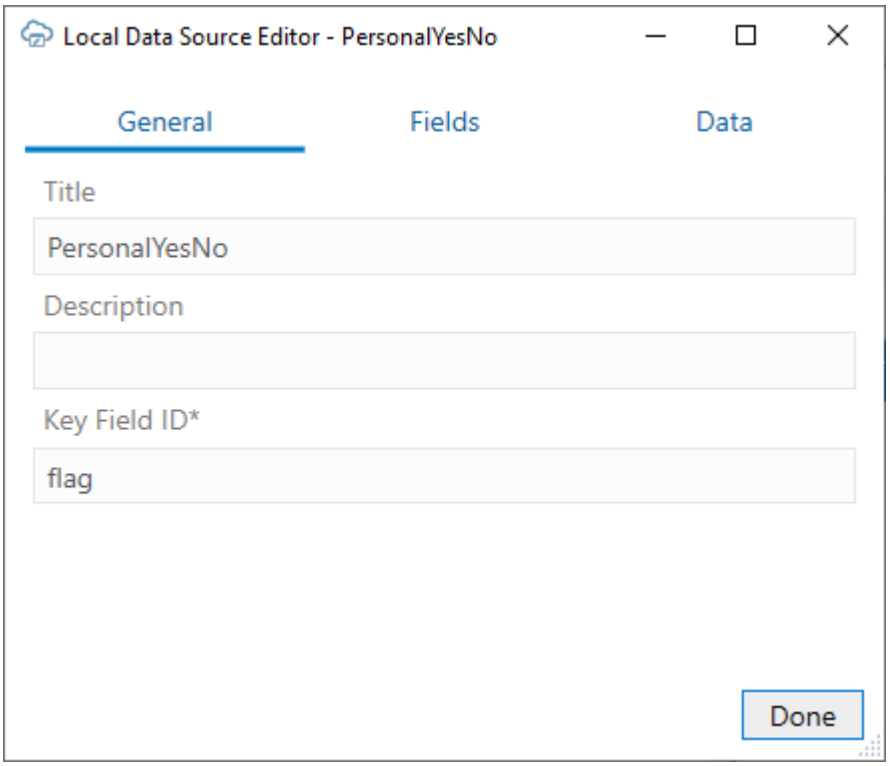
Local data sources have a couple of key benefits. They convert field codes to user-friendly display values automatically to improve the user experience. They also provide the business user with a drop-down list that restricts the choices to a fixed set of appropriate values.

Let's consider an example. You may have a business object, `Computer`, with a Boolean field, `personal`, that captures whether the device is a business or personal one. In this case, the field expects only "True" and "False" values.

You may want to create an LOV on the `personal` field that displays "Yes" and "No" values in the list. To do this, create a new local data source in the catalog that includes these values, then configure an LOV for the `personal` field that uses this data source.

To create a local data source for a list of values:

1. From the **Oracle Visual Builder** tab, click **Manage Catalogs**.
2. In the **Manage Business Object Catalogs** window, select the catalog where you want to define the data source and click the **Edit Business Object Catalog** icon.
3. From the **Business Object Catalog Editor**, click **Local Data Sources**.
4. From the **Local Data Sources** tab, create a new data source:
   a. Click the **Add** icon (+) to create a new data source.
   b. Select the new data source from the list, then click the **Edit** icon (✎) to open the new data source in the **Local Data Source Editor**.
   c. From the **General** tab, enter a title, description, and key field ID for the data source.



- **Title**: Identifies the local data source when creating a list of values. The title only appears in designer windows, such as the **List of Values** tab in the **Business Object Field Editor**.
- **Description**: (Optional) Helps workbook developers understand the purpose and use of the data source. The description only appears in the designer.
- **Key Field ID**: The unique key for identifying the data in the local data source. In this example, the key field ID is `flag`. You'll create this key field from the **Field** tab, as described in the next steps.

  When exporting strings for translation, rows are identified by their key field values. Key field values themselves are not translated.

5. From the **Fields** tab, create fields for the data source.

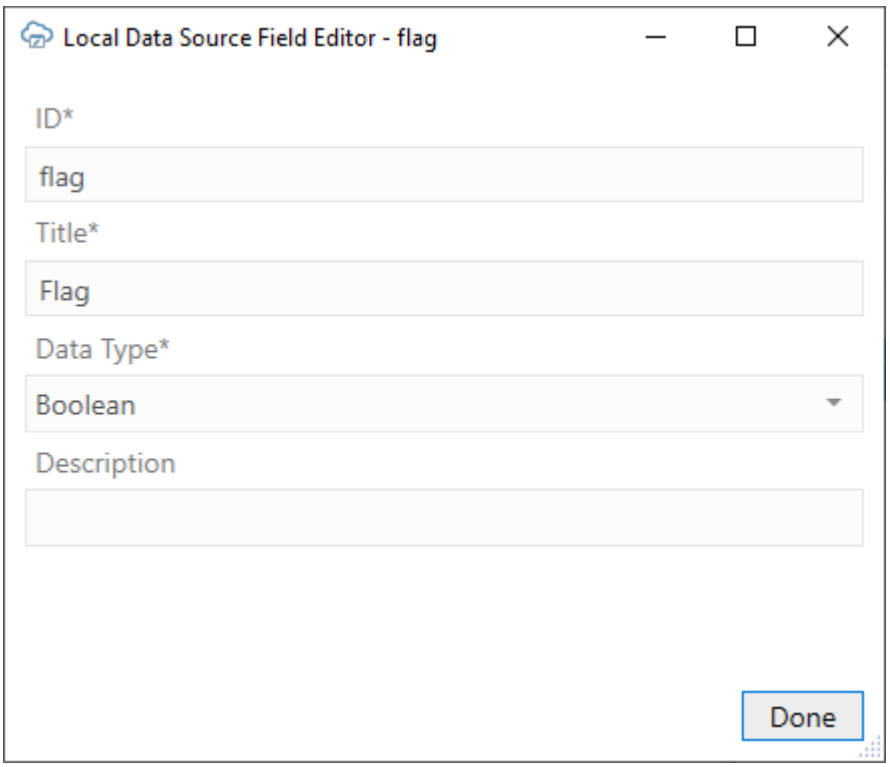
   In this example, you will create two fields: one with a Boolean data type (`flag`) and the other with a string data type (`display`). `flag` is the key field ID in our example. `display` stores the values that appear in the list.

   > **Note:**
   >
   > In some cases, you may want to show just the identity values ("True" and "False" instead of "Yes" and "No") to your business users. In this situation, create one field and use it later as both the identity field and display field in your LOV.

   a. Click the **Add** icon (+).

   b. Select the new field from the list, then click the **Edit** icon (✎) to open the field in the **Local Data Source Field Editor**.

   c. Provide details for the field, then click **Done**.



   The value you type into the ID field should match the value you entered into the **Key Field ID** field previously. In this example, type in `flag` for the ID field and choose "Boolean" from the "Data Type" list.

   Also, provide a title and description for the field. These values help workbook developers identify the field when displayed in designer windows.

> **Note:**
>
> The **ID**, **Title**, and **Data Type** fields are required. The **Description** field is optional.

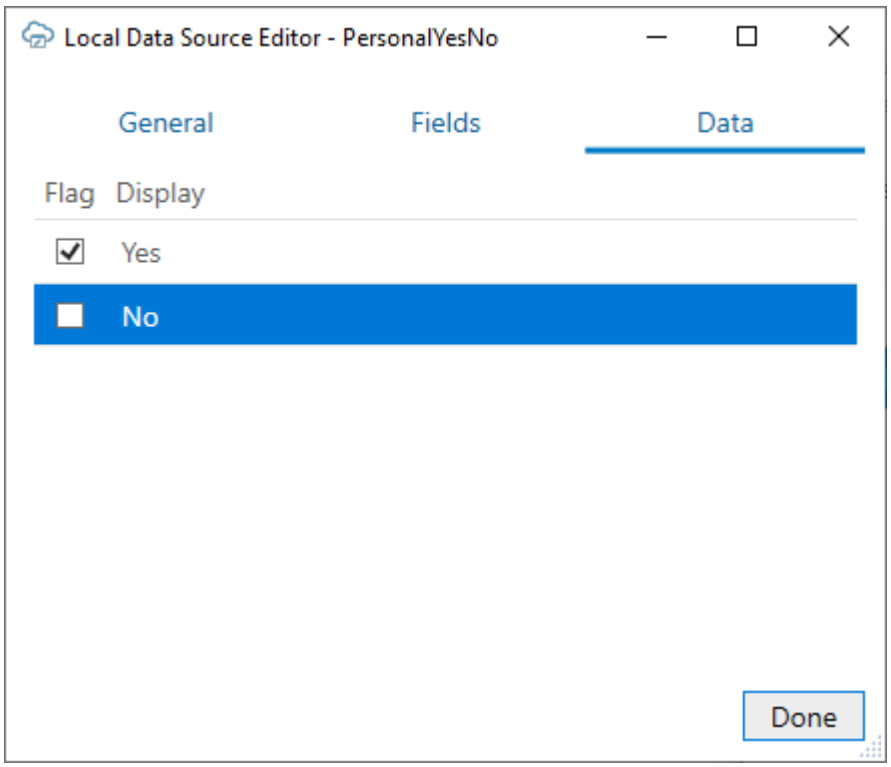
d. Repeat these steps for the next field. In this example, create the `display` field for your data source, using "String" for the data type.

> **Note:**
>
> Make sure all the fields are configured properly before opening the **Data** tab, as described in the next step. If you make any changes in the **Fields** tab, any data entered in the **Data** tab is cleared.

6. From the **Data** tab, add values to your data source in the newly-created fields.

   In this example, the **Flag** (key ID) field has a Boolean data type and displays check boxes for the expected values. Select the check box for "True" and leave it unselected for "False".

   Then, enter the user-friendly value (for example, "Yes" for the selected check box) in the **Display** field, as shown in this image.



   To add a row, select the last row and press Enter. To delete a row, select it and press Delete. To select more than one row, use the Shift key.

7. Click **Done**.

Now that you have created a local data source, you are ready to use it as the data source for an LOV. See Configure a List of Values with a Local Data Source.

Configure a List of Values with a Local Data Source

Configure a list of values that references a local data source. Lists of values are supported for business object fields, custom action payload fields, and row finder variables.

Local data sources are listed along with business objects in the Choose a Data Source window when you configure your LOV. If you need to create a local data source first, see Create a Local Data Source for a List of Values.

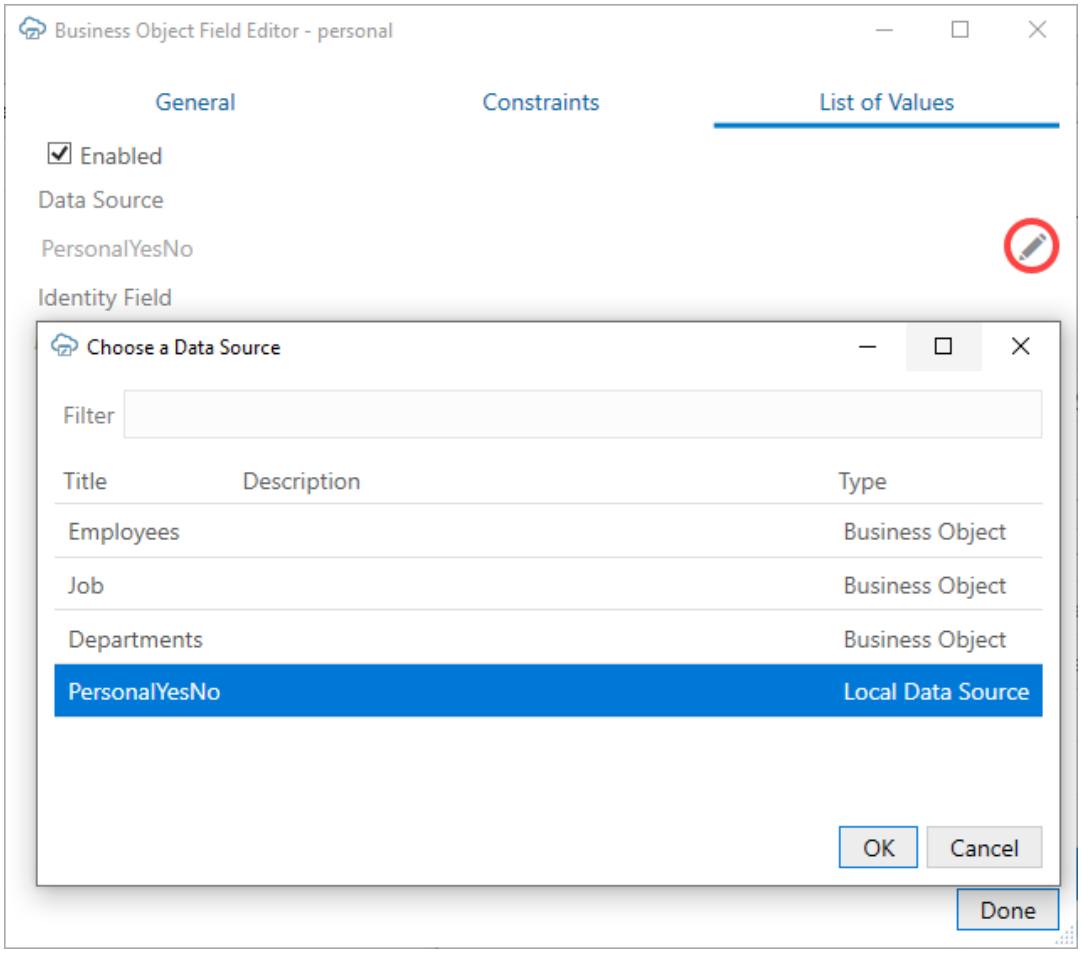
When a business user selects a cell from the personal column in the Computer layout, a popup window shows a list of values drawn from your local data source for the user to choose from.

You can configure a list of values to show more than one display field in the popup window. The add-in shows all the display fields, in separate columns, in the popup window but only those you configure are shown in the Excel cell. If you choose to show more than one field in the cell, the values are concatenated.

> **Note:**
>
> Lists of values based on local data sources do not support filters or search in the drop-down list.

To create a list of values with a local data source:

1. From the Layout Designer, click the **Edit** (✎) icon next to the Business Object field.
2. From the Business Object Editor, click the **Fields** tab, then select the business object's field.
3. Click the **Edit** (✎) icon to open the Business Object Field Editor, then click the **List of Values** tab.
4. Select the **Enabled** check box on the **List of Values** page.
5. Click the **Edit** (✎) icon next to the **Data Source** field, then pick an appropriate local data source.



This data source provides the display values for the corresponding identity values.

6. Click the **Edit** (✎) icon next to the **Identity Field** field, then choose the appropriate identity field from the local data source.

   This is the field used to look up the display values for the identity values in the current field.

7. Click the **Add Field** (+) icon to open the **Available Business Object Fields Editor**, then choose the desired display field.

   These fields come from the data source and are shown instead of the identity values where this field is used in a layout.

   You can choose multiple display fields for one list of values. Repeat this step to add additional display values.

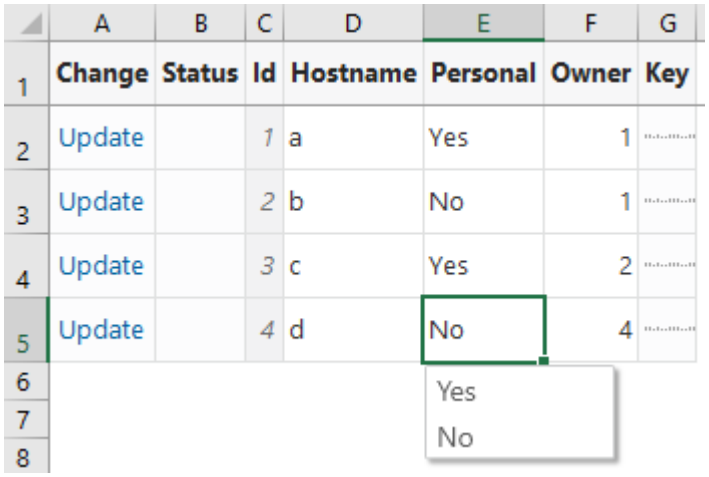
8. For each display field, select either **Picker and cell** or **Picker only** from the **Display Type** list.

   If you configured only one display field, use **Picker and cell** to display the value in both the Excel cell and the popup window. For additional display fields, use **Picker only** if you don't want to display the value in the Excel cell.

> **Note:**
>
> When configuring the display values, make sure the information in the cell is unique and meaningful for your business users. Take for example a Contact field in your layout. To ensure your business users have enough information to determine the right contact for a purchase order, you may want to include the contact name, company, and email as display fields in the Excel cell. In this case, ensure that the display type for these display fields are set to **Picker and cell**.

9. Click **Done**.

During runtime, the business user sees the user-friendly values from the display field of the data source in the LOV, as shown here:



The choice list will appear wherever that business object or payload field appears. For the row finder variable, the choice list will appear wherever that row finder variable appears during download.

The add-in caches the data of list of values in the workbook. After you modify the configuration of any list of values, click **Clear List of Values Cache** from the **Advanced** menu.

List of Values for Descriptive Flexfields

Oracle Visual Builder Add-in for Excel supports lists of values (LOV) for descriptive flexfields (DFF) in ADF REST services in some scenarios.

About DFFs

DFFs are a feature of ADF REST polymorphic business objects. A DFF is a set of fields for a particular record that differs based on the value of a discriminator field (also known as a context segment). There are three types of fields in a DFF field set:

- Global fields. Global fields are available for all values of the discriminator field.
- The discriminator field or "context segment". The discriminator field determines which context-sensitive fields are displayed in the layout.
- Context-sensitive fields. Context-sensitive fields are dynamic based on the value of the discriminator field.

Let's suppose you have an Employee polymorphic business object that you are using in a layout and that this business object includes a DFF for the employee's location. This DFF may include a couple of global fields ("Site" and "Time Zone"), a discriminator field ("Region"), and context-sensitive fields ("Zip Code", "Postal Code", "State", and "Province").

The global fields are available for all values of the discriminator field but the context-sensitive fields are either available or unavailable depending on this value. If the business user selects "United States" from the Region field, only "Zip Code" and "State" are available.

LOVs and DFFs

LOVs are supported on all three field types—global, discriminator, and context-sensitive fields—and for both LOV types—independent and dependent—with some restrictions. Dependent LOVs are based on a dynamic filter so that the values in the LOV depend on the value of another field. For example, a dependent LOV may display a list of cities based on the U.S. state selected in another field. Independent LOVs do not have a dynamic filter.

Independent LOVs are supported for all three types of fields and are automatically configured based on the service's OpenAPI service metadata document. However, only LOVs for the discriminator field are visible in the designer and configurable by the workbook developer.

Dependent LOVs are supported in these scenarios:

- On a global field based on the value in another global field. These LOVs are automatically configured based on the OpenAPI service metadata document.
- On a context-sensitive field based on another context-sensitive field. These LOVs are automatically configured based on the OpenAPI service metadata document. Typically, these LOVs use segment-type client binds.

Client Binds

There are two types of client bind: segment-type and parameter-type.

Segment-type client binds are supported automatically as long as the service's OpenAPI service metadata document provides all the proper metadata. If the metadata is present, the add-in configures the list of values at runtime.

Parameter-type client binds must be configured from the Polymorphic Information tab of the Business Object Field editor before they become available. See Configure the Bind Parameters for a Descriptive Flexfield's List of Values.

Keep in mind these limitations:

- Client binds are only supported for context-sensitive fields and not global fields.
- Only binds of data type string and number are supported. Binds of other date types are not supported.
- The add-in uses client binds if and only if the OpenAPI service metadata document provides proper metadata.

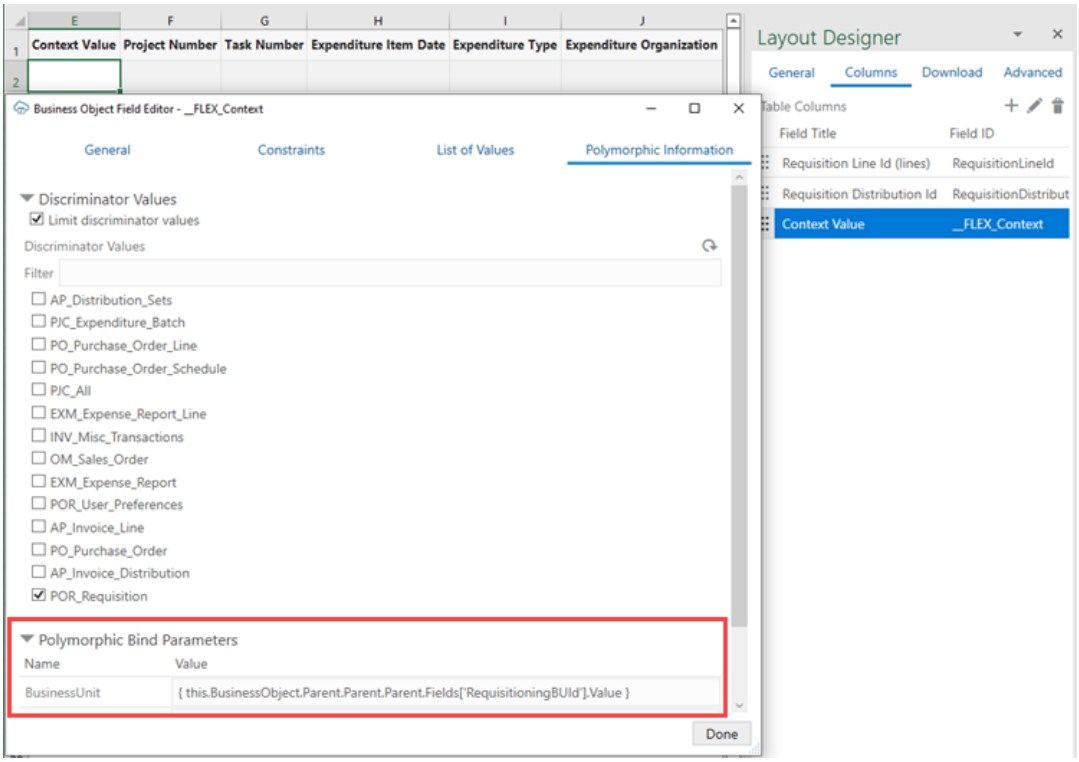
Configure the Bind Parameters for a Descriptive Flexfield's List of Values

Configure parameter-type client binds used by fields in a descriptive flexfield (DFF) using the Business Object Field Editor.

Oracle Visual Builder Add-in for Excel currently only supports configuring the polymorphic bind parameters (also known as "flex bind variables"). Other details of the list of values for DFFs must come from the metadata and are not configurable.

To configure a bind parameter:

1. From the Layout Designer, click the **Columns** tab and then select the polymorphic field set (the Context Value column in this example).
2. Click the **Edit** icon (✎) to open the **Business Object Field Editor**.
3. From the **Polymorphic Information** tab, configure the bind parameters, as shown in the following image:



   This example uses a bind parameter, **BusinessUnit**, for the list of values based on the value of the `RequisitioningBUId` field.

The value of a bind parameter can be a literal value like `100001`, `Requisition001`, or a string that contains one or more expressions surrounded by curly braces (`{ }`).

An expression can refer to the value of a field in any business object in the current business object hierarchy. The field referred to must be exposed in an existing layout.

In this example, the value is a single expression, where

- *this* is the polymorphic field set
- *this.BusinessObject* is the business object that owns the polymorphic field set (the DFF business object)
- *Parent* gets the parent business object in the hierarchy (bottom-up: `RequisitionDistributions/projectDFF`, `RequisitionDistributions`, `RequisitionLines`, `PurchaseRequisitions`)
- *this.BusinessObject.Parent.Parent.Parent* is the PurchaseRequisitions business object that owns the field. *RequisitioningBUId*
- *Fields['<field ID>'].Value* gets the current field value of the field with the given field ID. Note the single quotes (required) around the ID.

> **Note:**
>
> Expression syntax is limited to what's shown in the example (key parts: *this*, *BusinessObject*, *Parent*, *Fields[<field ID>].Value*). See About Expressions for more information.

Clear Cache for a List of Values

A list of values' choices are always cached in the workbook.

The cache for a list of values based on a business object can contain up to 300 items, plus all used items. It is populated during the first download or the first time the search-and-select window is used. The search-and-select window shows the cached list of values, if available. An upload also uses cached data.

As a workbook developer, remember to click **Clear List of Values Cache** from the **Advanced** menu:

- Whenever you change any list of values configuration, and always when you publish the workbook
- When the service host is changed.
- When cached data is not what the user expects.

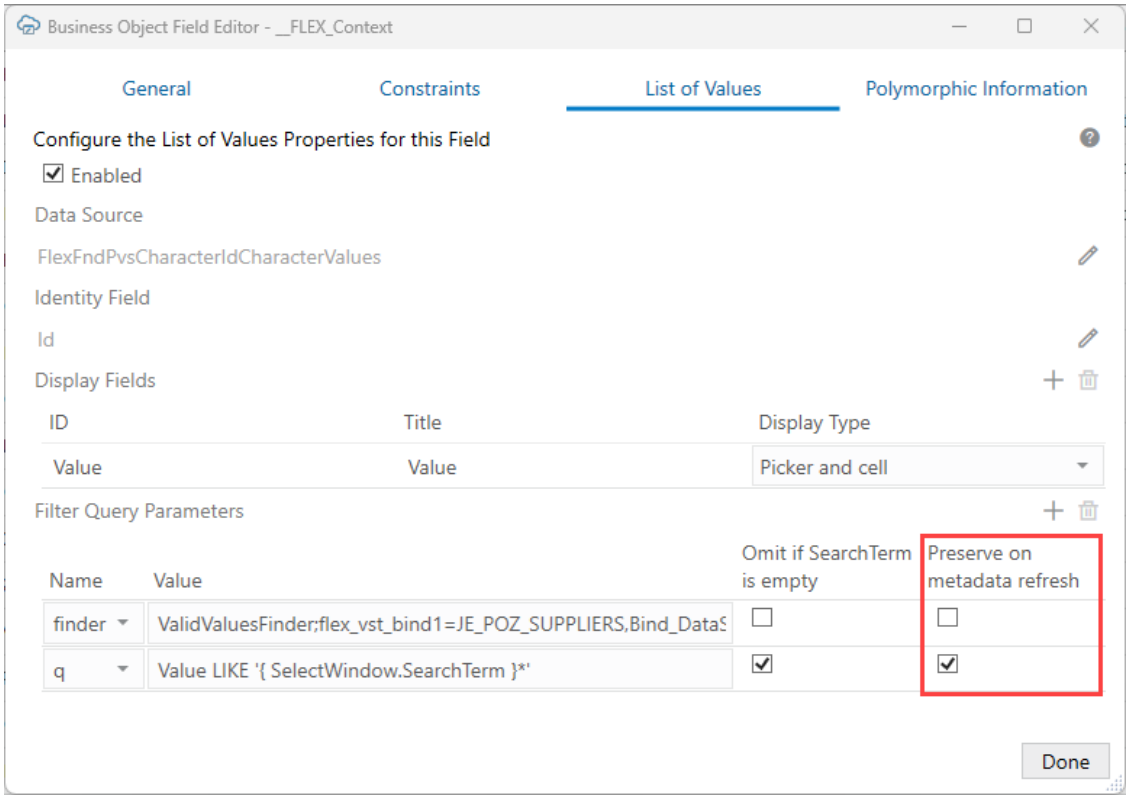
Refresh Parameter Definitions for a Lists of Values

When you create a list of values (LOV), you can configure whether or not Oracle Visual Builder Add-in for Excel refreshes the LOV's query parameter definitions used in an LOV filter.

The add-in refreshes service metadata, including LOV query parameter definitions, when you refresh a catalog from the Manage Business Object Catalogs window. See Refresh a Business Object Catalog.

It also refreshes polymorphic metadata, including LOV query parameter definitions on polymorphic fields, on the first download after opening a saved workbook. See Refresh Polymorphic Business Object Metadata.

Whether this metadata is maintained or discarded on refresh is controlled by the **Preserve on metadata refresh** check box on the List of Values tab.



If **Preserve on metadata refresh** is enabled, the add-in keeps the existing LOV query parameter definition during the refresh.

If **Preserve on metadata refresh** is disabled, the add-in instead discards the existing definition in favor of the new version from the service. Keep in mind that any changes you have made to the parameter definition—such as the **Value** and **Omit if SearchTerm is empty** settings—is replaced by a copy from the service.

> **Note:**
>
> If the **Preserve on metadata refresh** check box is deselected but there is no parameter definition defined in the service metadata, the add-in removes it from the LOV.

Preserve on metadata refresh is enabled by default when you configure a filter query parameter for an LOV.

Notes and Limitations for Lists of Values

Review the notes and limitations listed here when planning to use lists of values (LOV):

- Only one identity field is supported in a list of values.
- LOVs support strings, Boolean values, and integers. Decimal numbers, dates, and date-time values are not supported.
- Excel drop-down lists using data validation may not be compatible with the add-in.

- For ADF REST services, if the service metadata includes information for an LOV, the add-in reads the metadata and configures the LOV automatically. Supported ControlTypeHints: "combo_lov", "choice", and "input_text_lov".
- For any read-only field (column or form), the add-in still swaps identity values for display values. However, the search-and-select window does not appear when the cell is selected.
- When using an expression that refers to a field in an ancestor layout, ensure:
  - The referenced field is present in the ancestor layout.
  - The ancestor columns in the current layout are configured properly even when you don't expect to create rows. See Add Ancestor Columns to Dependent Layouts.
- A field's LOV can be dependent on an ancestor field as long as that ancestor field does not have an LOV that is dependent on another field.
- Dependent LOVs are not supported on ancestor columns.
- When creating a cascading set of LOVs, you can use a "local" LOV (one based on a local data source) as the initial list, but not as one of the subsequent lists. A "remote" LOV (one based on a business object as the data source), however, can depend on a value in a local LOV.
  In a cascading list of values, the value selected in one list determines the range of values that users can select from subsequent lists.

10

Key Flexfield Support

Oracle Visual Builder Add-in for Excel supports using key flexfield (KFF) fields in a layout. When you include these fields, the add-in displays a contextual popup or "picker" when a business user selects the cell.

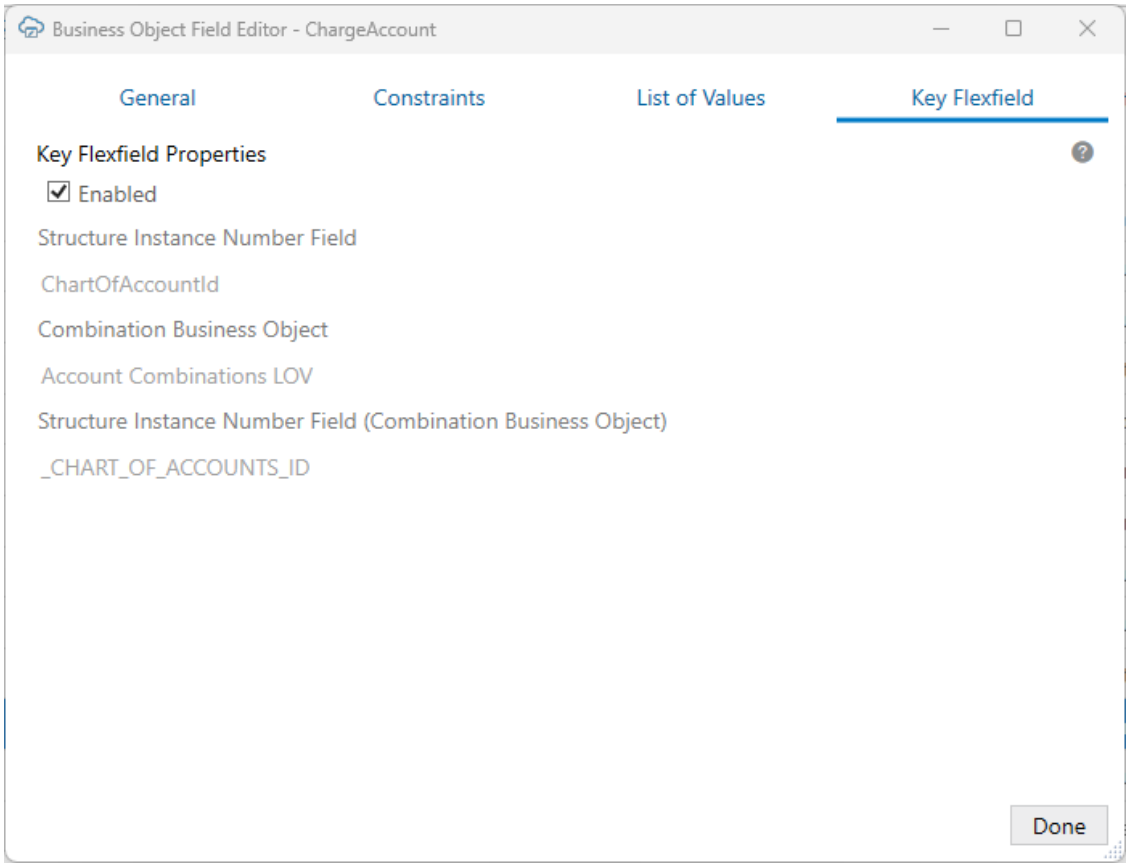
About Key Flexfields

A key flexfield is a polymorphic field that consists of one or more segments where each segment can be used to capture a different part of a key. These key flexfields may be used for keys such as a part numbers, job codes, or account codes. Key flexfields are a feature of ADF REST services. See Key Flexfields in *Implementing Applications*.

Key Flexfield Configuration

If your service includes key flexfield fields, the add-in can extract the required details and configure the key flexfield without the need for any manual setup.

You can review the configuration details for a key flexfield field from the Business Object Field Editor's **Key Flexfield** tab.



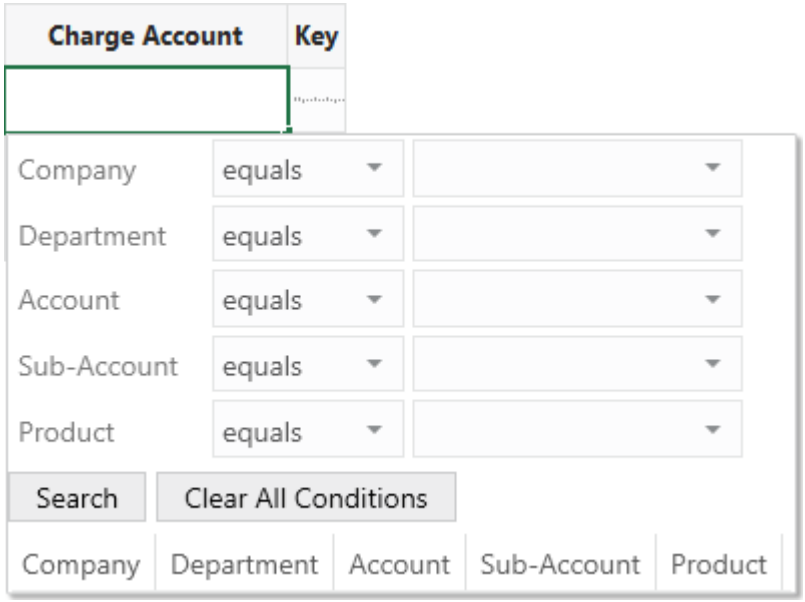
This tab displays these details about the key flexfield:

- **Structure Instance Number Field**: The field from the consumer business object that determines the structure of the key flexfield
- **Combination Business Object**: The polymorphic business object that provides the combinations (keys) for the key flexfield
- **Structure Instance Number Field (Combination Business Object)**: The field from the combination business object that determines the structure of the key flexfield

If required, you can disable the picker by deselecting the **Enabled** check box. When disabled, the picker does not appear. Instead, the cell is treated like a normal string-based cell and the business user must type in a value manually.

The Key Flexfield Picker

A key flexfield picker allows the business user to search for and select an appropriate value rather than having to type the value in from memory. This image shows a picker for a charge account field that includes a segment for each component of the number such as `Company`, `Department`, and so on.



To enter a charge account number for a row, the business user would create queries on one or more segments, then use **Search** to show numbers that match the filter. For more information about how to use the picker, see Edit Downloaded Data in the Workbook in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Add a Key Flexfield Field to a Layout

After you create a layout with a key flexfield field, you'll want to ensure that both the field and the Structure Instance Number field are included in your layout. The content of the picker is dependent on the value of the Structure Instance Number field. If you are unsure which field this is, check the Business Object Field Editor's **Key Flexfield** tab.

To add these fields to your form or table, see Manage Fields in a Form or Table.

> **Note:**
>
> To ensure the Structure Instance Number field has an appropriate value, consider configuring a list of values on a related field (such as Ledger) and include the Structure Instance Number field as a "driven field". See Configure a List of Values to Populate Related Fields.
>
> If the Structure Instance Number field is missing from the layout or the current value is null or invalid, the picker is not displayed.

Notes on Key Flexfields

The following are the requirements for key flexfield support:

- Key flexfields are specific to ADF REST services.
- This feature is relevant to key flexfield-based fields. Key flexfield-based child business objects are not included.
- The key flexfield field must have the `x-oj` hint in its OpenAPI service metadata, like this: `"controlType": "oj-sp-input-key-flex-field"`
- Every segment must have the proper LOV-related hints in the OpenAPI service metadata.
- The structure instance number field must use the integer data type.

The following features are not supported:

- Dynamic lists of values between segments
- Aliases for segment values
- Default segment values
- Validation rules
- Dynamic combination insertion

11

Appearance of an Integrated Excel Workbook

Oracle Visual Builder Add-in for Excel automatically manages the appearance of an integrated Excel workbook through built-in styles and data format types.

The add-in automatically formats cells in a workbook by creating a set of named Excel styles when the workbook is first initialized. The styles created by the add-in are consistent with Oracle accessibility guidelines. The format portion of the styles is sensitive to the user's preferences as defined in the Windows region settings. So a US user will see US dates and a French user will see French dates.

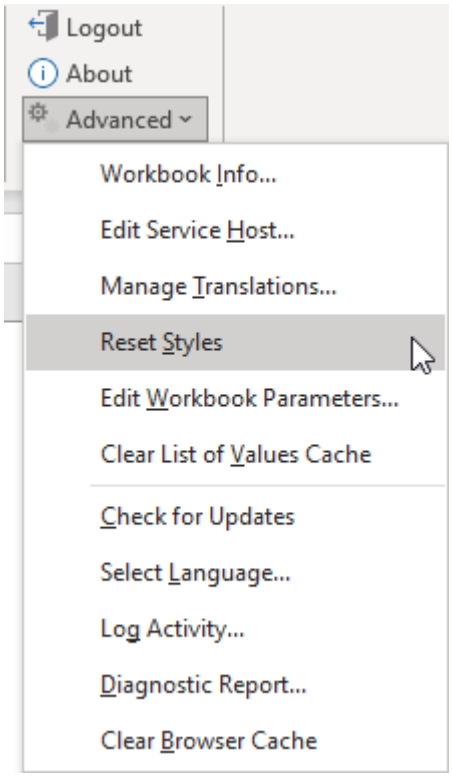
Once the styles are initialized, the add-in applies those styles to the integrated cells, according to the field's properties, at key points (such as during a data download). Typically, the styles are created once for a workbook and are never updated, but you can take advantage of the options described in this chapter.

Reset Workbook Styles

You have the option to reset styles in an Excel workbook when a new add-in version has updated style definitions and you want to use those styles in an older workbook, or when a user has changed style definitions in a workbook and wants to revert to the current definitions.

To reset the style definitions in a workbook, select **Reset Styles** from the Advanced menu.

> **Note:**
>
> The **Reset Styles** command is only available when the designer tools have been enabled.



You are prompted to confirm. After confirmation, each style definition used by the add-in is updated. The list of styles includes any style that would be created by the current add-in, including the Normal style.

Choose Field Formats

Cell formats in an Excel workbook are applied by the add-in according to a field's data type and based on whether the field is editable (create or update). While this behavior works for most scenarios, sometimes you might want to extend a field's formatting options.

Let's consider the example of an employee workbook that includes ID and Salary fields, both of which are integers. A value of 10,000 for the Salary field can be formatted with the thousands separator (10,000 versus 10000). But an ID value of, say, 12300 seems odd with the thousands separator (12,300). In this case, you can override the ID field's default format to choose a format without the thousands separator.

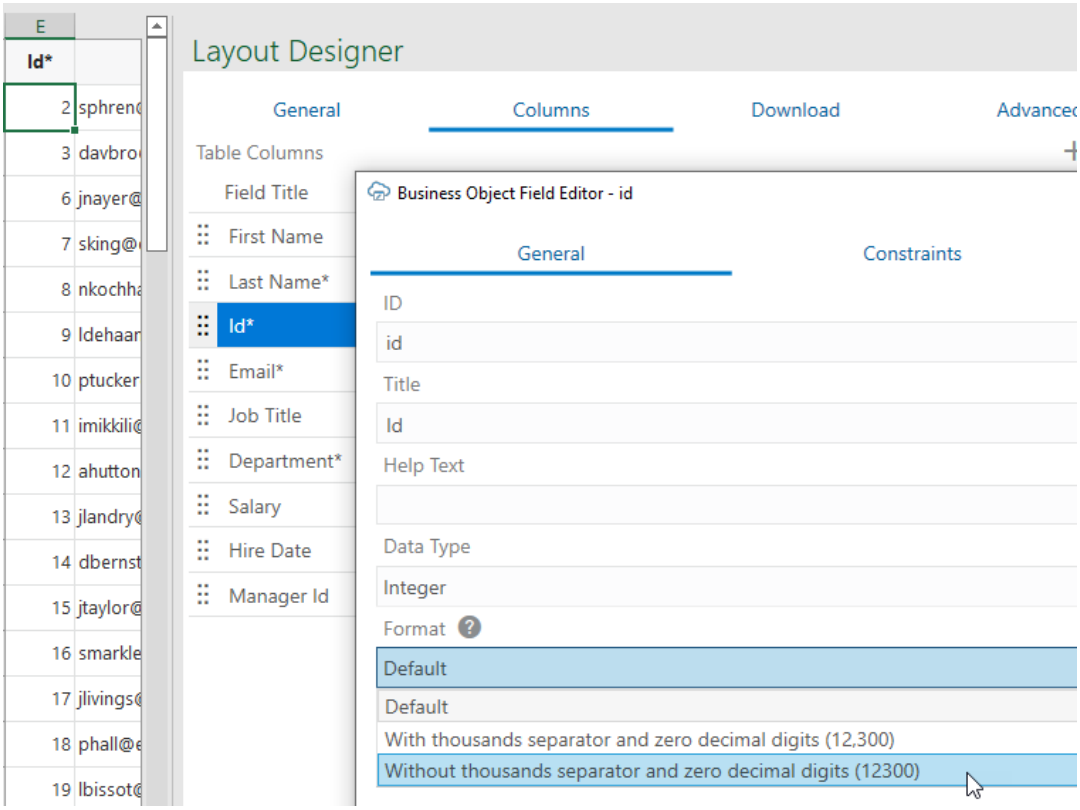
You can also configure a number field to show the maximum number of decimals for greater precision—rather than to zero or two decimal places which would be more appropriate for something like a dollar value. To show a number like pi (π) without truncating or rounding to two decimal places, choose the **With thousands separator and maximum decimal digits** option. π would appear in this case as "3.1415926535898" rather than "3.14".

> **Note:**
>
> Microsoft Excel's limit for numbers is 15 significant digits.

To override the default format for a field:

1. In the Layout Designer's Columns tab, select the field you want to update and click the Edit icon.

2. In the Business Object Field Editor that appears, select an option in the **Format** drop-down. The default value is `Default`, which tells the add-in to apply the usual automatic styles without any override.
   Remember, the format types shown to you are based on what's appropriate for the field's data type. In this image of an employee's workbook, the ID field uses the Integer data type and the available options let you format the ID with or without the thousands separator. The options also follow user preferences as defined in the Windows Region settings, so a US user will see a decimal point (π = 3.14) while a French user will see a decimal comma (π = 3,14).



3. Click **Done**.
4. Click **Download Data** to process the change.

Add Help Text to Your Workbook

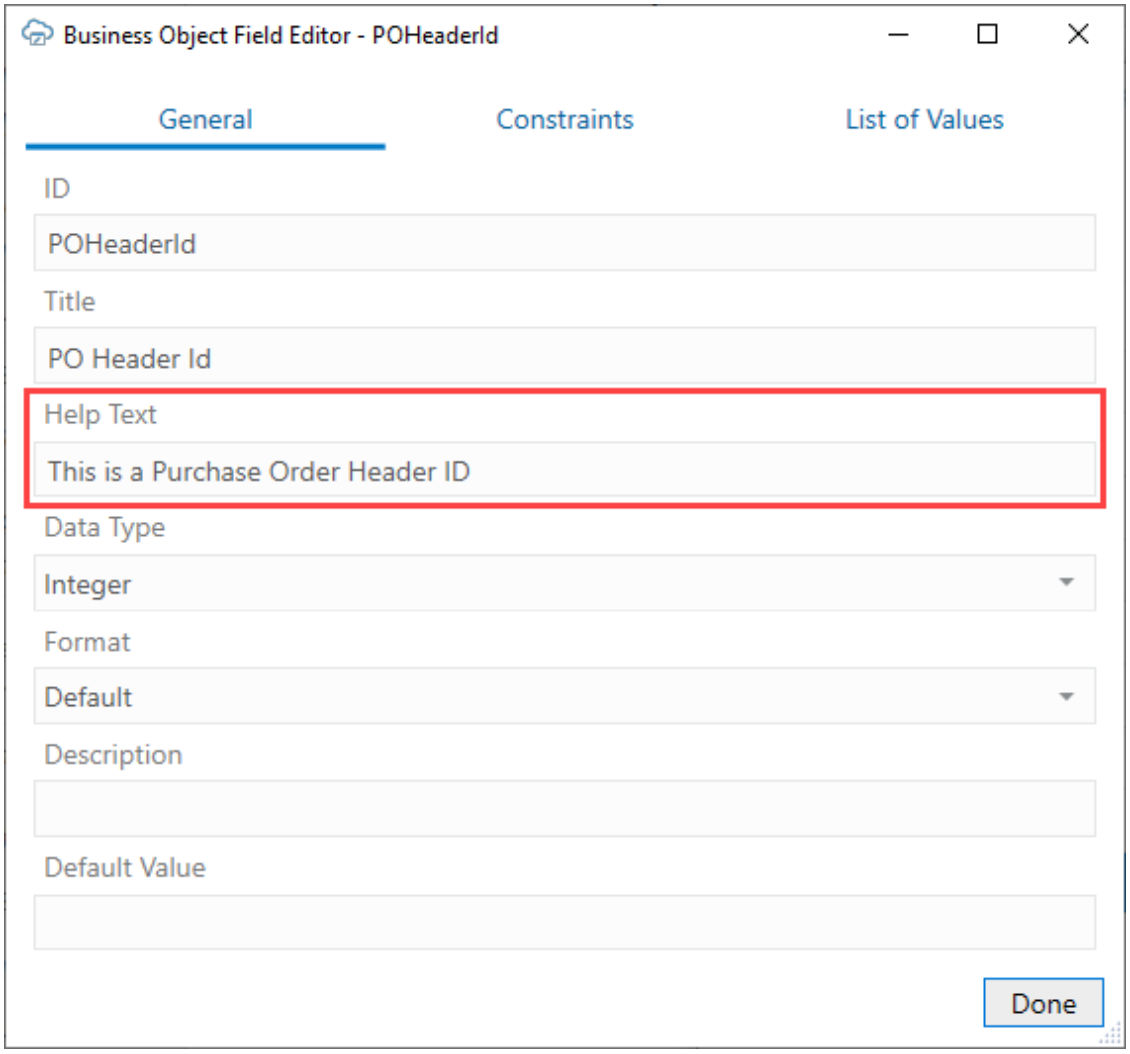
You can add help text for business object fields you use in forms and tables in your workbook to provide help to your business users. Your users will see a popup with this help when they select a form field or column header in a layout.

You can also add help to:

- Custom actions
- Custom action payload fields
- Row finders
- Row finder variables
- Row variables

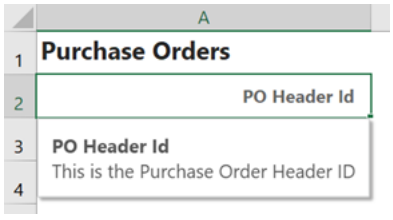
You can add help text for an individual field using the editor for that field. You can also copy descriptions included in the service metadata document to the relevant Help Text properties. For the steps to copy descriptions to help text, see Copy Descriptions to Help Text.

To add help text for an individual field, open the editor for the field and type help text in the **Help Text** field on the **General** tab, as shown here:



Help Text values, as well as **Title** values, can be localized. See Manage Workbook Translations.

Once configured, help text for business object and item-level custom action payload fields is displayed when the user selects the form field label or table column header in a layout.



Help text for custom actions and business object-level payload fields is displayed in the custom action wizard, either in the **Description** field for custom actions or in a popup when you hover over the payload field. Help text for row finders and variables is displayed in the Search Editor when you hover over the finder or variable label.

Copy Descriptions to Help Text

If desired, you can copy the Description property values to the Help Text properties for a given catalog.

When Oracle Visual Builder Add-in for Excel creates a catalog, it can harvest descriptions from the service's metadata and populate the description properties for various elements such as business object fields, custom actions and their payload fields, and row finders and their variables.

These descriptions may be technical in nature and might not be suitable for business users. As a result, these descriptions are not displayed to business users and are not "localizable".

OpenAPI does not specify any property that corresponds to "help text". Furthermore, none of the supported REST frameworks provide help text or "tooltips" using x-hints, for example. As a result, the Help Text properties are empty for new catalogs.

If your REST service provides business-user friendly descriptions in the metadata, you can copy these values to the help text properties in one go.

Before you begin, make a backup copy of the source workbook in case you don't like the results.

To copy descriptions for a catalog to the help text properties:

1. Click **Manage Catalogs**.
2. From the Manage Business Object Catalogs window, select the desired catalog from the list.
3. Click the **Menu** icon (☰), then select "Copy Descriptions to Help Text" from the popup menu.
4. When prompted to confirm the operation, click **Yes**.
5. When the copying is complete, review the Help Text values in the catalog to make sure they are appropriate for your workbook's users.

> **Note:**
>
> You may need to remove markup since the help text values are displayed as is.

When you've copied the descriptions over and refined the Help Text values as you see fit, you might want to consider translating these text strings into other languages. See Manage Workbook Translations.

12

Data Validation

Oracle Visual Builder Add-in for Excel provides data validation during data entry based on business object field, custom action payload field, and row variable definitions as well as custom validation rules. Data validation catches errors during data entry when it's easy to correct a mistake rather than after a failed upload attempt.
When a business user works with data in a workbook, the add-in validates data in new and updated fields and raises a data entry error if it detects an invalid value. Business users will need to fix all data entry errors before they can upload successfully.

Cells that fail validation are marked with a red border. Rows that contain validation errors are also flagged as `Invalid` in the **Status** column. The add-in displays an error message in a popup if the business user selects the cell with the error.

The add-in validates data at these key points when the business user is working in a layout:

- When a row is added to the form or table of a layout. All validations are checked at this time for this row.
- When the business user completes an edit on a field. At this point, the add-in validates all editable fields in that row.
- At the beginning of an upload

Local field properties are used for validation. See Configure Business Object Fields.

> **Note:**
>
> Validation that is enforced by the REST service is not triggered at the points mentioned here. REST service validation is generally triggered by the requests sent during upload. For details on upload failure handling, see Upload Changes.

When a row is marked for a custom action, the custom action's payload fields also receive the same validation. See Custom Actions.

Here are the validation conditions the add-in checks for:

Condition	Description
The entered value matches the data type of the field.	If a value doesn't match the data type, the add-in displays a popup with the validation failure message. For example, if the business user enters a string, such as `one thousand`), in a field with a `Number` data type, the add-in displays the message: "`The value is not valid for the expected data type: Number`". Data types include `String`, `Date-Time`, `Integer`, `Number`, and `Boolean`. A field's data type is defined in the Business Object Field Editor for the field. See Configure Business Object Fields.
A value has been entered for a field that is required for update or create.	If no value is entered, the add-in displays a popup with the validation failure message. During table or form row creation, the add-in automatically flags all required fields. Whether a field is required for update or create depends on the **Required for update** and **Required for create** check boxes on the **Constraints** tab of the Business Object Field Editor. See Configure Business Object Fields.

Condition	Description
A value entered for a string-based field meets the minimum and maximum lengths configured for the field	If a value is entered that has too few or too many characters, the add-in displays a popup with the validation failure message. Minimum Length and Maximum Lengths are set on the **Constraints** tab of the Business Object Field Editor. See Configure Business Object Fields. **Note:** When comparing the length of the field value with the minimum and maximum lengths, the add-in counts Unicode code units (encoded as UTF-16). Such a count usually matches what the business user expects. However, in cases involving Unicode extended characters (graphemes and supplementary code points), a single text element can require multiple code points. In these cases, a field value might be considered invalid when it appears to have fewer visible or voiced characters than the configured maximum length.
The value of a field with a list of values is a value available from that list.	If, rather than selecting a value from the list, the business user types in a value that is not in the list, the add-in displays a validation failure message.
The value of the field matches the criteria configured for the field in a custom field validation rule.	A custom field validation rule is an expression that restricts the range of allowable values during data entry. When a business user enters a value, the add-in evaluates the expression based on the value and, if the expression evaluates to true, the value is judged to be valid. If the expression evaluates to false, the value is invalid, and the add-in displays the validation failure message configured for the rule. See About Field Validation Rules.

A Note on Data Validation Interactions

Note that a field may have two or more data validation constraints set that affect how the add-in validates a cell during data entry. Suppose you have a field that is required for update and also has a minimum length of 10. If a business user clears the value, the add-in first evaluates the cell based on the **Required for update** setting. In this case, the cell is marked `Invalid` and a popup validation message is displayed (`A value is required`). The minimum length setting is not considered.

If a value is *not* required for update, a null value is now considered valid. However, when it is evaluated based on the minimum length constraint, it is considered invalid and a different popup validation message is displayed (`The value must have at least 10 characters`).

About Field Validation Rules

A field validation rule is an expression you define for a business object field or custom action payload field that restricts the range of allowable values during data entry. When a business user enters a value in a form field or table cell, Oracle Visual Builder Add-in for Excel checks the value against the configured rule and raises a data entry error if the value does not match the set criteria.
These rules can catch invalid values during data entry when it's easy to correct a mistake rather than having to find errors during upload.

Field validation rules are expressions using the add-in's expression language that evaluate to true or false. When a business user enters a value, the add-in evaluates the expression based on the value and, if the expression evaluates to true, the value is judged to be valid. If the expression evaluates to false, the value is invalid, and the add-in displays a popup with the validation failure message.

You can define a rule that compares the value the business user enters with a constant or with the result of an expression involving other field values in the same row. Expressions must compare values that match the field's data type. For example, an expression on a integer field such as `{ this.Value <= 500 }` compares an integer entered by the business user (`this.Value`) with an integer constant (`500`).

Validation rules are supported on business object fields, row variables, and custom action payload fields.

Supported Expression Syntax

When constructing a field validation rule, these expressions are supported:

- `this.Value` returns the current field value, where "`this`" is a regular business object field or custom action payload field.
- `this.BusinessObject.Fields['<FieldID>'].Value` retrieves another field value in the same row as the current field, where:
  - `this` refers to the current field
  - `BusinessObject` refers to the business object (the same row) of the current field
  - `Fields['<FieldID>']` refers to the ID of a field (<*FieldID*>) in the set of fields for the business object. The field must be shown as a column in the layout.
  - `Value` refers to the value of the given field (<*FieldID*>)

- `this.CustomAction.PayloadFields['<FieldID>'].Value` gets a given payload field value (`PayloadFields['<FieldID>'].Value`) in the same custom action (`this.CustomAction`).

The field validation rule does not support:

- The `Parent` keyword (for example, `this.BusinessObject.Parent.Fields['<FieldID>'].Value`) to get a field value from a parent or other ancestor business object
- Accessing field values from child and other descendant business objects
- The `CustomActions` keyword (for example, `this.BusinessObject.CustomActions['CustomActionID']`) to access a custom action from a regular field

Field validation rules support the `Today()` and `Now()` functions.

Sample Validation Rule Expressions

Field validation rules use the add-in expression language. See About Expressions for more information.

> **Note:**
>
> When creating expressions for a field validation rule, keep in mind that empty cells may produce undesirable results. For information about handling null values in expressions, see Handling Null Values in Expressions.

Expression	Use
`{ this.Value <= 500 }`	This rule compares the value in the cell (`this.Value`) to a constant (`500`) and displays a validation failure message if the value is more than this amount. You would use this rule for the Amount field of an Expenses layout to limit the amount for each expense in an expense report to $500 or less.
`{ this.BusinessObject.Fields['UnpaidAmount'].Value > 10000 ? this.Value == 'Group 1' : this.Value == 'Group 2' }`	This rule checks the value in an Invoice Group cell (either `this.Value == 'Group 1'` or `this.Value == 'Group 2'`) to ensure the correct value is provided based on the value in the Unpaid Amount cell (`this.BusinessObject.Fields['UnpaidAmount'].Value > 10000`) in the same row. If the unpaid amount is over $10,000, then the invoice group should be Group 1; otherwise, the invoice group should be Group 2. If the wrong group is selected, the add-in displays a validation failure message. You would use this rule for the Invoice Group field of an Invoices layout to require the business user to enter the correct invoice group based the unpaid amount of an invoice.

Expression	Use
`{ this.Value > 0 && this.Value < 50 }`	This rule checks the value in the cell and displays a validation failure message if the value is outside a given range (between `0` and `50`). You would use this rule for a Commission Percentage field to ensure the commission is larger than zero but less than 50%.
`{ (this.Value - this.BusinessObject.Fields['OldSalary'].Value) / this.BusinessObject.Fields['OldSalary'].Value * 100 < 5 }`	This rule compares the value in the cell (`this.Value`) to the value in another cell (`this.BusinessObject.Fields['OldSalary'].Value`) and displays a validation failure message if the increase is 5% or greater. You would use this rule on the New Salary field to ensure an employee's raise is within your company's salary cap.
`{ this.Value < Today() }`	This rule compares the date value in the cell to today's date and displays a validation failure message if the date is after today. You would use this rule on the Hire Date field to ensure a future date isn't used by mistake.
`{ this.Value > Now() }`	This rule compares the date-time value in the cell to the current moment and displays a validation failure message if the date and time is after the current moment. You would use this rule on the Deadline field to ensure a future date is used.

Create Field Validation Rules

You can define field validation rules to ensure your business users are entering valid values when they create or update a row or form in a layout.

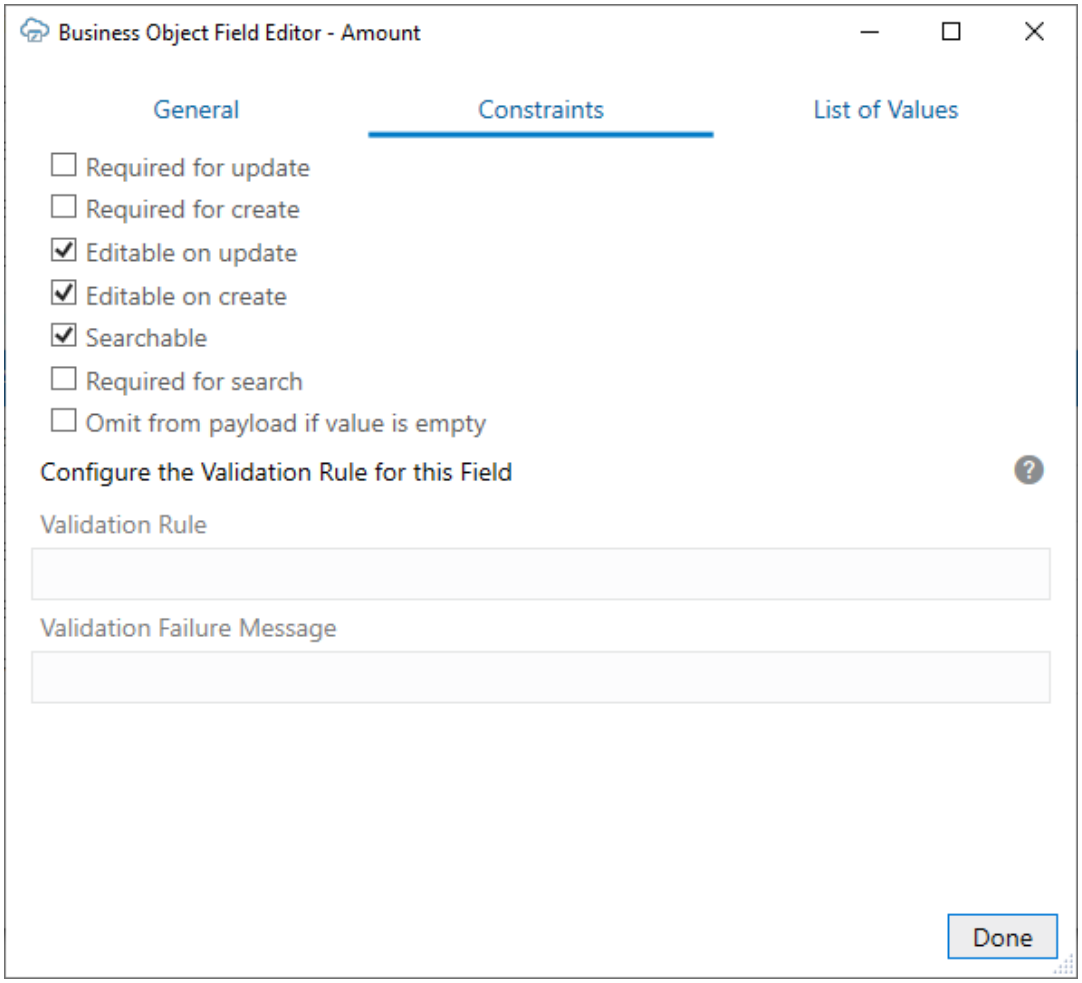
For example, you may want to limit the amount for each expense in an expense report to $500 or less. To do this, you can enter a rule for the "Amount" field like this: `{ this.Value == null ? true : this.Value <= 500 }` where `this.Value` refers to the value of the currently-selected cell. During data entry, the add-in marks the row as "Invalid" if the value entered exceeds $500.

If the business user selects the cell with the error, the add-in displays a popup with a description of the error. You can also provide a custom error message when you define a rule.

> **Note:**
>
> This example expression uses a conditional to set the value of the expression to "true" (`this.Value == null ? true`) if the selected cell is empty (null).
>
> For help on the add-in expression language and null handling in expressions, see About Expressions and Handling Null Values in Expressions.

To create a field validation rule:

1. Open the **Business Object Field Editor** of the field you want to set a validation rule for, then click the **Constraints** tab.



> **Note:**
>
> For a custom action payload field, navigate to the **Custom Action Payload Field Editor** instead.

2. Type in your validation rule in the **Validation Rule** field using the add-in expression language. The expression must comply with the add-in's expression language and evaluate to true or false.

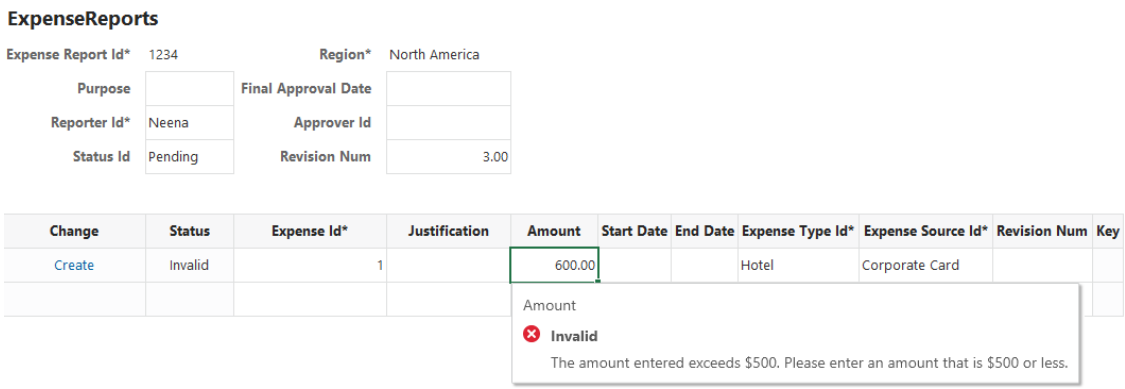
   For example, to limit the value in an Amount field to $500 or less, type:

   ```
   { this.Value == null ? true : this.Value <= 500 }
   ```

3. To provide a custom error message, type your message in the **Validation Failure Message** field.

   Use this field to provide a brief explanation for your business users of what values are expected for this field. This value can be localized.

At runtime, if a business user enters a value that violates the validation rule, the add-in marks the row as invalid and displays a red outline around the invalid cell. If selected, the add-in displays an error popup with the validation failure message.



Notes on Custom Field Validation Rules

Here are some things to keep in mind when creating custom field validation rules.

Supported Fields

Fields with these data types are supported:

- `Boolean`
- `Date (no time)`
- `Date-time`
- `Integer`
- `Number`
- `String`

Unsupported Fields

These fields are not supported:

- Fields with `Object` or `Unsupported` data types
- Fields with a configured list of values. The add-in already performs validation on these fields to ensure the value is a valid entry from the list.
- Discriminator fields from a polymorphic business object
- Ancestor fields

Limitations

- A rule expression for a field cannot refer to a field value from a different row or from a different layout.
- Multiple validation rules on one field are not allowed. However, you can use logical operators, `&&` and `||`, in one validation rule to the same purpose.

Validation Behavior

- Validation rules are not evaluated on download. So, if the downloaded data includes values that violate the rules, the violations are not highlighted after download completes.
- When a cell is modified, all editable cells in that row are validated.

13

Upload Changes

When you are done editing data in an Excel workbook, you are ready to upload the changes to the REST service.

Upload Changes from a Table Layout

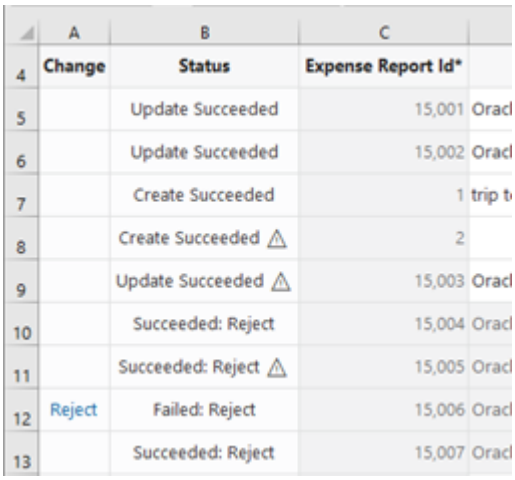
When you click the **Upload Changes** button for a Table layout, here's what happens:

1. The add-in checks the table for pending changes. If there are no pending changes, upload is skipped.
2. If a Pre-Upload macro is configured, it is invoked. If the macro throws an exception or returns any value other than `true`, the upload process quits.
3. For pending Create and Update operations, the rows are first validated locally, for example, data type, required, and so on. See Data Validation. Any failures are marked as failed and skipped (these rows do not produce requests on the business object service).
4. Pending changes are generally processed as follows:
    a. Updates result in a PATCH or PUT request on the item path.
    b. Creates result in a POST request on the collection path.
    c. Deletes result in a DELETE request on the item path.
    d. Rows marked for action result in a POST request on the item action path.

    For more details on how rows are sent in separate requests, see Upload Table Changes Using Separate Requests for Each Row.

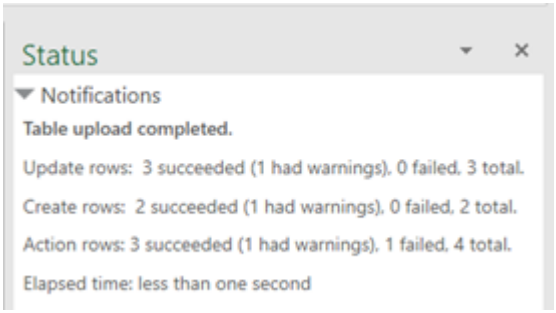
> **Note:**
>
> Multi-Row processing is handled differently. See Upload Changes Using Multi-Row Requests.

5. Success and failure is noted in the Status column. Rows with warnings are indicated with a warning icon.



	A	B	C	
4	**Change**	**Status**	**Expense Report Id***	
5		Update Succeeded	15,001	Oracl
6		Update Succeeded	15,002	Oracl
7		Create Succeeded	1	trip t
8		Create Succeeded ⚠	2	
9		Update Succeeded ⚠	15,003	Oracl
10		Succeeded: Reject	15,004	Oracl
11		Succeeded: Reject ⚠	15,005	Oracl
12	Reject	Failed: Reject	15,006	Oracl
13		Succeeded: Reject	15,007	Oracl

Errors and warnings are cached and displayed in the Status Viewer when the business user selects a row. Here is a sample of the Status Viewer showing three warnings on 8 successful changes.



6. Successful Create rows are updated from the service if possible. These rows are converted into existing rows that can be edited.

> **Note:**
>
> If the service does not return the newly created rows, you will need to download them before doing any further editing.

7. Successful Delete rows are removed from the Excel worksheet.

For Create and Update operations, the request payload only includes values for editable cells in a new or updated row. Read-only fields and custom action payload fields are not included. Whether a value is included for an editable cell depends on your add-in settings:

- For updated rows, the request payload includes a value *for each editable field* unless **Send Only Changed Data for Updates** is enabled in the Layout Designer. When enabled, only values that have changed since the last download or upload are included in the payload. See Send Only Changed Data During Upload.
- For new and updated rows, the request payload includes a null value for each empty cell in the row unless **Omit from payload if value is empty** is enabled for a field in the Business Object Field Editor.

When this check box is enabled and the corresponding cell is empty, the field is not included in the request payload. When this check box is not checked and the corresponding cell is empty, the field is included in the request payload with a null value. See Omit Empty Values During Upload.

The add-in may send up to four requests to the service at a time for improved performance (on different threads). As a result, the order in which the rows are sent to the server is non-deterministic. It is not guaranteed to be in the same order as in the table. See Enable Parallel Requests During Upload.

Upload Changes from a Form-Over-Table Layout

When a user clicks the **Upload Changes** button for a Form-Over-Table layout, here's what happens:

1. The add-in checks the form for a pending Update or pending Create operation and the table for pending changes. If there are no pending form or table changes, upload is skipped.
2. If a Pre-Upload macro is configured, it is invoked. If the macro throws an exception or returns any value other than `true`, the upload process quits.
3. When the form has a pending Update or pending Create operation:
   For a pending Update:
   a. A GET request is sent to the parent's item path.
   b. Form field values (for example, data type, required, and so on) are validated; read-only fields are skipped. If validation failures exist, the upload is stopped and no subsequent REST requests are made.
   c. When all form fields are valid, a request (PATCH/PUT) is sent to the parent's **item** path. The payload for this request contains the values of all form fields that have been changed. If configured, empty values are skipped. See Omit Empty Values During Upload.

   For a pending Create:
   a. Form field values are validated based on properties such as data type, required, and so on; read-only fields are skipped.
   If validation failures exist, the upload is stopped and no subsequent REST requests are made. For more information about data validation, see Data validation in *Managing Data Using the Oracle Visual Builder Add-in for Excel*.
   b. A POST request is sent to the parent's **collection** path. The payload for this request contains a value (possibly empty) for every editable form field in the form. If configured, empty values are skipped. See Omit Empty Values During Upload.

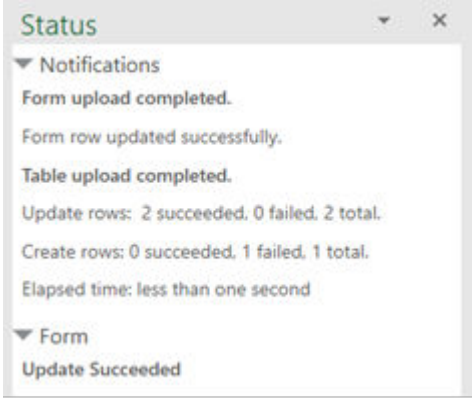
   The child table is not involved in this step.
4. The results of the form upload are reflected in the status viewer immediately.
5. If the form upload fails, the add-in stops and does not attempt an upload on the child table.
6. If the form upload succeeds, the add-in proceeds with the child table as follows:
   a. Checks the child table for pending changes, creates, deletes, custom actions, etc.
   b. If changes are found, the child table upload proceeds in the same manner as a Table layout upload with one important difference: the child business object's paths are used for each request.

The number of requests used in the upload depends on the scope of the changes as well as the state of the **Send Descendant Rows in Parent Payload** check box. If this check box is

selected, the add-in sends parent changes and all child changes in a single request. See Upload Parent and Child Changes in the Same Payload.

If this check box is deselected *and* the form and table both have pending changes, then the add-in sends the changes using a minimum of two requests: one for the form and one (or more) for the child table.

For the form changes, notification of success and failure is noted in the Status Viewer.



For table changes, success and failure is noted in the table's Status column and well as in the Status Viewer. See Upload Changes from a Table Layout for more information about table status after Upload.

Invoke Custom Actions via Upload

For services that support custom actions that correspond to the item path, the user can mark rows so that the custom action is called on those rows during the upload. For custom actions that require payload fields, table columns that correspond to those fields must be added. See Custom Actions.
During the upload operation (which may also include update, create, and delete tasks), the add-in performs the following steps for each row marked for action:

1. If any values are invalid (missing when required, incorrect data type, Excel formula error), the row is omitted from the Upload and marked as failed. See Data validation in *Managing Data Using Oracle Visual Builder Add-in for Excel*
2. Creates the payload by collecting the cell values for each custom action field column and adding the value to a simple JSON object (member name/value pairs) in the payload. If configured, empty values are skipped. See Omit Empty Values During Upload.
   The entire payload body follows this example format:

   ```
   {
     "rejectionReasonCode": "Other: contact approver",
     "notes": "Details with manager"
   }
   ```

   There is no other content in the POST request body (no action name, no array of argument values).
3. Prepares the request:
   - REST-Framework-Version header added (see Configure the REST-Framework-Version)

- Content-Type header added based on each custom action's Request Media Type property on the **Custom Action Editor** (available from the **Business Object Editor** > **Custom Actions** tab)

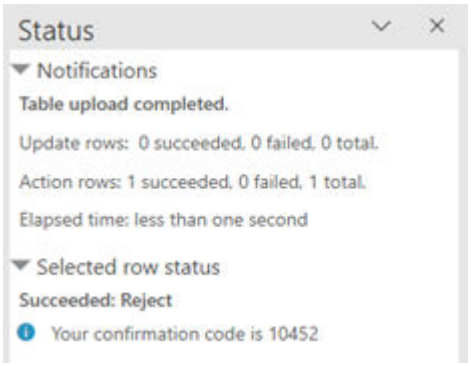
4. Makes the request:
   - Sends the POST to the custom action path (POST is the only HTTP method supported for invoking custom actions)
5. Processes the response:
   - For 200 response status, the row is marked as succeeded.
   - For 400 response status, the row is marked as failed and the response payload is parsed for ADF REST service error content. Error details can be seen in the Status Viewer pane.
   - A 412 response status indicates that the row was modified by some other agent or user after it was downloaded to the Excel table; such a status is treated as a row-level error

   Cell values in action rows are not refreshed. If the custom method logic in the service has altered any values in the row, those changes will not be reflected in the table row until the next download.

   If the response payload for the Custom Action request includes a "result" member, the add-in parses it as follows and displays the result in the Status Viewer:
   - Simple scalar values are displayed verbatim
   - The first level of JSON objects and arrays are unpacked
   - Deeper levels of JSON are displayed verbatim

   > **Note:**
   >
   > Row status custom actions have their own expected "result" schema. See Row Status Custom Action Response Schema.

   For example, after invoking the custom action "Reject" on an Expense Report table row, the REST service may return a result such as `"result": "Your confirmation code is 10452"`. This result is displayed in the Status Viewer, as shown here:



   See Executing a Custom Action in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

For more information on custom actions, see Custom Actions.

Upload Table Changes Using Separate Requests for Each Row

If the target REST service does not support multi-row processing or multi-row processing is turned off, Oracle Visual Builder Add-in for Excel sends each changed row in a separate request ("row-wise" processing) during upload.
Row-wise processing is also used when either:

- Multi-row support has been disabled by the workbook developer for a specific ADF REST business object.
- When the set of pending changes includes a row marked for a custom action that does not support multi-row requests. See Multi-Row Mode for Custom Actions.

> **Note:**
>
> "multi-row" processing refers to sending multiple rows' worth of changes in a single POST request. This processing is enabled by default on ADF REST services. See Upload Changes Using Multi-Row Requests.

Row-wise Behavior

When a user clicks the **Upload Changes** button (and multi-row processing is not supported or enabled), here's what happens:

1. The add-in divides all changed rows into sequential collections (blocks) of 25 rows each. For example, block 1 has changed rows 1-25, block 2 has rows 26-50, and so on.

   > **Note:**
   >
   > Each block may contain a mixture of kinds of changed rows such as Update, Create, Delete, and custom action invocation.

2. The add-in creates up to 4 background threads, if the **Supports Parallel Requests** check box is enabled, and assigns each thread a block as a unit of work. See Enable Parallel Requests During Upload.
3. Each background thread then processes the block by:
   - Sending a separate REST request for each of the 25 changed rows in the block (`PATCH` or `PUT` for update, `POST` for create, `DELETE`, or `POST` for invoking a custom action).
   - Waiting for the response for each row before sending the next request.
4. When the add-in has received responses from the REST service, it processes the responses for each block, in order.
   The add-in updates the Change and Status columns. If the REST service returns the row's field values in the response, the add-in writes these values for changed rows back into the table rows.
5. The add-in repeats these steps for the entire set of changed rows, processing up to 4 blocks of 25 changed rows at a time, until all changed rows have been processed.

Upload Changes Using Multi-Row Requests

By default, Oracle Visual Builder Add-in for Excel uses multi-row requests to upload changes from a table layout to an ADF REST service.

During an upload operation, the add-in sends multiple rows per request using a batch API. Rows marked for Update, Create, Delete, and custom actions are included in the multi-row requests.

If required, you can adjust multi-row processing settings to improve performance. See Configure Multi-Row Uploads. You can also disable multi-row requests if your service requires a separate request per row. See Disable Multi-Row Requests for Upload.

Multi-row requests are only supported on ADF REST services. For other REST service types, the add-in sends just one row per request. See Upload Table Changes Using Separate Requests for Each Row.

About Multi-Row Processing

Oracle Visual Builder Add-in for Excel uses an algorithm to determine the optimal size of the payload (number of rows) for a multi-row request as the upload operation proceeds. This is done to optimize performance and avoid timeouts.

The add-in's adaptive multi-row upload algorithm for table layouts use two configurable settings: "Target Response Time" and "Initial Row Count". The default target response time is 10 seconds. The default initial row count is 10 rows. For more information about these settings, see Configure Multi-Row Uploads.

Based on these values, the upload algorithm dynamically adapts to the REST service's request-processing performance by sending more or fewer rows for each multi-row request. It works like this:

1. The add-in sends an initial set of multi-row requests using the initial row count.
2. The add-in measures the response times for those initial requests and calculates the approximate number of rows that can be processed in the target response time.
3. Using the new value for number of rows per request, the add-in sends one or more multi-row requests to the service.
4. After receiving the responses from each set of requests to the service, the add-in recalculates of the number of rows, and sends another set of requests until all rows have been processed.

> **Note:**
>
> The target response time is not guaranteed, since actual response times vary depending on the REST service business logic, server load, network latency, and so on.

If a multi-row request contains one or more errors, the ADF REST service rejects the entire set of changes in that request. In that case, a second multi-row request containing only the rows that succeeded during the first request is sent. If the second request fails, the add-in falls back to sending one row per request. For more information, see Making Batch Requests in *Developing Fusion Web Applications with Oracle Application Development Framework*.

If a set of changes includes custom actions and those actions have multi-row requests enabled, the changes are included in a multi-row request. However, if one or more rows is marked for a custom action that has multi-row requests disabled, the add-in reverts to sending the changes row by row. (See also Multi-Row Mode for Custom Actions.)

If you configure the add-in for parallel requests, it sends up to four multi-row requests in parallel. See Enable Parallel Requests During Upload.

If you configure the add-in to send descendant rows in the parent payload, then multi-row requests are not used when a Form-over-Table layout is the primary layout. See Upload Parent and Child Changes in the Same Payload.

Configure Multi-Row Uploads

If required, you can configure how Oracle Visual Builder Add-in for Excel uploads multiple rows of data for an ADF REST business object.

You can access multi-row processing settings from the Upload tab of the business object's Business Object Editor. This tab includes a check box, **Supports Multi-Row Requests**, to enable and disable multi-row requests. This check box is selected by default for ADF REST business objects.

The tab also includes two fields for configuring multi-row processing: **Initial Row Count** and **Target Response Time**.

By default, the add-in's adaptive multi-row upload algorithm uses a target response time of 10 seconds and an initial row count of 10 rows. It is strongly recommended that you do not change the default values for these properties. They should provide good functionality, UI responsiveness, and overall good performance. See About Multi-Row Processing.

However, there may be scenarios where you would want to adjust these properties. For example, your REST service may require that all of a small number of rows be included in a single multi-row request. In this scenario, you could increase the initial row count value something like 100, then instruct your business users to limit changes to sets of 100 at a time.

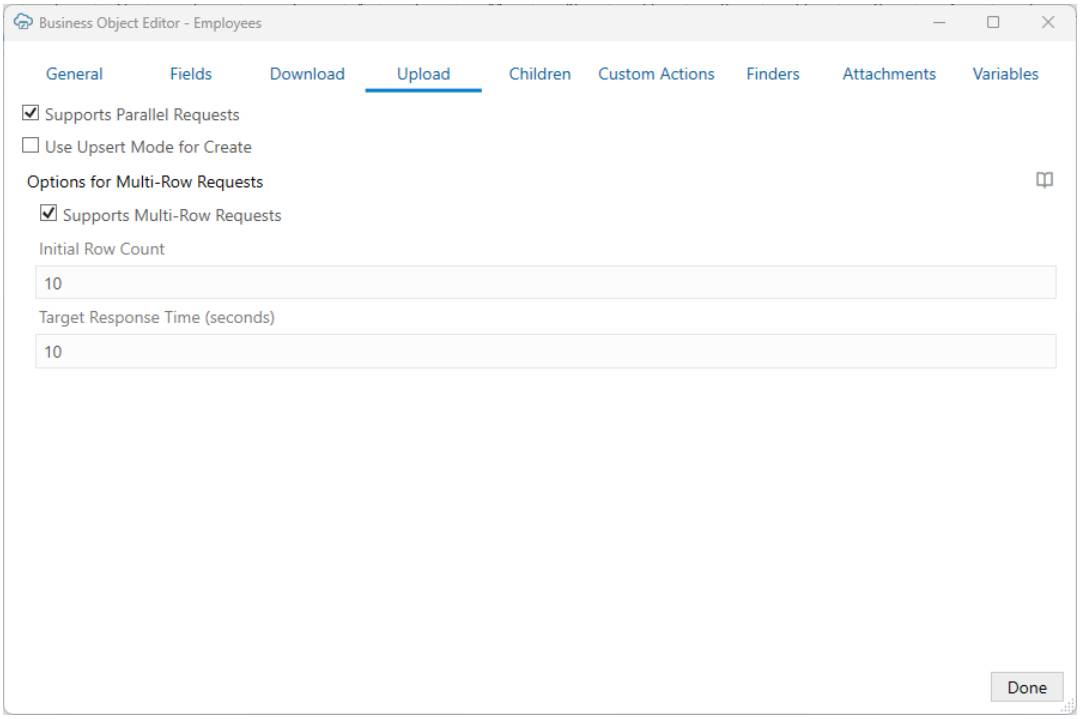
Another example might be a slow REST endpoint where a certain rate for processing rows per minute needs to be achieved. In this scenario, you may want to tune these properties to balance UI responsiveness and the ability to process a targeted number of rows in a certain time.

> **WARNING:**
>
> Modifying these properties may negatively affect performance and UI responsiveness. Performance may degrade if there are too many smaller requests, or timeouts may occur with fewer, very large requests.
>
> You should thoroughly test the affect of your changes in a realistic environment before distributing the integrated workbook.

To configure multi-row processing:

1. Open the Business Object Editor for the target business object. then click the **Upload** tab.

   This image shows the Upload tab for an Employees business object with the **Supports Multi-Row Requests** check box as well as the **Initial Row Count** and **Target Response Time** fields.



You'll also find check boxes for parallel requests and upsert mode. For information about these features, see Enable Parallel Requests During Upload and Upload Changes Using Upsert Mode.

2. To enable multi-row processing, select **Supports Multi-Row Requests**.

   If this check box is not selected, **Initial Row Count** and **Target Response Time** are not available.

3. Provide valid values for these fields:

   - **Initial Row Count**: Specifies the number of rows that the add-in includes in the initial set of multi-row request(s). After the initial set of requests, the number of rows per request is determined dynamically. Enter a value of 1 or more.
   - **Target Response Time**: Specifies the duration of a single response in seconds that the add-in tries to achieve during upload. Enter a valid value between 1 to 180 seconds.

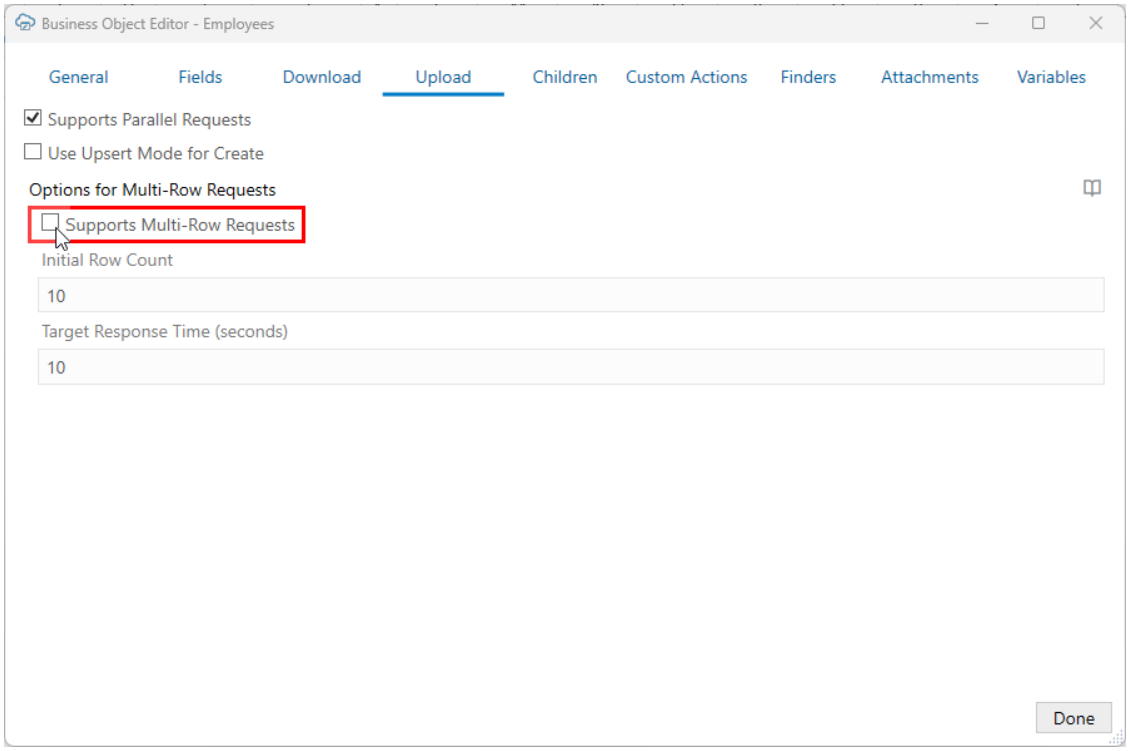
When enabled, the add-in sends multiple rows per request during an upload. For information about multi-row processing behavior, see About Multi-Row Processing.

Disable Multi-Row Requests for Upload

Some ADF REST services require a separate REST request for each changed row. For services with this requirement, you'll experience issues if the add-in uses multi-row requests. In this scenario, you can disable multi-row requests from the Business Object Editor.

If your business object doesn't require separate REST requests, it is strongly recommended that you use the default multi-row request behavior as described in Upload Changes Using Multi-Row Requests.

To disable multi-row requests for a business object, deselect the **Supports Multi-Row Requests** check box from the business object's Business Object Editor as shown here:



This check box is selected by default for ADF REST business objects. This check box is not present for other REST service types.

When you disable multi-row requests for a business object, the add-in uploads each changed row using a single request as it does for other service types. See Upload Table Changes Using Separate Requests for Each Row.

Depending on network characteristics and REST service overhead and performance, disabling multi-row requests may result in a negative effect on performance, such as adding significant wait time for the business user.

Workbook owners are responsible for any performance issues that arise. Make sure to test that your use cases will succeed in all customer environments and scenarios, if multi-row requests are disabled.

EffectiveOf Headers in Multi-Row Requests

If your ADF REST service includes date effective objects, you can add the `Effective-Of` header for each of the required REST operations. Oracle Visual Builder Add-in for Excel evaluates these headers when preparing each part of a multi-row request.

The add-in does not add the defined extra request headers to the header during a multi-row upload. Instead, the add-in checks for the `Effective-Of` header and, if defined, it adds it to the multi-row part using the `effectiveOf` member.

To support date effective objects when uploading using multi-row requests, define `Effective-Of` headers for the different REST operations configured for a business object. See REST Request Headers.

For more information on these objects, refer to Manage Date Effective Objects.

Upload Parent and Child Changes in the Same Payload

Some services require that parent and child row changes be uploaded using the same REST request payload. If your service requires this, you can configure your integrated workbook to upload all changes in a single, "nested" upload.

This feature is only available for Oracle ADF REST Resource services and should only be enabled if the service requires that parent and child changes be included in a single request. For example, your REST service may include logic that requires you to create at least one child (for example, a "Line" item) when you create a parent item (for example, a "Supplier Negotiations" item). In this case, the REST service enforces this by requiring the POST request that includes a new Supplier Negotiations item to also include at least one new Line item.

Likewise for updates. The target REST service may include back-end logic that requires operating on all changed child items at once and so all changes must be included in a single request.

If the service doesn't have this requirement, it is strongly recommended that you use the standard upload logic. See either Upload Changes from a Form-Over-Table Layout or Upload Changes from a Table Layout.

When enabled on a Form-over-Table layout, Oracle Visual Builder Add-in for Excel includes changes to parent (form fields) and child (table rows) in the payload of a single REST request.

You can also enable this for a set of dependent layouts. When you do this, all descendant rows ("children", "grandchildren", "great-grandchildren", and so on) of a parent are included in a single REST request. See Use Multiple Layouts for Multi-level Business Objects.

> **Note:**
>
> Read-only table layouts (and any descendant layouts) are skipped.

Before you continue, please review Notes and Limitations of Single Payload Uploads to determine if you should enable this feature.

To configure your workbook to use nested upload:

1. Open the worksheet with the top-level layout for the parent and child business objects that need to be uploaded together.

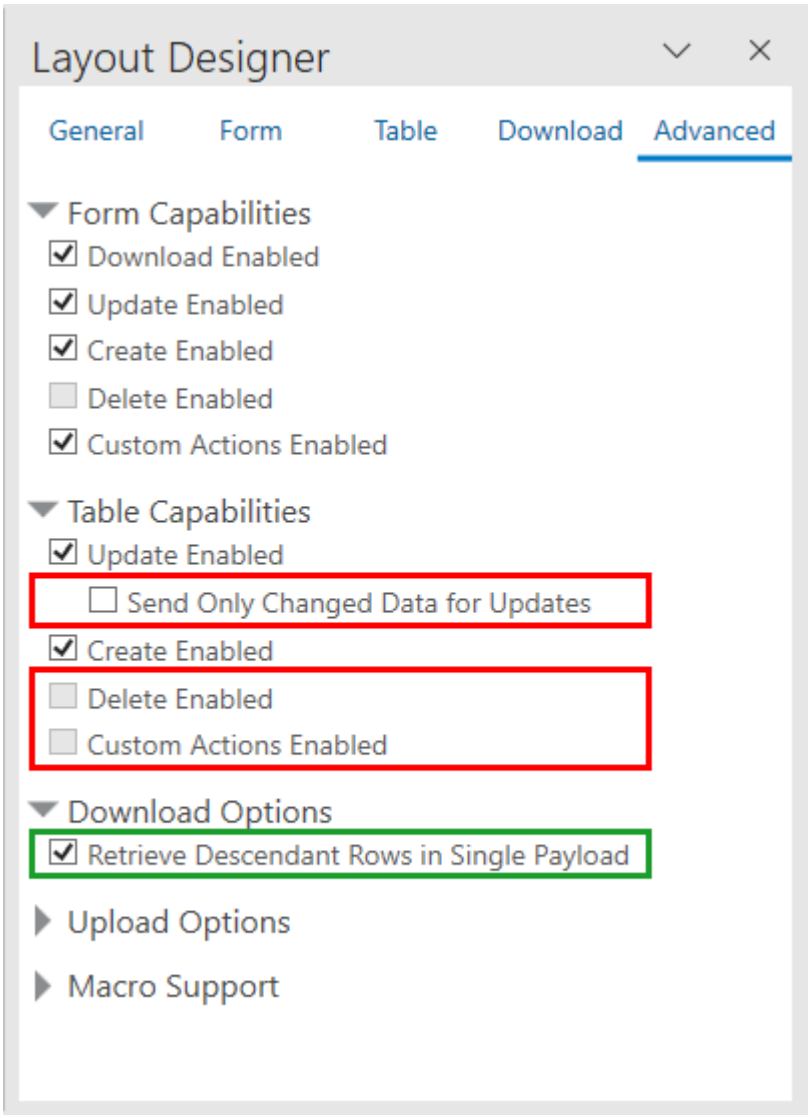
   This could be a standalone Form-over-Table or the primary (Form-over-Table or Table) layout of a set of dependent layouts.

2. In the Oracle Visual Builder tab, click **Designer** to open the Layout Designer in the Excel Task Pane.

3. From the Advanced tab of the Layout Designer, expand **Upload Options**, then select **Send Descendant Rows in Parent Payload**.

   When you select this check box, the add-in deselects these options for the table layout as well as any descendant table layouts:

   - **Send Only Changed Data for Updates**
   - **Delete Enabled**
   - **Custom Actions Enabled**



4. If you use this feature for a set of dependent layouts, add primary key columns in the child and descendant table layouts to support updates.

   As with other dependent layouts, add ancestor (direct parent) columns to support creates. See Create a Table Layout in an Excel Workbook and Add a Parent Column to Support Row Creation.

When a business user performs an upload, the add-in sends parent and child changes in a single REST request and then reports the success or failure of the upload operation. If there are read-only table layouts, these layouts and their descendants are skipped.

If the upload was a success, the add-in makes these updates for each changed row:

- The Change column in the table is cleared.
- The Status column displays an appropriate success status message, such as "Create Succeeded" or "Update Succeeded".
- The Status Viewer displays notification messages for the successful form and table operations.

> **Note:**
>
> Uploaded child and grandchild table rows are not refreshed automatically after the upload. The Status Viewer indicates this with a message: "This row may not contain the latest information from the service. Download to refresh the information for this row.". The business user will need to download data again to see the latest complete row data.

If the upload fails for any row at any level in a hierarchy, all associated rows in the hierarchy—starting at the top-level parent row and cascading down to its child rows, grandchild rows, and so on—are marked as failed.

In this scenario:

- The Change column in the table is left as is for all failed rows, either because the row itself failed or was part of a row hierarchy that included a failure.
- The Status column displays an appropriate failure status message for each failed row, such as "Created Failed" or "Update Failed".
- The Status Viewer displays the same notification messages for failed rows. The messages contain any parent failure messages along with the failure messages for descendant rows.

> **Note:**
>
> Each row from the primary table is treated separately, so a failure in one does not affect the success or failure of another. If there are other top-level parent rows with nested changes, these are re-sent following the failure and should succeed.

When an error occurs, the business user may have to visit each layout and scan the rows to find which row caused the failure.

Notes and Limitations of Single Payload Uploads

Single payload uploads (also referred to as "nested" uploads) are only supported for Oracle ADF REST Resource services. They can be enabled on standalone Form-over-Table layouts or on the primary layout (Form-over-Table or Table layout) for a set of dependent layouts. Review this section to see if the feature is suitable for your integrated workbook.

> **Note:**
>
> If your service doesn't require that parent and child changes are included in a single request, it is strongly recommended that you use the standard upload logic described in Upload Changes from a Table Layout or Upload Changes from a Form-Over-Table Layout.

Requirements

- The service must support request payload structures that include child and descendant rows included with the parent fields. Rows for child business objects are included as array-typed members as peers of the parent's fields. These child rows in turn include the child fields along with arrays of grandchild rows, and so on.

- The business object must support the use of Upsert mode when uploading both new and changed rows in the same upload request.
  This requirement refers to the service itself. Oracle Visual Builder Add-in for Excel does not have to be configured to use Upsert mode for the business object to support this feature.

Table Capabilities

Some Table capabilities are incompatible with the nested upload feature. When this feature is enabled, these capabilities are disabled for the table in a Form-over-Table layout, the Table layout used as the primary layout in a set of dependent layouts, and any descendant table layouts:

- **Send Only Changed Data for Updates**
- **Delete Enabled**
- **Custom Actions Enabled**

Tables for attachment business objects are not supported as part of a nested upload.

Form-over-Table Layouts

Form rows marked for deletion or custom action are not affected by this feature. When you delete the form row, the `DELETE` request is sent immediately with no payload. When you invoke a custom action on the form row, the corresponding `POST` request is sent immediately with only the expected custom action payload. In both cases, there is no nested content from the child table or child layouts.

Scalability

Be aware that this feature will not scale to handle large, enterprise-level volumes. If you enable this feature for your workbook, you are responsible for any scalability issues that arise. Make sure to test your workbook to verify that all required use cases will succeed in all customer environments and scenarios.

Potential scalability issues include:

- Complex business logic or the processing of large numbers of nested items could push the request duration beyond what some load balancers and other network components, such as Akamai, support. This could result in the request failing due to a timeout.
- Large request payload sizes will likely consume large amounts of memory and CPU. This could potentially affect other clients working against the same endpoint, virtual machine, or container.
  For example, consider a set of dependent layouts with a Table as a primary layout that has a large number of changes. If all parent rows and associated descendant rows are sent in a single request, the payload size and processing time for the request may be significantly larger than if you sent each parent row and associated descendant rows separately.
- Large request payload sizes could exceed the maximum supported by the endpoint.

Data Consistency and Conflict Detection

When this feature is enabled, requests with parent and child changes in the payload may result in changes previously uploaded by another user being overwritten without notification. This can occur when multiple users are acting on the same business object rows using different workbooks.

Consider a scenario where two users, user A and user B, both download data at the same time. If user A uploads their changes to child or descendant rows and then later user B uploads their changes to the same rows, user B's changes will overwrite user A's changes.

Also note that the "Upsert" scenario, where an existing parent item is uploaded with new child rows, does not currently support conflict detection.

Child and Descendant Primary Keys Included in Tables

Additional configuration may be needed by the workbook developer to support cases where descendant rows are updated. To ensure that updates succeed, you must include primary/alternate key fields as columns in child and other descendant Table layouts.

The child items' key values must appear in the payload, in the array of child items. The service must be able to locate the existing child item in order to update it.

REST Request Headers

Sending configured REST request headers is only supported for the primary table rows during nested upload. Rows from child and descendant tables will not include such headers, even if configured. See REST Request Headers and EffectiveOf Headers in Multi-Row Requests.

Upload Changes Using Upsert Mode

ADF REST services support an optional "upsert" mode that you can use when uploading changes to your REST service.

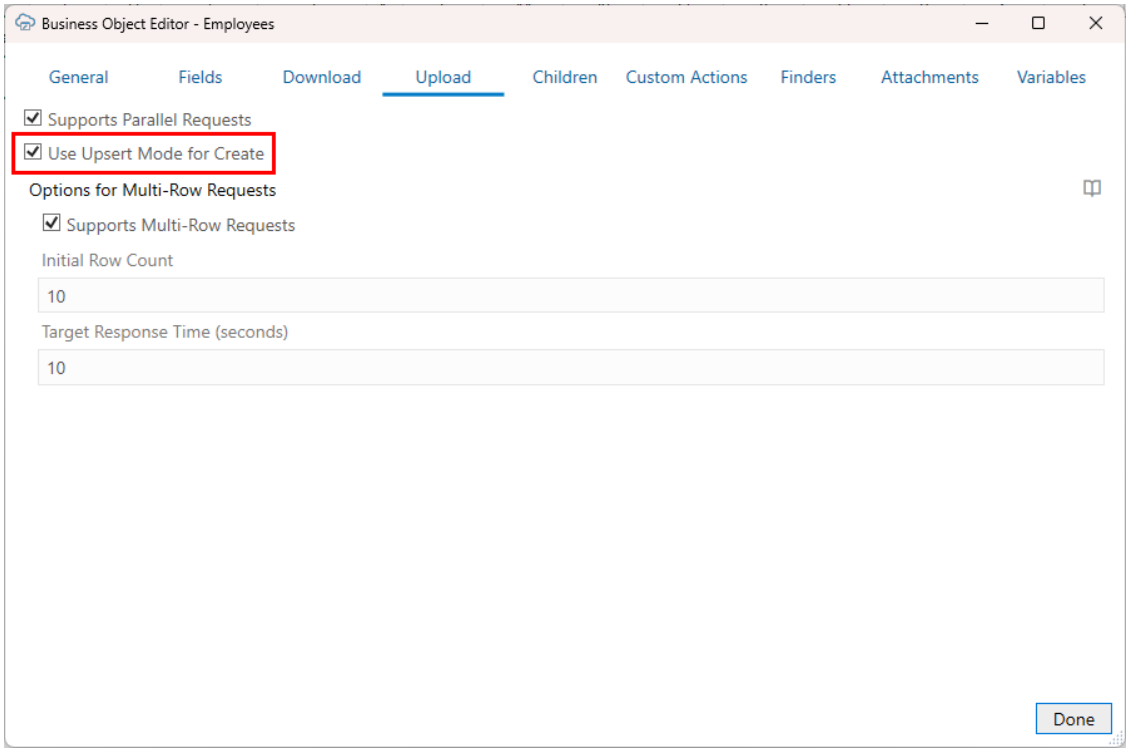
Ensure that your REST services business object supports upsert mode before you enable it.

When upsert mode is enabled, the REST service should behave as follows: first, it examines the incoming payload of a create row request. Next, if the payload contains enough information to match an existing row, the operation updates that existing item. If there is no match, the operation should create a new row. See Updating or Creating Resource Items (Upsert) for more information.

The exact behavior of upsert mode is determined by the service.

You enable upsert mode by selecting the **Use Upsert Mode for Create** from the Upload tab of the Business Object Editor.

> **Note:**
>
> This check box is only available for ADF REST services and is hidden for all other service types.



When upsert mode is enabled, the add-in sends the HTTP header, `Upsert-Mode: true`, with POST requests for single-row creates. For multi-row requests, the add-in sets the operation value for each batch part to "upsert" instead of "create".

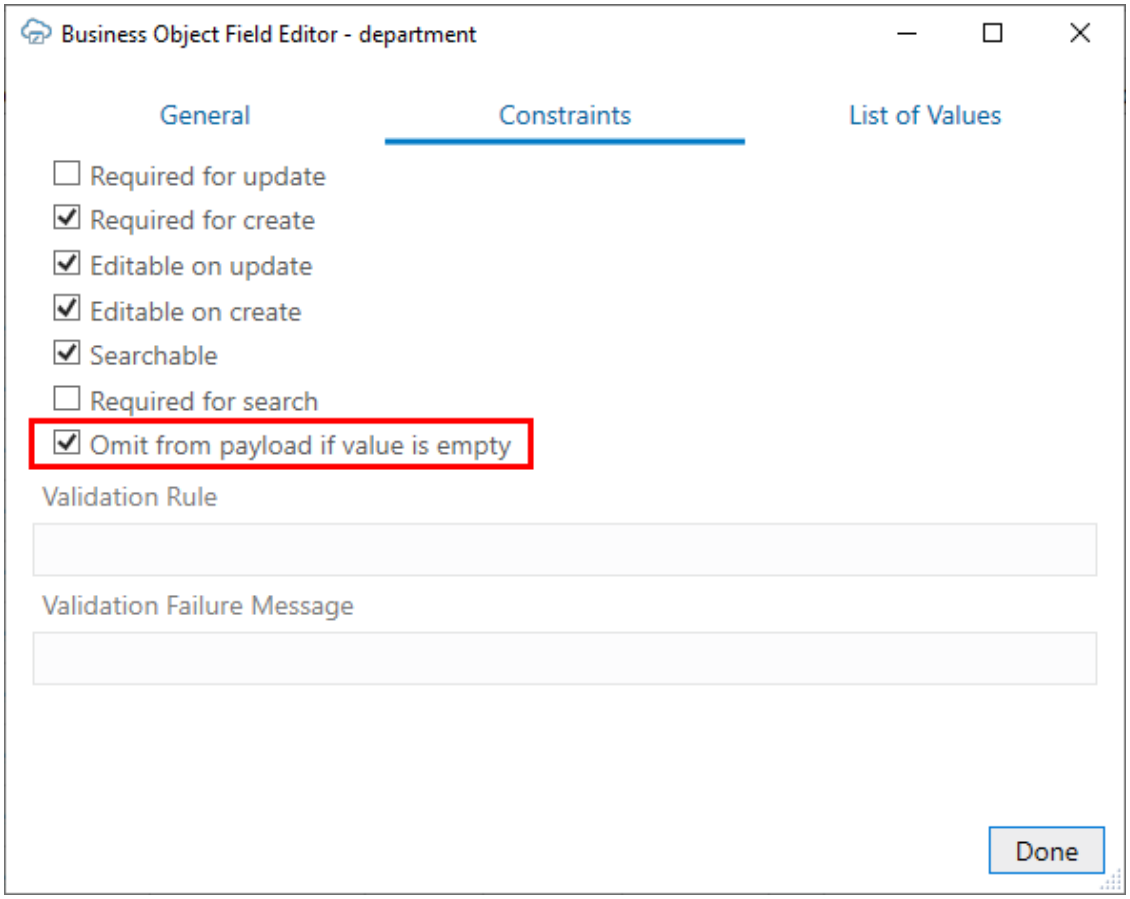
Upsert mode does not affect how the add-in treats existing rows that have pending (update) changes during upload.

Omit Empty Values During Upload

You can configure Oracle Visual Builder Add-in for Excel to omit fields with empty values during upload.

Request payloads for Create and Update operations normally include a value for every column in the table (except those for read-only fields and custom action payload fields) even if the cell value is empty. For empty cell values, a null value is included for that field in the payload. Null values can potentially cause validation errors.

To avoid these errors, you can configure the add-in to omit fields with null values from the payload by selecting the **Omit from payload if value is empty** check box in the Business Object Field Editor.



When selected, the add-in won't include the field in the payload during upload if the value in the field is empty.

The add-in sets the default value for the **Omit from payload if value is empty** check box based on the "nullable" property in the OpenAPI document for this field. If "nullable" is true, the check box is deselected and cells with empty values are uploaded; if "nullable" is false (or the property is missing), then the check box is selected and cells with empty values are omitted from the upload.

> **Note:**
>
> If this check box is selected (empty cell values are skipped), you won't be able to change a "non-empty" value to an empty value for a field in an existing row.

Send Only Changed Data During Upload

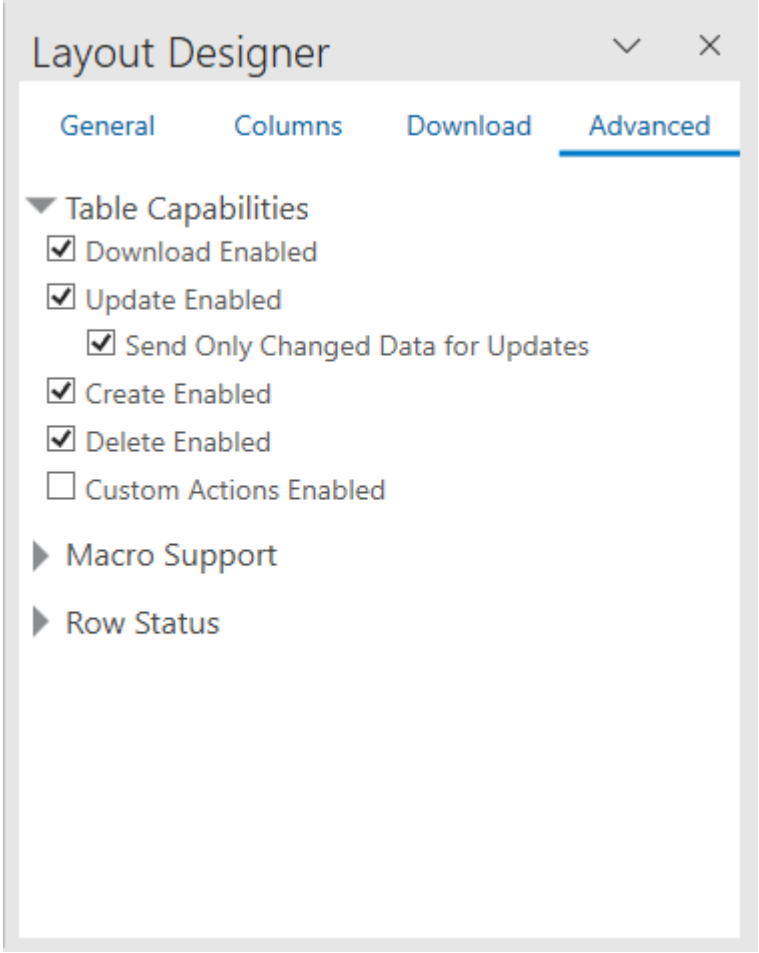
You can configure Oracle Visual Builder Add-in for Excel to send only changed data from a Table layout or the table part of a Form-over-Table layout during an upload. The add-in always sends only changed values from the form area of a Form-over-Table layout.

Request payloads for Update operations normally include a value for every editable cell in a row marked for update even if the value hasn't changed since the last download or upload operation. For some REST services, sending data that has not been modified in a PATCH, PUT, or multi-row update request can cause issues.

To avoid these issues, you can configure the add-in to send only changed values in the payload by selecting the **Send Only Changed Data for Updates** check box for your table. When this is enabled, cells that are unchanged are not included in the payload.

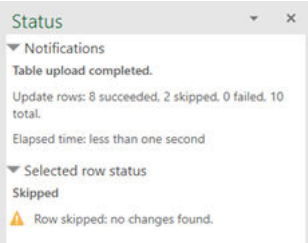
> **Note:**
>
> This feature is not available for NetSuite.

To enable this feature, select the layout, then open the **Advanced** tab of the Layout Designer. The **Send Only Changed Data for Updates** check box can be found under **Table Capabilities**.



When the check box is enabled, the add-in caches a copy of all of the data in the table during download. Then, when the business user triggers an upload, the add-in compares values in editable cells in rows marked for update to the cached values, and only includes fields that are different in the request payload.

If a row is marked for "Update" in the Change column, but no values have been changed, the row is omitted and the Status column displays "Skipped" after the upload. The Status viewer also displays the results of the upload.



On a successful Create or Upload of a row, the cache is updated with all the current values to be compared during the next upload, if the service provides the row in the response.

> **Note:**
>
> If a field in a table is required for update (the **Required for Update** check box is selected in the Business Object Field Editor), the value is included in the payload even if it is unchanged.

Performance

Please note that this feature may affect performance during download and upload, since the add-in must capture and store a duplicate set of downloaded table data. You can expect workbooks using this feature to increase in size. Also, you may see a slow down in downloads and uploads since the add-in must read from, and write to, the cache.

If you choose to use this feature, it is recommended that you run relative performance tests in realistic environments and assess the performance impact before delivering workbooks with this feature enabled.

REST Request Payload

Let's suppose a business user changes the salary value for an employee named Steven King. When this check box is not selected, the add-in includes both changed and unchanged data in the payload, like this:

```
{
  "FirstName": "Steven",
  "LastName": "King",
  "Email": "SKING",
  "HireDate": "2003-06-17T04:00:00Z",
  "JobId": "AD_PRES",
  "Salary": 120000,
  "CommissionPct": null,
  "ManagerId": null,
  "DepartmentId": 90
}
```

When the check box is selected, only the salary data is included:

```
{
"Salary": 120000
}
```

Data Consistency

When a workbook uses a compatible REST API service that supports data consistency verification using an entity tag (Etag) mechanism, the add-in detects and reacts to the following scenario:

1. Person A downloads information from a business object into a table in their integrated workbook.
2. Person B downloads the same information into a table in their integrated workbook, edits it, and uploads changes.
3. Person A then edits the same information (downloaded in Step 1) and uploads the changes.
4. The add-in provides the service with the necessary information (entity tags) to prevent Person A's changes from overwriting those changes made by Person B.

   The add-in sends an `If-Match` request header containing the entity tag for single row requests or includes the entity tag along with the row changes as part of a multi-row request. See Upload Changes Using Multi-Row Requests.

   When the server inspects the entity tag and detects such a change, its response (either an `HTTP Status 412`, or an error code of `11412`) allows the add-in to display an error message for any such rows in the table like this:

   ```
   This row has been modified by another user. Please download before editing.
   ```

   If you see this message, you'll have to discard your changes by downloading the latest data and then redoing your changes as needed.

> **Note:**
>
> Some services do not support this conflict detection functionality. If the service does not, then Person A's changes will replace Person B's changes with no warning. Contact the service owner for more information.

For information about the entity tag (ETag) mechanism, see Data Consistency Tasks in *Accessing Business Objects Using REST APIs*.

If your workbook uses other types of REST services, the last writer wins. So, for the scenario just outlined, Person A's changes in Step 3 will overwrite the changes of Person B in Step 2.

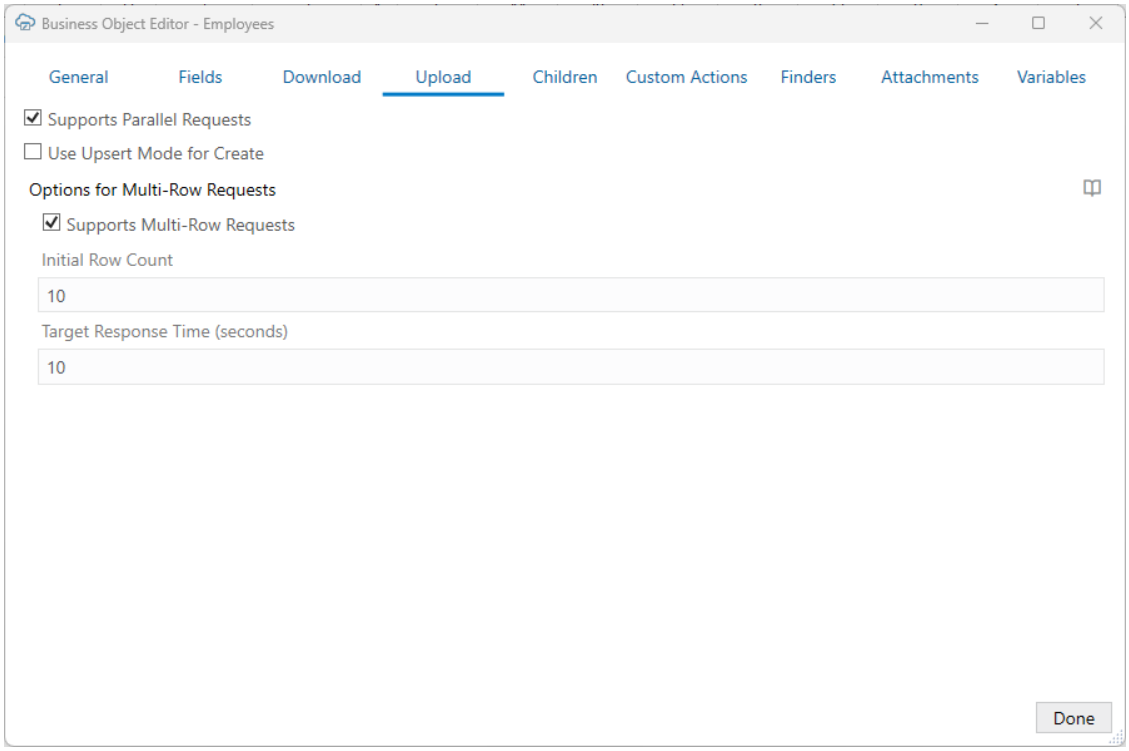
Enable Parallel Requests During Upload

You can enable Oracle Visual Builder Add-in for Excel to send multiple requests in parallel—up to a maximum of four—during upload. This can improve performance and shorten the overall time to upload a large set of changes.

Parallel requests behave differently than multi-row requests. With a multi-row request, the add-in sends the changes for multiple rows in each request; With parallel requests, the add-in sends multiple requests at the same time, regardless of whether the request contains one row or multiple rows. See Upload Changes Using Multi-Row Requests.

> **Note:**
>
> Although the add-in processes changed rows in "table order" (from top to bottom), the threading used for parallel requests may result in rows arriving at the REST service endpoint "out of order". For example, request 7 may arrive at the endpoint before request 6.

To enable parallel requests, open the Business Object Editor for the target business object, then click the **Upload** tab and select **Supports Parallel Requests**.



14

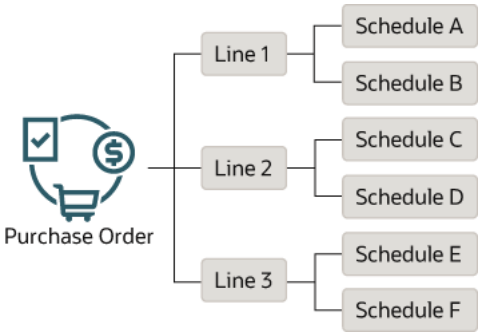
Use Multiple Layouts for Multi-level Business Objects

When business objects have a parent-child relationship, you can create a set of dependent layouts and then perform operations, such as downloading data and uploading your changes, on all your layouts with a single gesture.

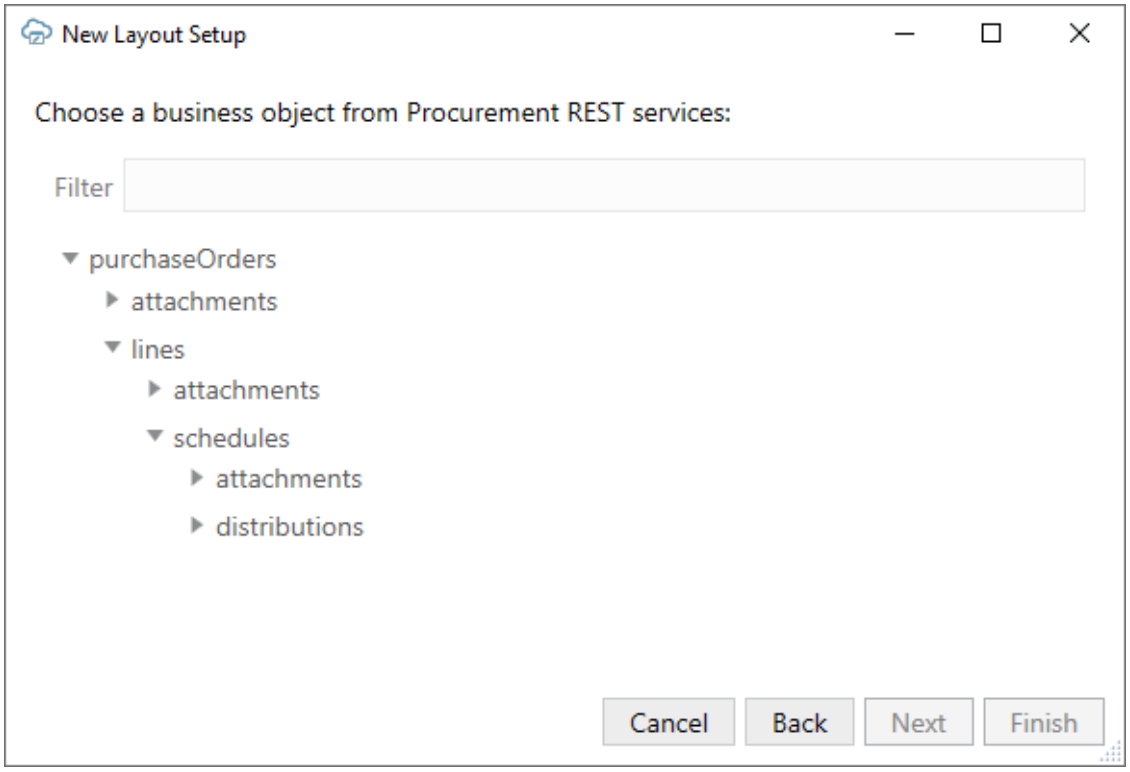
Business Object Hierarchies

Consider an example hierarchy of business objects where **purchaseOrders** is the parent, **lines** is the child, and **schedules** is the grandchild.

In this hierarchy, **purchaseOrders** is a collection of top-level purchase orders each with one or more lines for managing the details of each order. Each of these lines may include one or more schedules for tracking shipping details.



A hierarchy of business objects can go to even more levels ("great-grandchildren" and "great-great-grandchildren") or have more than one business object at each level ("siblings"). For example, the following hierarchy has "sibling" grandchildren (**attachments** and **schedules**) and a "great-grandchild" (**distributions**) under **schedules**.

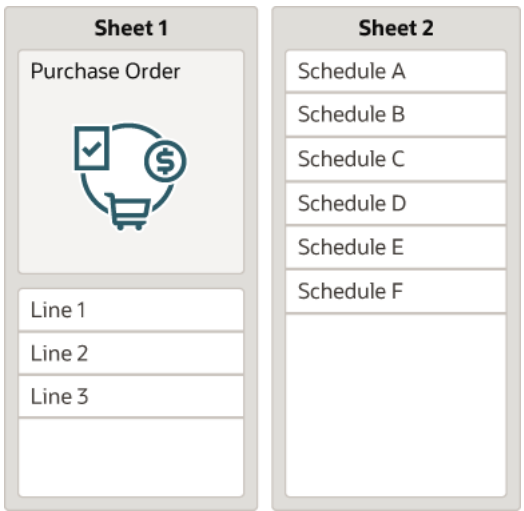


Configuring Dependent Layouts

When you create a set of dependent layouts, you create either a Form-over-Table or a Table layout as your top-level or "primary" layout on the first worksheet, as well as Table layouts for each subordinate level on separate worksheets. See Create a Set of Dependent Layouts.

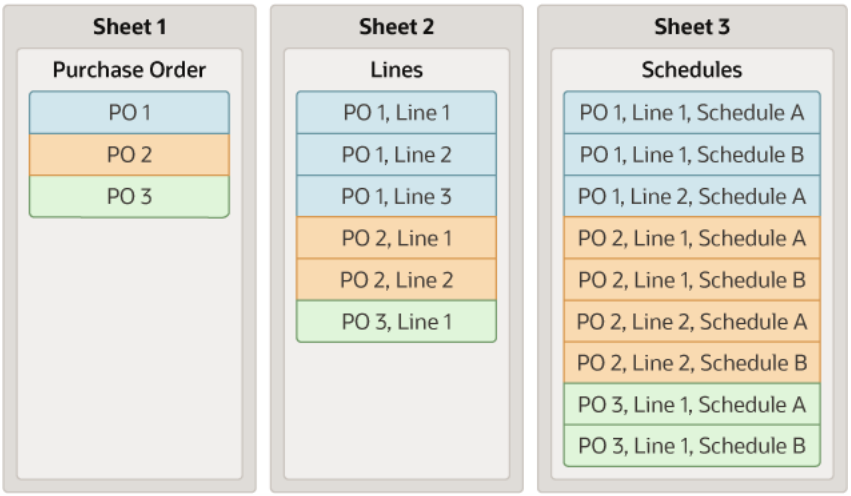
If you want to show a single purchase order and all its associated lines and schedules, create a Form-over-Table layout for the first two levels (**purchaseOrders** and **lines**) and a Table layout for the third level (**schedules**). You can then configure a search to prompt the user to enter a purchase order ID.

When the layouts are populated, the form part of the Form-over-Table layout displays details for the purchase order, the table part shows all the lines for the purchase order, and the dependent Table layout shows all the schedules for the associated lines.



You can also use a Table layout as your primary layout instead of a Form-over-Table layout. In this case, create Table layouts for each level in the hierarchy (**purchaseOrders**, **lines**, and **schedules**) on separate worksheets. As before, link each dependent layout back to its parent.

In this configuration, you can display one or more purchase orders in the primary Table layout, and all associated lines and schedules in the subsequent Table layouts.



Note:

Keep in mind that a single parent may have multiple children with each of these having multiple grandchildren. Consequently, a large number of results in the primary layout may result in very large volumes of data in the subordinate layouts. To avoid performance issues when downloading and uploading data, Oracle recommends that you configure an appropriate query to limit the number of results in the primary layout. See Configure Search Options for Download.

"Siblings" are business objects at the same level and can be at the second level or deeper in your hierarchy. For example, you may have a parent business object (**Expenses**) at the top level and two children (**Itemizations** and **Distributions**) at the second level. In this example, **Itemizations** and **Distributions** are siblings of each other.

To create a set of dependent layouts for this hierarchy, you have two options. You can create a Form-over-Table layout for the parent and one of the children (for example, **Expenses** and **Itemizations**) on one worksheet and a Table layout for the other child (**Distributions**) on a second worksheet. Or you can create Table layouts for each business object (**Expenses**, **Itemizations**, and **Distributions**) on separate worksheets. Again, use a Form-over-Table layout if you want to show a single expense record in the form and list all associated itemizations and distributions in the tables.

Once the dependencies are established, download, upload, and clear operations act on all the linked layouts, starting from the primary layout, followed by the child layouts, the grandchildren layouts, and so on.

Create a Set of Dependent Layouts

Create a set of dependent layouts for a hierarchy of business objects and link the layouts together.

> **Note:**
>
> Before you create a set of dependent layouts, ensure your REST service meets the requirements as set out in Requirements for Dependent Layouts.

To create a set of dependent layouts, run the New Layout Setup wizard on the first worksheet and select the parent and child business objects you want to include. If you choose to create a Form-over-Table layout as the primary layout, you'll need to indicate which child business object you want to use for table part of the layout. When you finish, Oracle Visual Builder Add-in for Excel creates a new worksheet with a Table layout for each additional descendant business object in your hierarchy.

When you have created your layouts, you can configure appropriate queries to limit the number of rows to display in your layouts.

If you create a set of layouts but later decide you want to add another descendant business object, simply create a Table layout for the descendant business object and then link the new dependent layout to its immediate parent layout. See Add a Layout to a Set of Dependent Layouts.

When you create a dependent layout, you can choose to include columns from the parent or higher layout to help your business users track which child rows are associated with which higher-level rows. See Add a Parent Column to Support Row Creation.

To enable the creation of items in a Table layout, you may need to add one or more columns to the layout from the layout's immediate parent. Including parent columns in the layout allows your business users to specify a unique parent for any new rows they create.

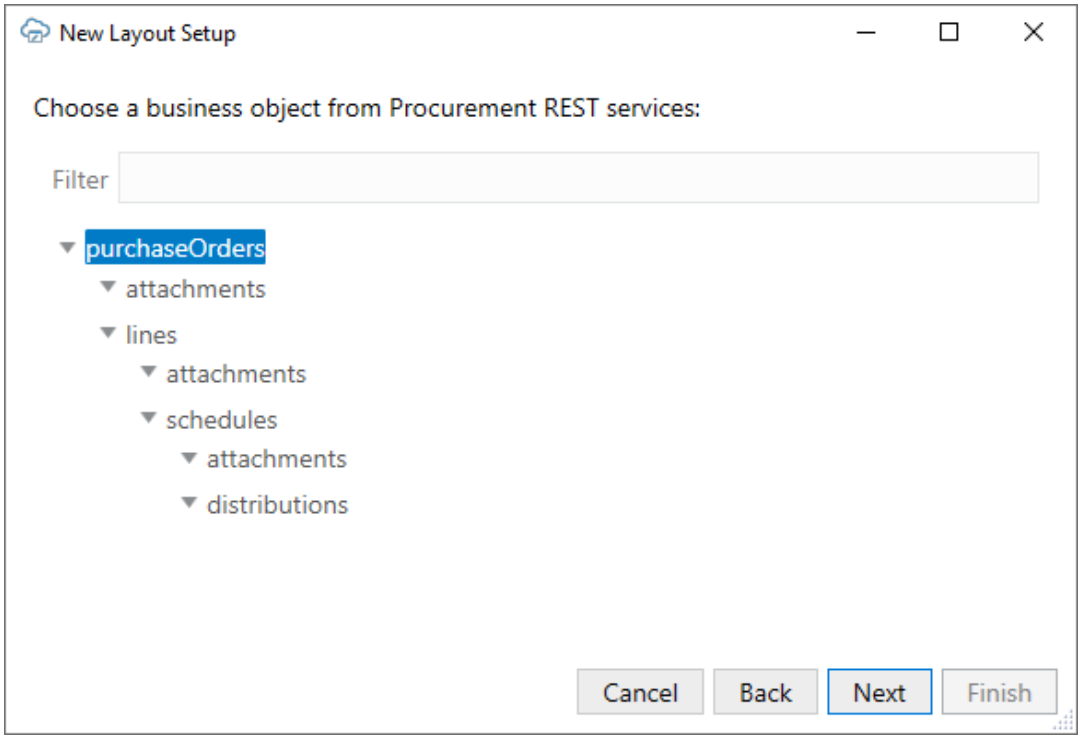
Let's use the example in this section to create a hierarchy of dependent layouts that mirrors your business object hierarchy.

The primary layout in a set of dependent layouts can be a Form-over-Table layout or a Table layout. This topic covers how to create a set of dependent layouts that uses a Form-over-Table as the primary layout, but the principle is the same for using a Table layout as the primary layout.

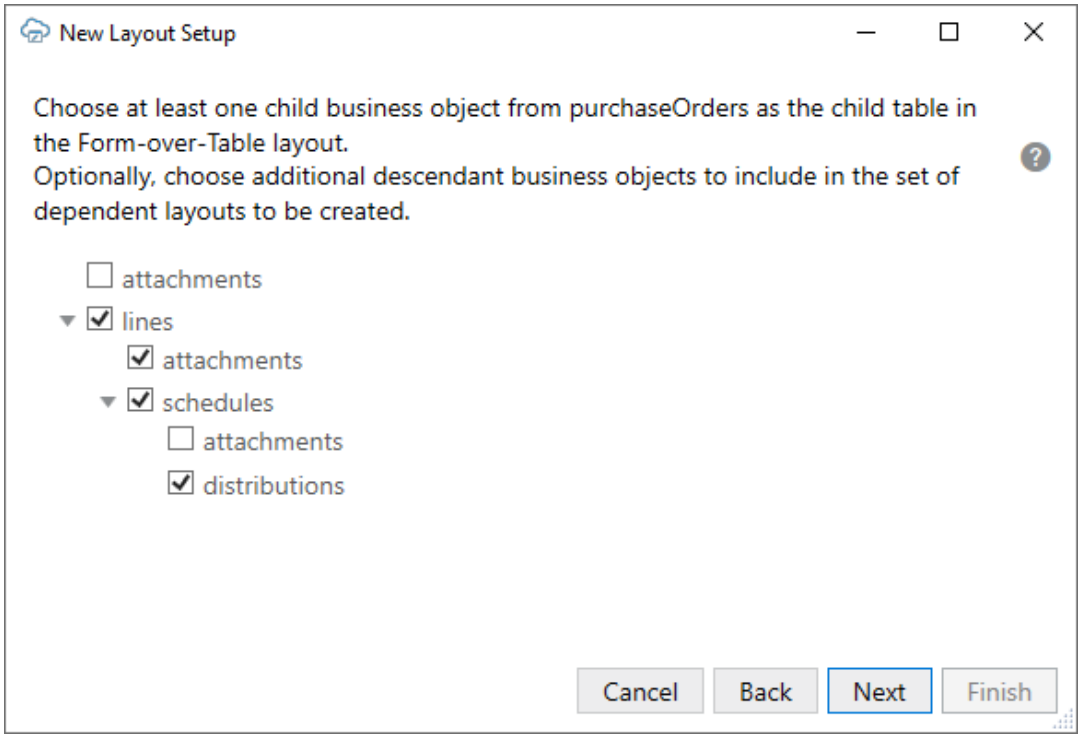
To create a set of dependent layouts:

1. Select the cell in a blank worksheet where you want to locate the primary layout.
2. In the Oracle Visual Builder tab, click **Designer** to launch the New Layout Setup wizard.
3. When prompted, provide the service metadata document.

   If the service includes five or more business objects, select the business objects you want to include in the catalog, then click **Next**.
4. Choose a top-level business object, then click **Next**. In our example, select **purchaseOrders** for a Form-over-Table layout showing purchase orders over lines (the first two levels in our hierarchy).



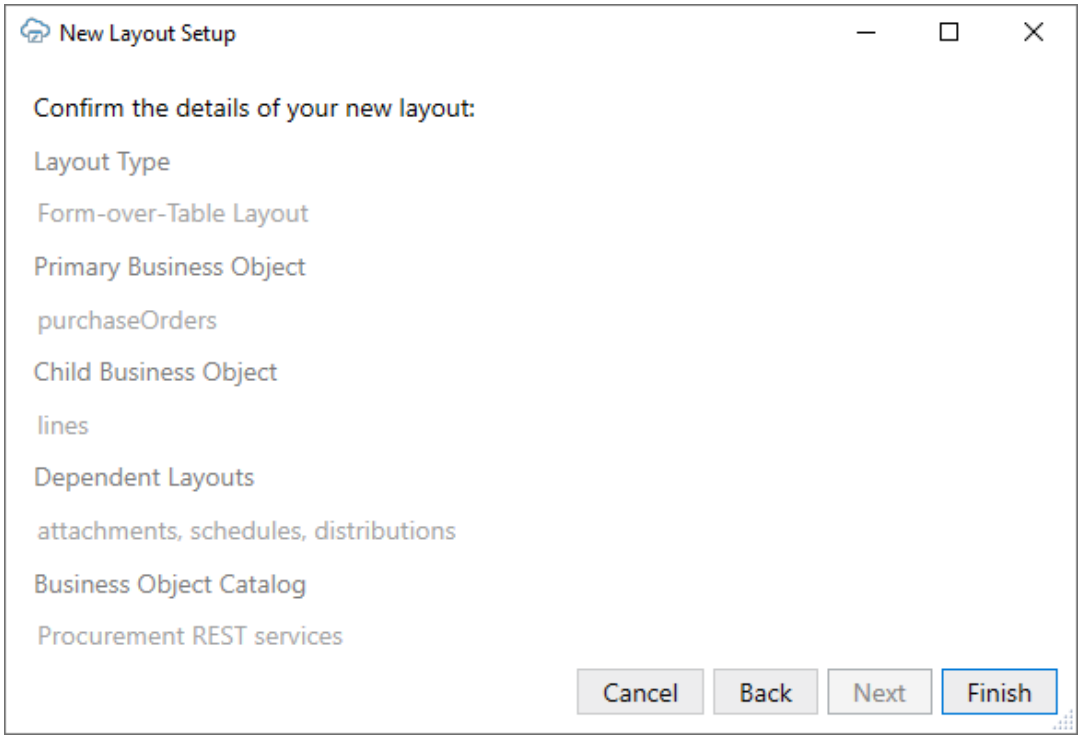
5. Select the desired layout type for the primary layout, then click **Next**.
6. Choose the child business objects you want to include in the set of dependent layouts, then click **Next**. In our example, select **lines**, **attachments**, **schedules**, and **distributions**.



If you are creating a Form-over-Table layout as your primary layout and you selected more than one direct child of the primary business object, you will be prompted to select which

business object you want to show in the table of the primary layout. Select a business object, then click **Next**.

The add-in then displays a review of your choices. In this case, the add-in will create a Form-over-Table layout with purchaseOrders in the form and lines in the table. It will also create three linked Table layouts for attachments, schedules, and distributions.



7. Review the layout details and then click **Finish**.

   The add-in creates a Form-over-Table layout for purchaseOrders and lines on the first worksheet as well as Table layouts for attachments, schedules, and distributions on separate worksheets in the workbook.

After you have created the set of dependent layouts, you can continue to configure your layouts as required.

Here are some of the changes you might want to consider making:

- If you want to allow your business users to create new rows in a dependent layout, you must add a parent column to the layout. See Add a Parent Column to Support Row Creation.
- Even if Create is disabled on a table layout, you might still want to add an ancestor column to a layout to help your business users track which child rows are associated with which higher-level rows. To add ancestor columns to a layout, see Manage Fields in a Form or Table.
- You may want to configure download parameters on each dependent layout for downloading data. See Filter Data for a Set of Dependent Layouts.

> **Note:**
>
> Before you publish and distribute your workbook to users, test the workbook to ensure that download, upload, or clear operations work on all layouts in the hierarchy as expected. See Manage Data in a Dependent Layout in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Add a Layout to a Set of Dependent Layouts

You can add a Table layout based on a descendant business object to an existing set of dependent layouts by linking it to the parent business object's layout.

Let's suppose you have created a set of dependent layouts for a hierarchy of business objects that includes **purchaseOrders** (parent), **lines** (child), and **schedules** (grandchild). Now, you want to add another business object, **distributions**, which is the child business object of **schedules** and the great-grandchild of **purchaseOrders**.

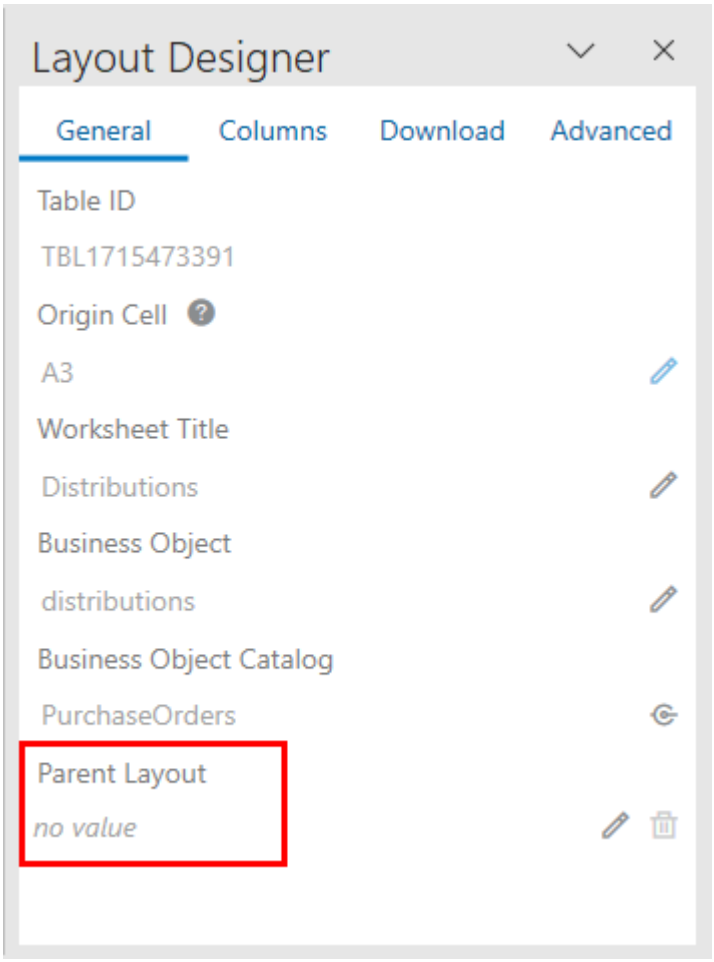
To do this, start by creating a new worksheet and then adding a Table layout for distributions. When you create your Table layout, ensure that you select the same business object catalog that is used by your primary layout. When you've finished, use the Layout Designer to choose a parent layout for your new Table layout.

To add a new layout to a set of dependent layouts:

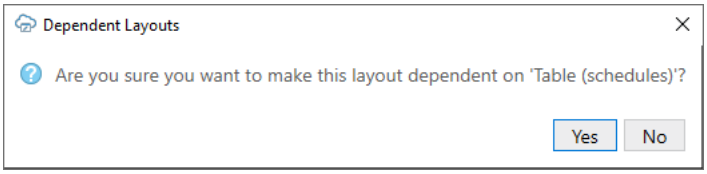
1. Click the New Sheet icon to add a new worksheet.
2. Click **Designer** to launch the New Layout Setup wizard.
3. From the first screen of the wizard, choose the business object catalog and click **Next**.
4. Choose the business object that's next in the hierarchy—for example, **distributions**—and click **Next**.

   The next screen of the wizard displays the layout type for the new layout. Because distributions is not a top-level business object in the hierarchy, only the **Table Layout** option is available.
5. Click **Next**.
6. Review the layout details and then click **Finish**.

   A new Table layout is created for the selected descendant business object in your hierarchy. Notice the **Parent Layout** field in the Layout Designer's General tab, shown here for a distributions layout. This field is only displayed in layouts where the business object is a child of another business object in the same business catalog.



7. To set the parent layout for the new Table layout, click the Choose Parent Layout icon (✎) in the Layout Designer's General tab.
8. Select the appropriate layout from the Dependent Layouts window and click **OK**. If there is only one possible parent layout, a prompt appears asking you to confirm a parent for the layout. Click **Yes** to confirm the parent layout in the hierarchy, for example:



> **Note:**
>
> If you see a message, "No layouts found that this layout can depend on", it may mean the layout is not part of the same catalog or that the sibling business object you are trying to link has already been used in the table part of a Form-over-Table layout.

Add Ancestor Columns to Dependent Layouts

When you create a dependent layout, you can choose to include columns from parent or higher layout. Columns from these higher-level layouts are referred to as "ancestor" columns.

You might choose to add at least one column from the layout's immediate parent if you want to allow your business users to create rows in the dependent layout. See Add a Parent Column to Support Row Creation.

You might also just add an ancestor column to a dependent layout to help your business users track which child rows are associated with which higher-level rows. See Add Ancestor Columns to Provide Additional Context.

Add a Parent Column to Support Row Creation

When a business user adds a row to a dependent layout, they need to indicate the parent row that the child row should be associated with.
Which layouts need parent columns depends on the type of layout used for your primary layout. Let's start with the Form-over-Table case. Let's suppose you have a business object hierarchy where **purchaseOrders** is the parent, **lines** is the child, and **schedules** is the grandchild. For this hierarchy, you create a Form-over-Table with **purchaseOrders** in the form and **lines** in the table, as well as a separate Table layout for **schedules**.

When you download data for this set of dependent layouts, you download a single purchase order for the form, all associated lines for the Form-over-Table's table, and all schedules associated with these lines in the Table layout.

Now let's suppose you want to create a new line for the purchase order. Because the Form-over-Table layout shows only one purchase order in the form *and* only associated lines in the Table, any line you create is automatically associated with the selected purchase order. You don't have to enter a purchase order number to associate it with the selected one.

But what if you want to create a schedule and associate it with one of the lines? The Schedules layout may include schedules from different lines in the purchase order. To make sure the schedule you create is associated with the right line, you'll need to specify an existing line for the new schedule. Therefore, you need to have at least one parent column showing in the Schedules layout that uniquely identifies the line (for example, `LineNumber`).

Likewise for any descendant level below the second level. Suppose our sample hierarchy includes a business object, **distributions** that is the child of **schedules**. A Distributions layout will need to include at least one parent column (for example, `ScheduleNumber`) so that you can specify which schedule a new distribution should be associated with.

> **Note:**
>
> If you have a sibling business object at the second level in this scenario, you don't need to add fields from the parent level. Suppose our sample hierarchy includes an **attachments** business object that is the child of purchaseOrders and the sibling of lines. If you create a Table layout for attachments that you link to the primary layout, then the add-in only downloads attachment rows that are associated with the selected purchase order. If you create a new attachment, it is automatically associated with this purchase order.

Now let's look at the second case: when the primary layout is a Table layout. In this scenario, the primary Table layout may display multiple purchase orders. The second-level Table layouts (attachments and lines) may then include rows for each of the purchase orders downloaded in the primary layout. Similarly, the third and fourth-level layouts (Schedules and Distributions) will include schedules and distributions from different purchase order and lines.

If you want to create a new line for a purchase order, you'll need to specify an existing purchase order. Therefore, you'll need to have at least one parent column showing in the Lines layout that uniquely identifies the purchase order (for example, the `PO Header Id` field).

Add Ancestor Fields to a Layout

To ensure a business user can add a row to a dependent layout, include columns from the parent layout to the dependent layouts in the set.

If you can't find a set of fields from the parent business object that can uniquely identify a new row, you can add ancestor columns that are already included in the parent layout to this child layout for parent row matching. The add-in can match the new row to a parent row by considering values from all ancestor columns in this child layout.

Take our example of set of dependent layouts: **purchaseOrders** (parent), **lines** (child), **schedules** (grandchild), and **distributions** (great grandchild). Suppose you want to create a new distribution row but the fields from the schedules business object cannot uniquely specify a single schedule. In this case, the add-in may be able to match a distribution row to a schedule using a combination of fields from other levels: "Buyer" from the purchase order business object, "LineNumber" from lines, and "Carrier" from schedules.

To support row creation, you would then add these columns to the distribution layout, as long as "Buyer" and "LineNumber" are added as ancestor columns and "Carrier" is a column in the schedules layout.

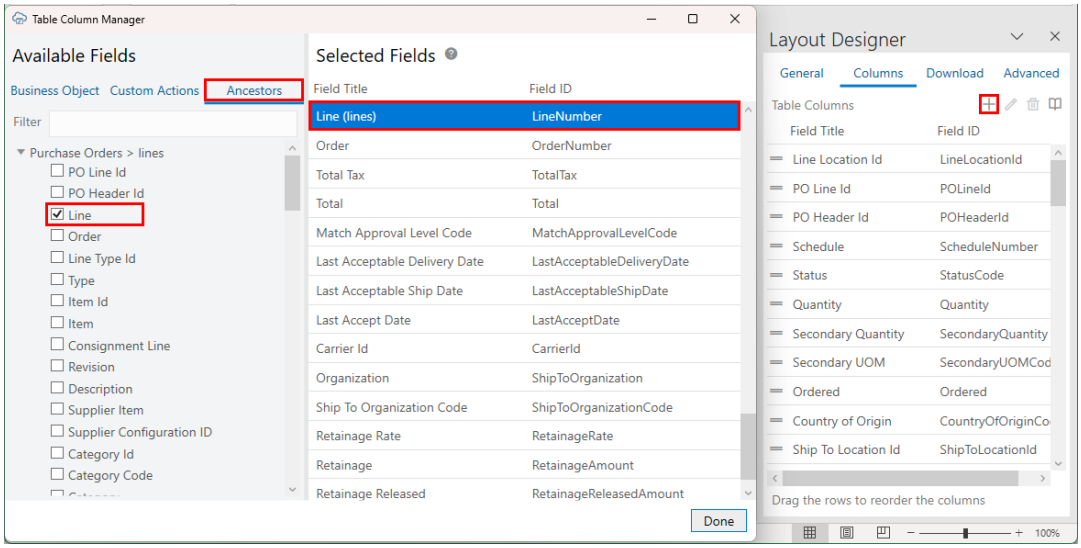
Here are a few things to keep in mind when adding ancestor columns:

- The parent column you choose must uniquely identify the parent record and must be included in the parent layout.
- If no parent columns are configured, the table cannot support row creation and rows inserted into the table are ignored during upload.
- If a desired parent field is already displayed as a column in the child layout—for example, a foreign key column—remove it and add that field from its original layout as an ancestor column in this child layout.
  This doesn't apply to a field from grandparent or higher layouts that is required for create or update.
- If parent columns are not sufficient to uniquely specify a parent record, you can add non-direct ancestor columns that are already included in the parent layout to support row creation. In this case, all ancestor columns you choose must uniquely identify the parent record.

To add an ancestor column to your layout:

1. Open the Table Column Manager from the Layout Designer.
2. Click the **Ancestors** tab from the **Available Fields** pane.

   Available ancestor columns are grouped by business object. In this image, fields for the lines business object are shown under **lines**.



3. Expand the list if necessary, then select the parent field you want to add to your layout.

> **Note:**
>
> To add a parent field before an existing column in the table, select the existing column in **Selected Fields** list, then select the parent field check box. For example, to display an parent field first in the table layout right after the Status column, select the first column.

4. Click **Done** to close the Table Column Manager.

Oracle Visual Builder Add-in for Excel redraws the table in the layout to include the parent column. The table header for the parent column uses the format "*<field title>* (*<business object title>*)" such as "Line Number (lines)".

Refer to Manage Fields in a Form or Table for more information.

Add Ancestor Columns to Provide Additional Context

You can add an ancestor column to a dependent layout to provide information about the ancestor.

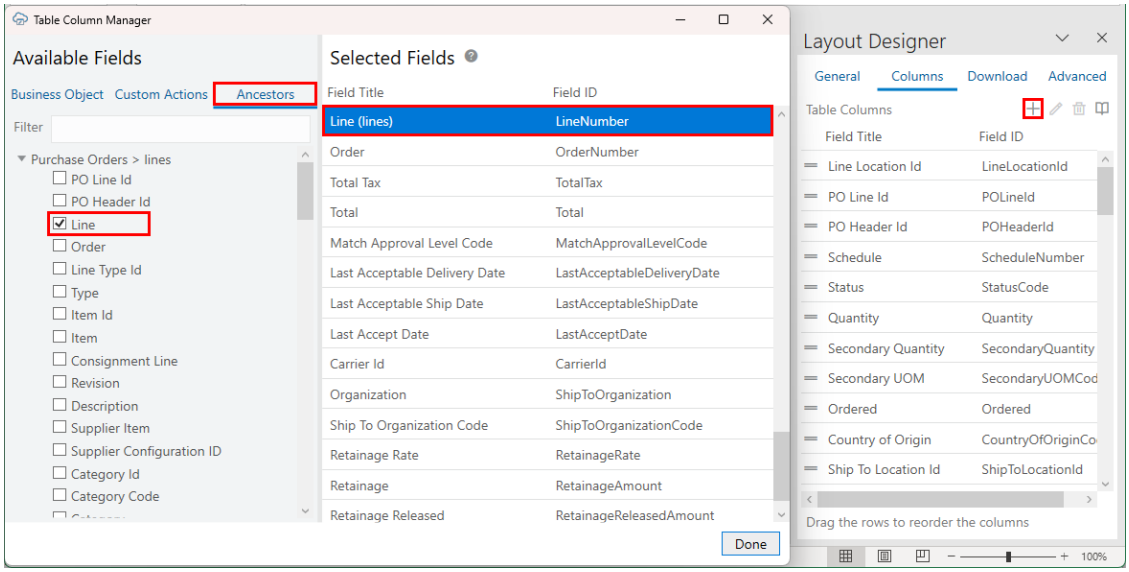
Let's suppose you have a set of dependent layouts where **purchaseOrders** is the parent, **lines** is the child, and **schedules** is the grandchild. When a business user is looking at rows in the **schedules** layout, it may not be readily apparent which schedule is attached to which line.

To help them sort out which is which, you could add a column from the parent layout (**lines**) to the child layout (**schedules**), such as `POLineId`.

> **Note:**
>
> The field must be included as a column or form field in the ancestor layouts. If you don't see the field you want, you'll need to add it to the ancestor layout.

To add an ancestor column to your layout, open the Table Column Manager from the Layout Designer, then select an appropriate ancestor field from the Ancestors tab.

> **Tip:**
>
> To add an ancestor column before an existing column in the table, select the existing column in **Selected Fields** list, then select the ancestor column check box. For example, to display an ancestor column first in the table layout right after the Status column, select the first column.



See Manage Fields in a Form or Table for more details.

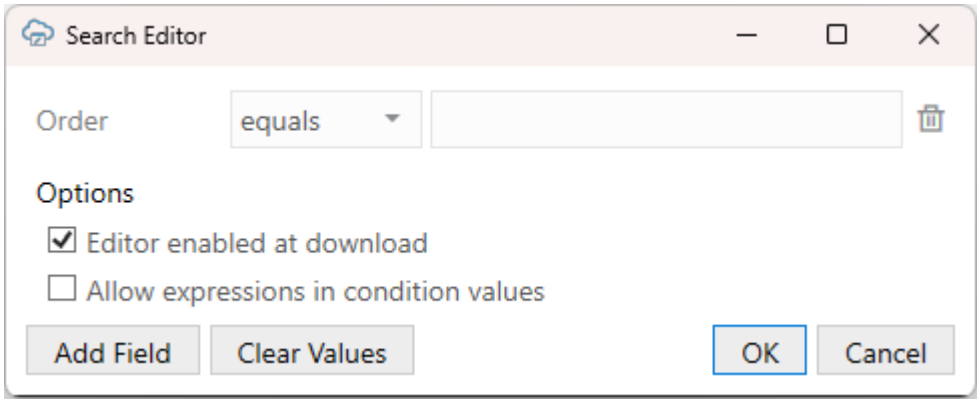
Filter Data for a Set of Dependent Layouts

You can use all the search options described in Configure Search Options for Download to determine the rows downloaded for the primary layout. The add-in downloads all the children and other descendant rows for the row or rows in the primary layout.

If you would like to further restrict the children or descendant rows downloaded for a given dependent layout, you can define one or more download parameters in that layout.

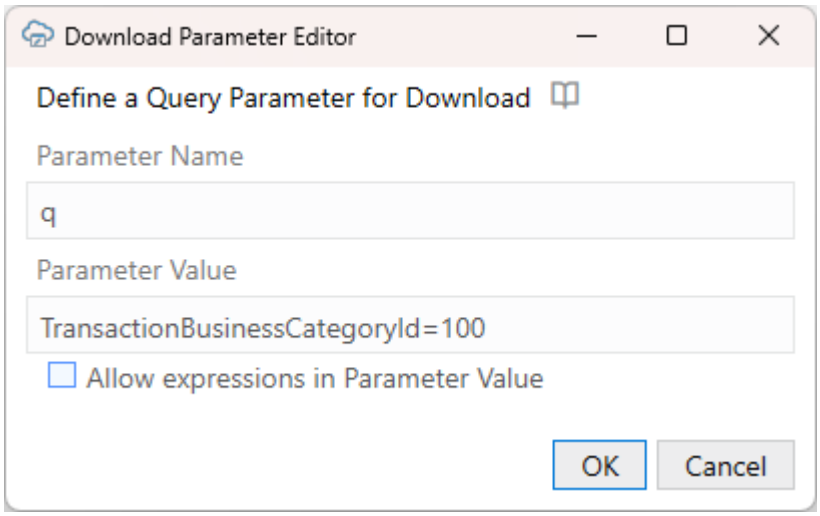
Take our example of a hierarchy where **purchaseOrders** is the parent, **lines** is the child, and **schedules** is the grandchild. In this scenario, you may have a Form-over-Table layout as the primary layout with **purchaseOrders** in the form and **lines** in the table. You also then have a subordinate Table layout for your **schedules** business object.

To show a single purchase order in the form, you would configure a search in the Layout Designer that prompts the user to enter an order number like this:



Without any other download parameters, Oracle Visual Builder Add-in for Excel populates the form with the user-provided purchase order, and the two tables with all available lines and schedules associated with this purchase order.

To limit the lines and schedules, you can configure one or more download parameters on each table using the Layout Designer. For example, to show all schedules with the same transaction business category, create a search parameter for the schedules Table layout, such as `q=TransactionBusinessCategoryId=100`. On download, the add-in returns all schedules with the same transaction business category (in this case, with an ID of "100").



This example produces a set of GET requests like this:

```
/purchaseOrders/{purchaseOrders_Id}/child/lines/{purchaseOrders_lines_Id}/
child/schedules?q=TransactionBusinessCategoryId=100
```

The add-in manages the path parameters automatically.

> **Note:**
>
> Some download parameters may not produce the desired result on dependent layout download. For example, using a "order by" parameter will not work as expected since the add-in sends multiple separate requests for child resources. Parameters such as "order by" should not be used.

Refer to Use the Search Editor to Find Required Data and Use Download Parameters to Limit Downloaded Data.

Download, Upload, and Clear Operations on Dependent Layouts

When you download, upload, or clear data for a layout in a dependent hierarchy, the operation takes effect on all layouts in the hierarchy, starting with the primary layout, progressing to the next layout in the hierarchy, and continuing down until the last level in the hierarchy.

If the layout is not part of a set of dependent layouts, the operation is performed on the active layout only.

Settings such as those for macros only apply to the primary layout and are not enforced on other layouts in the hierarchy.

Downloading Data

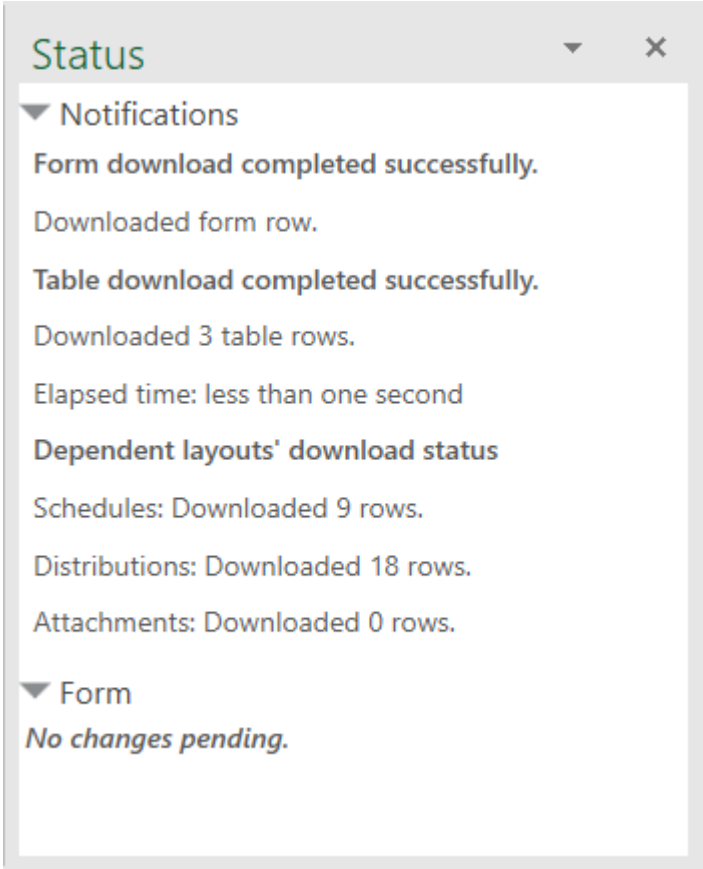
On download, Oracle Visual Builder Add-in for Excel first checks all layouts in the hierarchy for any pending changes. If there are changes pending, the user is prompted to confirm the download operation. If the user chooses to proceed, all pending changes are lost.

During download, the add-in first retrieves the values for the primary layout from the REST service. After the primary layout is populated, the add-in makes the next worksheet in the hierarchy active and retrieves all the appropriate items.

All matching items for all rows from the parent layout are downloaded at each level. For example, when Sheet 1 in your workbook contains Purchase Orders as the parent and Lines as the child (containing, say, 10 Lines) and Sheet 2 contains Schedules as the grandchild, the Schedules table is populated with all Schedule items for all Lines. If each of the 10 Lines had two Schedules, the Schedules table would download 20 Lines.

The download operation proceeds through the rest of the Table layouts in the hierarchy, retrieving all matching items for all rows from the parent layout. See Dependent Layout Download.

When the operation finishes, the primary layout becomes active and the Status Viewer shows results for the primary layout as well as a summary for each layout in the dependent hierarchy, as shown in this example for a download operation:



Following a download, you can edit data much as you would in a Table or a Form-over-Table layout.

Uploading Changes

On upload, the add-in makes the primary layout active and sends all pending changes. If the primary layout is a Form-over-Table layout, changes are sent first from the form and then from the table. The add-in then moves to the worksheet with the first dependent Table layout and uploads changes before proceeding to the next layout.

Pending changes may include creation of new items, update or deletion of existing items, and invocation of actions on items. For rows pending Update, values in the ancestor column cells are not uploaded.

For new items on layouts below the primary layout, values in a parent column must match a row in the parent layout. For example, to create a new distribution, you must specify an existing schedule in the parent column with which to associate the new item. Grandparent and higher columns are read-only and can't be updated. Empty parent column cells in the dependent layout or in its immediate parent layout result in creation failing.

The match is performed across all ancestor column cells using the local cell values from the parent table only. The service is not contacted while doing this matching. If one row in the parent table matches, it is used as the parent row. If more than one row matches or no rows match, then the row is marked as "Create Failed".

When the operation finishes, the primary layout becomes active and the Status Viewer shows results for the primary layout as well as a summary for each layout in the dependent hierarchy.

For more information about, and options for, upload, see Upload Changes.

Clear

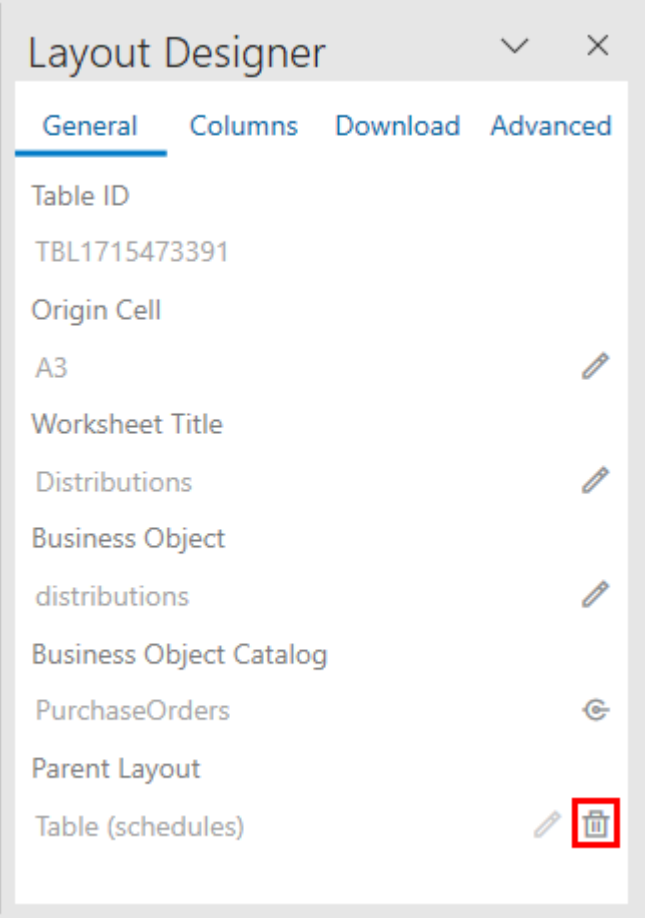
When the clear operation is performed, data is cleared from all the layouts in the dependent hierarchy.

Delete a Dependent Layout

When your layout is part of a hierarchy of dependent layouts, the layout cannot be deleted without first removing its dependency in the layout hierarchy.

To delete a dependent layout:

1. Open the Layout Designer of the Excel worksheet whose layout you want to delete.
2. In the General tab, click the Remove Dependency icon (🗑) next to Parent Layout.



3. When prompted, click **Yes** to remove the dependency.
4. Click **Delete Layout**, then confirm your selection.

Requirements for Dependent Layouts

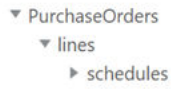
To ensure that your dependent layouts function without error, Oracle Visual Builder Add-in for Excel requires that the REST service complies with the requirements set out here.

URL Path Requirements

To ensure all operations of dependent layouts can be done without errors, the add-in requires the following:

- **Parent business object**:
  - **Collection path**: `/{parentResource}`
  - **Item path**: `/{parentResource}/{parentResource_Id}`
- **Child business object**
  - **Collection path**: `/{parentResource}/{parentResource_Id}/{childResource}`
  - **Item path**: `/{parentResource}/{parentResource_Id}/{childResource}/{childResource_Id}`
- **Grandchild business object**
  - **Collection path**: `/{parentResource}/{parentResource_Id}/{childResource}/{childResource_Id}/{grandchildResource}`
  - **Item path**: `/{parentResource}/{parentResource_Id}/{childResource}/{childResource_Id}/{grandchildResource}/{grandchildResource_Id}`

This example shows the collection and item paths for the parent, child, and grandchild business objects in the following three-level hierarchy:



Parent paths:

- Collection path: `/PurchaseOrders`
- Item path: `/PurchaseOrders/{PurchaseOrders_Id}`

Child paths:

- Collection path: `/PurchaseOrders/{PurchaseOrders_Id}/lines`
- Item path: `/PurchaseOrders/{PurchaseOrders_Id}/lines/{lines_Id}`

Grandchild paths:

- Collection path: `/PurchaseOrders/{PurchaseOrders_Id}/lines/{lines_Id}/schedules`
- Item path: `/PurchaseOrders/{PurchaseOrders_Id}/lines/{lines_Id}/schedules/{schedules_Id}`

GET and POST Response Requirements

GET and POST responses must contain self links that uniquely identify a record. For example:

```
"links": [
        {
          "rel": "self",
          "href": "http://localhost:8888/ords/hr_rest/ExpenseReports/15001"
        }
 ]
```

Notes on Oracle REST Data Services Support

- Use version 22.1.0 or later if you plan to create a set of dependent layouts for ORDS. Previous versions of ORDS are known to have an issue with an incorrect payload definition for the POST handler. See ORDS Release Notes 22.1.0.
- ORDS AutoRest does not provide service paths as described in this section. However, you can write custom ORDS using SQL and PL/SQL to satisfy these requirements. See ORDS RESTful Web Services Architecture Diagrams and Manually Creating RESTful Services Using SQL and PL/SQL.

15

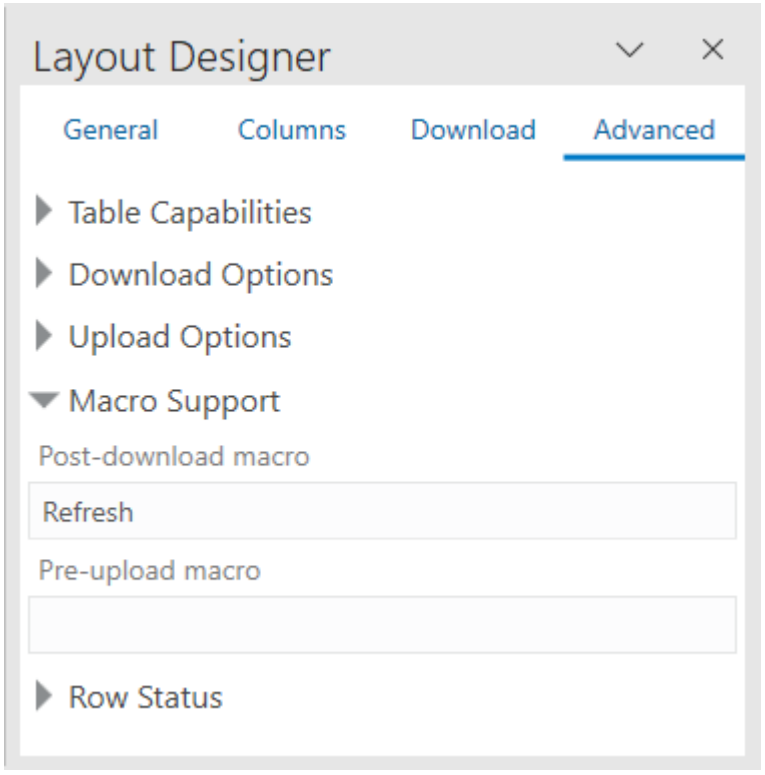
Use Macros in an Integrated Excel Workbook

You can configure macros that Oracle Visual Builder Add-in for Excel runs at specific points in the lifecycle of an integrated Excel workbook.

Use of this functionality requires you to use the Excel macro-enabled workbook type (`.XLSM`) and create your macros in a macro module. For more information about creating macros, see Microsoft documentation; describing how to create macros in an Excel workbook is outside the scope of this guide.

Some companies block the usage of Excel macros because they do not think macros are sufficiently secure. Consider your intended audience before you add a macro. You are also responsible for the security risks involved in using macros. So research the risks thoroughly before you deliver an integrated workbook to your customers. After creating a macro, take steps to protect the macro both from malicious and accidental alterations that might produce unexpected or harmful results. If a macro results in changes that are incompatible with the add-in or results in undesirable behavior, change the macro to avoid this behavior.

The Layout Designer's Advanced tab provides two properties where you can specify macros: the **Post-Download Macro** to run after download completes and the **Pre-Upload Macro** to run before an upload begins. Provide your macro names as the values of these properties. For example, when you've created a Refresh macro that you want to run after an upload, enter `Refresh` as the value of the **Post-Download Macro** property.



> **Tip:**
>
> Do not include the parentheses when specifying the name of the macro.

The macro that you specify for the **Post-Download Macro** property is not used if the user cancels download, if the table or form is empty, or in the event of an unexpected error. The macro that you specify for the **Pre-Upload Macro** property is used just before an upload. If the macro returns any value other than `true`, the upload operation quits and a notification appears in the Status Viewer. If the macro returns `true`, upload proceeds normally. To return a `true` or `false` value from a macro, define a Boolean Function. See Microsoft documentation for details.

Here's example logic of an `IsUploadReady` function for a Pre-Upload Macro:

```
Function IsUploadReady() As Boolean
   Dim returnVal As Boolean

   On Error GoTo ErrHandler:

   Dim table As Range
   Set table = Sheets("Sheet1").Range("TBL349543489")
   ' The named range, TBL349543489, is managed automatically by the add-in

   returnVal = True

   Dim cRows As Long
   cRows = table.Rows.Count
   Dim currentTableRow As Long
   Dim amount As Long
   For currentTableRow = 2 To cRows ' start with 2 to skip header row
       amount = table(currentTableRow, 10) ' Amount is the tenth column in
the table
       If amount < 0 Then
           returnVal = False
           Debug.Print "Found negative amount = "; amount
       End If
   Next

   IsUploadReady = returnVal
   Exit Function

ErrHandler:
   Dim failureMessage As String
   failureMessage = Err.Description
   MsgBox failureMessage
   IsUploadReady = False
   Exit Function
End Function
```

When an error occurs during the execution of a macro, Excel displays a Microsoft Visual Basic window to the user. We recommend that you implement a robust error handling strategy so that the window displays a useful message to the user who encounters an error during macro

execution. The following is a simplistic example. The appropriate error handling strategy for a given macro depends on the logic in the macro.

```
Sub Refresh()

    On Error GoTo ErrHandler:

    ActiveWorkbook.RefreshAll
    Exit Sub

ErrHandler:
    Dim failureMessage As String
    failureMessage = Err.Description
    MsgBox "Unable to refresh. Details: " & failureMessage
    Exit Sub
End Sub
```

> **Tip:**
>
> The add-in creates and maintains named ranges for the data table. Your macros should never modify these named ranges. However, your macros can access the named range to locate the data table on a dynamic basis.

Notes on Macros

- Macro recording is incompatible with add-in features such as download and upload and is not supported. Do not attempt to record any add-in features. In some cases, you may see unexpected exceptions.
- Do not leave the Excel Visual Basic editor's break mode on when you use **Download Data** or **Upload Changes**. It is not supported and can result in an unexpected exception.
- While the add-in runs either of the macros described in this topic, Excel events are disabled.

16

Publish an Integrated Excel Workbook

Once you complete configuring an Excel workbook, you can publish it for users who perform data entry. Publishing creates a copy of the workbook and prepares it for distribution; for example, it lets you hide the Design tools and turn on worksheet protection for each worksheet with a layout.

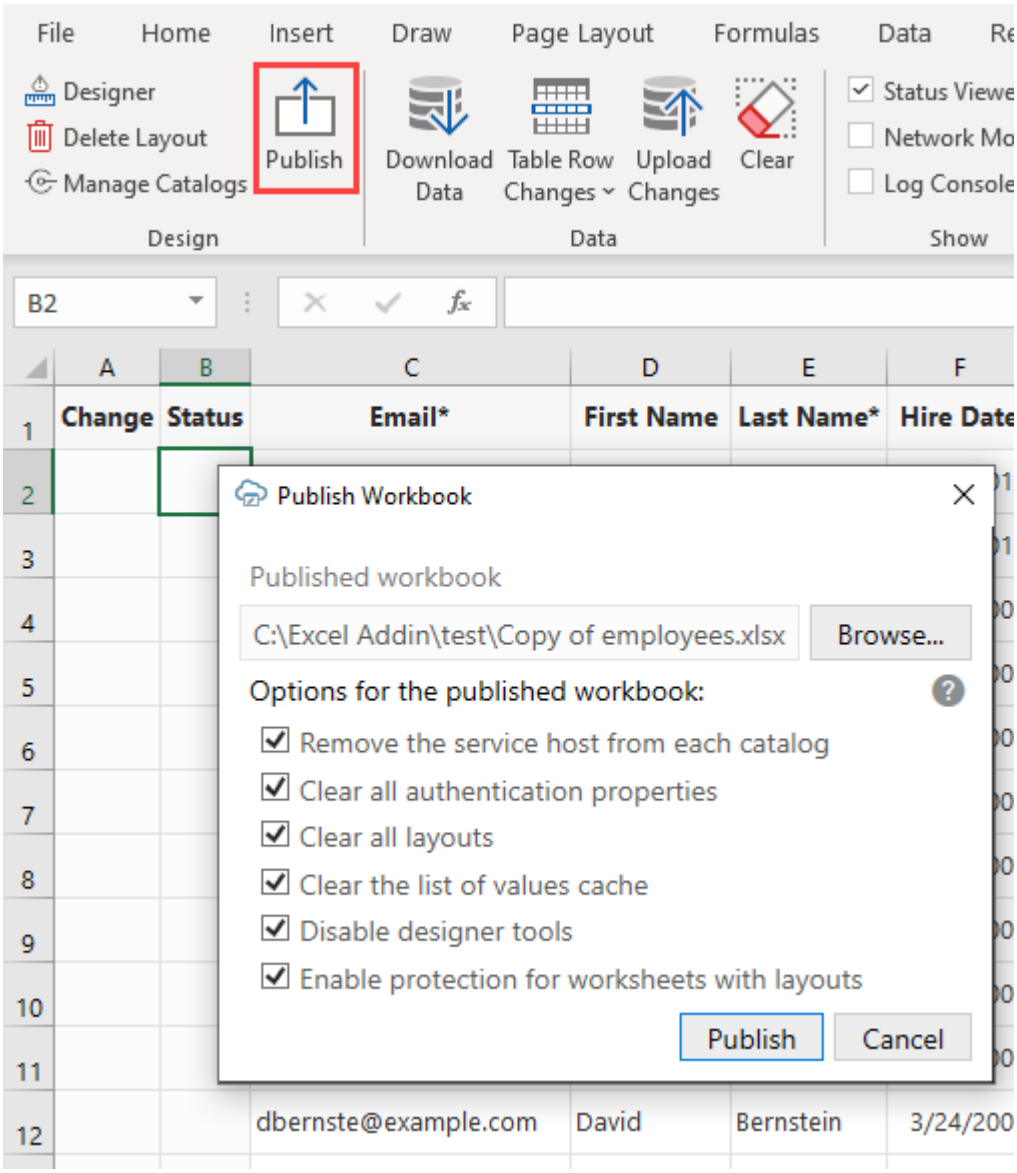
Alternately, you can use publish to create a clean, unlocked copy. See Publish an Unlocked Copy.

Publishing is optional. All data editing features of an integrated workbook are available in both published and unpublished copies of the workbook.

> **Note:**
>
> If you plan to distribute your integrated workbook publicly, use the Publish options to create a clean copy for distribution to avoid exposing internal information such as host names.

To publish an Excel workbook:

1. Complete configuration of the workbook.
2. Test the configuration thoroughly.
3. Use Excel's Inspect Workbook feature to review and remove personal information from the workbook.
   Access the Inspect Workbook feature from Excel's File menu. When you use the Document Inspector to choose content to inspect and potentially remove, ensure that the Hidden Worksheets check box is not selected. You must not remove hidden worksheets from workbooks that you distribute because the add-in uses hidden worksheets to integrate the Excel workbook with the REST service.
4. In the Oracle Visual Builder tab, click **Publish**.



The Publish Workbook window opens. If the name of the original workbook ended with `-source` (for example, `employees-source.xlsx`), the add-in will offer the same name without `-source`.

5. To change the published workbook's directory and file name, enter the desired name and location in the Publish Workbook window, or click **Browse**.
   Make sure the published workbook doesn't use the same name as any open workbook (Excel won't let you use the same name for different workbooks, even if the file paths are different).

> **Note:**
>
> You cannot publish to a location that starts with `http` such as OneDrive.

6. In the Publish Workbook window, choose other options for your published workbook:

Check Box	Use
Remove the service host from each catalog	This check box is selected by default. Leave it selected to ensure that users must enter the service host when they open the published workbook. This option lets users connect to another service host to access the REST service.
Clear all authentication properties	Select this check box to remove authentication configuration from the published copy of the workbook. If you are distributing a sample workbook outside of your organization, you may want to select this option to avoid exposing potentially sensitive information. You should always choose this option if the workbook is posted in a public place. However, if you are distributing a fully-configured workbook to business users inside your organization, you may want to deselect this option so that the target business users can log in using the properties you have chosen. See Authentication Options.
Clear all layouts	Select this check box to clear data downloaded to each layout in the published workbook. This option lets users download the latest data from the REST service in the published workbook.
Clear List of Values Cache	Select this check box to clear the list of values cache in the published workbook. This option is important if users might have different choices for a list of values in the published workbook.
Disable Designer tools	Select this check box if you don't want users of the published workbook to modify its layout. This option hides the design tools in the published workbook. If the add-in was installed with the design tools already disabled, this option has no additional effect. **Note:** This check box is disabled if the workbook uses OAuth2 authentication *and* **Clear all authentication properties** is selected.
Enable protection for worksheet with layouts	Select this check box if you want to prevent users from modifying read-only fields and performing Excel actions in the published workbook. This option enables Excel's worksheet protection for each worksheet with a layout.

7. When you have adjusted the settings as required, click **Publish**.

You'll see a notification in the Status Viewer that the workbook has been published to the specified directory.

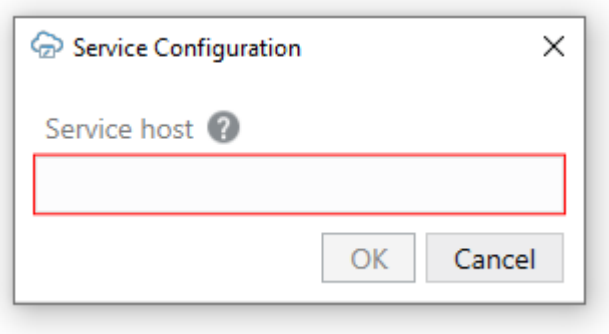
8. Save the source version of the workbook, in case you need to make configuration changes post-publication.

Now that your workbook is published, you can distribute it to users for data entry.

Differences Between a Published and a Source Workbook

There are a number of differences with respect to the source workbook:

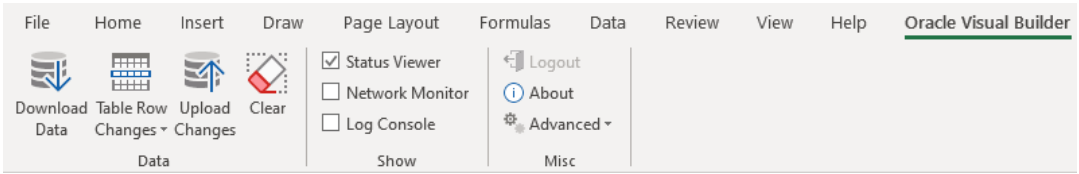
- If the service host value was removed for a workbook, users who open that workbook and perform an action that requires access to the REST service are prompted to enter the service host value, as shown here:



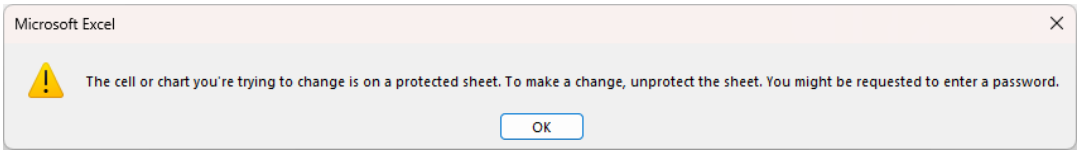
Actions that require access to the service include the Download Data and Upload Changes commands.

> **Tip:**
>
> To change the URL of the service host when you don't have the Design tools installed, use the **Edit Service Host** option in the Advanced menu.

- If the Designer tools were disabled, tools such as the Designer, Delete Layout, and Publish do not appear in the Oracle Visual Builder tab, as shown here:



- If worksheet protection was enabled and a user tries to modify a read-only field, a message similar to the following image is shown:

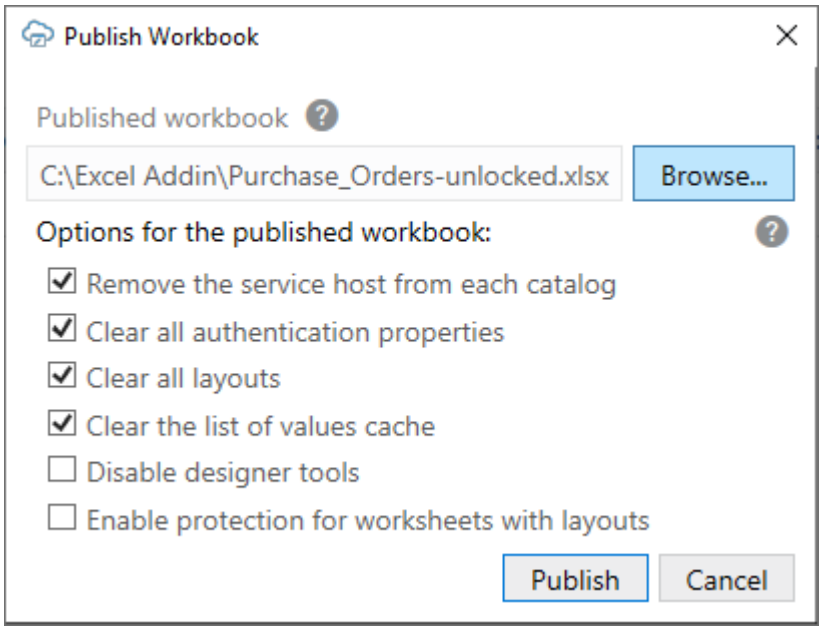


Worksheet protection also prevents the user from performing Excel actions that might disrupt the workbook's integration with the service. See Understanding Read-Only Behavior in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

- When a workbook is published all Workbook Parameters are removed. See Test a Download with Workbook Parameters.

Publish an Unlocked Copy

If you are distributing a sample integrated workbook and expect the target audience to modify the configuration, you may want to publish a copy with these settings:



17

REST Service Support

This chapter provides additional technical details about how Oracle Visual Builder Add-in for Excel supports integration with REST services. It also provides information about technical known issues and limitations.

Service Types

Oracle Visual Builder Add-in for Excel provides support for the following REST service frameworks:

- Oracle ADF REST Resource
- Visual Builder Business Objects
- Oracle REST Data Services (ORDS)
- NetSuite SuiteTalk REST Web Services
- Other services

Oracle ADF REST Resource

ADF REST services provide rich service metadata that Oracle Visual Builder Add-in for Excel can analyze to provide a business object catalog with many details already filled out.

ADF REST services include many services offered by Oracle Cloud Applications. When the add-in integrates an Excel workbook with ADF REST services, it supports special features such as:

- Search editor
- Row finders
- Multi-row upload
- Polymorphic business objects
- Custom actions
- Upsert mode

- Business object hierarchies

See Consuming ADF RESTful Web Services.

Support for Date Effective Objects

If you are working with a REST service that includes Date Effective objects, you can configure a row variable that allows your business users to specify a start date and an `Effective-Of` REST request header that references that start date. See Configure a Row Variable for a Layout and REST Request Headers.

For more information about Data Effective objects in ADF REST services, see Manage Date Effective Objects in the *REST API for Oracle Fusion Cloud HCM* guide.

Visual Builder Business Objects

Oracle Visual Builder Add-in for Excel supports catalogs that consist of custom business objects from Visual Builder applications that use the Visual Builder Business Objects (VBBO) API.

When the add-in integrates an Excel workbook with VBBO, it supports special features such as:

- Multi-row upload
- Custom actions (known as **Object Functions** in Visual Builder)
- Search editor

These features are not supported for VBBO:

- Automatic configuration of filter parameters for Lists of Values
- Oracle Fusion Applications Token Relay Authentication
- Row finders
- Polymorphic business objects

See Accessing Business Objects Using REST APIs.

Compatibility

Workbooks with a catalog of type VBBO cannot be used with add-in versions before 3.6.

VBBO catalogs created with versions 3.5 and earlier display an API type of "ADF REST" in version 3.6 and later. If desired, you can update the API type from the Advanced tab of the Business Object Catalog Editor.

Oracle REST Data Services

Oracle Visual Builder Add-in for Excel supports Oracle REST Data Services (ORDS) when you provide OpenAPI service metadata for an ORDS service. ORDS with AutoREST can provide an OpenAPI service metadata document.

For example, use `http(s)://myhost.example.com:8888/ords/hr_demo/open-api-catalog/employees/` where:

- `myhost.example.com:8888` is the host and domain portion
- `hr_demo` is the schema/application
- `employees` is the database table

For information about AutoREST, see Automatic Enabling of Schema Objects for REST Access (AutoREST) in the *Oracle REST Data Services Developer's Guide*.

For manually-created REST services using ORDS, you'll need to define modules, templates, and handlers in order to get an OpenAPI service metadata document. See Manually Creating RESTful Services Using SQL and PL/SQL in the *Oracle REST Data Services Developer's Guide*.

After importing ORDS service metadata, you can use the Business Object Field Editor to provide additional information about each field to improve the overall user experience. For example:

- Edit the field titles
- Designate certain fields as required
- Define lists of values. See Use Lists of Values in an Excel Workbook.

Known Issues with ORDS

Refer to these known issues when planning to use Oracle REST Data Services:

- In some cases, the ORDS server returns Create Failed for rows, when in fact the Create operation was successful. Re-downloading rows into the table will show the created rows.
- With an ORDS service, the PUT operation on the item path performs an "upsert" (see Update/Insert Table Row in *Oracle REST Data Services Developer's Guide*). So if you are about to update an existing row and someone else deletes that row, your update attempt may re-create that row. There's no warning or notice when this behavior occurs.

NetSuite SuiteTalk REST Web Services

Oracle Visual Builder Add-in for Excel provides limited support for integrating Excel workbooks with NetSuite services.

Unlike other services, you'll need to do some manual configuration to the NetSuite catalog to get up and running.

For layouts that reference parent-child business objects, you'll need to import a catalog and add required child business objects before you create your layout. If you try to create a layout without first doing these tasks, the generated catalog will be missing the child business objects. See Configure a NetSuite Catalog for Parent-Child Business Objects.

You may also be required to configure some fields following Table layout creation. See Add NetSuite Reference Fields for a Table Layout.

About NetSuite Services

When you create a layout, you'll need to provide an appropriate NetSuite URL for the service metadata document. You'll also need to provide a config file with OAuth 2.0 settings for your account since NetSuite services require OAuth 2.0 to authenticate.

To support OAuth 2.0 authentication with the add-in, you'll also need to create a NetSuite integration record to use with your integrated workbook.

Review this topic for information about NetSuite service metadata, OAuth2 authentication, and integration records.

NetSuite Concepts

Here are some key NetSuite concepts you should be familiar with:

- **Record**: The NetSuite concept of a "Record" is roughly equivalent to the concept of "Business Object" used in this document.
- **Reference Field**: NetSuite records support a concept known as a "Reference Field" where a field in one business object can reference a row in a different business object. For example, the NetSuite "contact" record has a field called "company". The company field can refer to a customer, partner, vendor, and so on.

For more information on NetSuite services, see SuiteTalk REST Web Services Overview and Setup in the *NetSuite Applications Suite* documentation.

NetSuite Service Metadata

NetSuite provides support for retrieving OpenAPI service metadata for NetSuite records available via REST. For example, to obtain the service metadata for "contact", use a URL similar to:

```
https://<YOURACCOUNT>.suitetalk.api.netsuite.com/services/rest/
record/v1/metadata-catalog?select=contact
```

> **Note:**
>
> The add-in can retrieve and process the entire NetSuite catalog using `...services/rest/record/v1/metadata-catalog` (without `?select`). However, it is very large and takes a long time. So, selecting a specific record is recommended.

For information on OpenAPI metadata with NetSuite services, see Working with OpenAPI 3.0 Metadata.

OAuth2 Authentication

NetSuite services require OAuth 2.0 authentication. When you import a catalog or create a layout, you'll select OAuth 2.0 Authorization Code (PKCE) from the first screen of the wizard. The add-in then displays a screen for entering the required authentication properties. This screen includes an **Import** icon ( ) you can use to import a JSON configuration file with the required values for your workbook. See Configure OAuth 2.0 Authorization for a Catalog.

Here is a sample OAuth2 config file for NetSuite:

```
{
  "type": "oauth2",
  "authorizationCode": {
    "clientId": "<value provided by your account admin>",
    "authorizationEndpoint": "https://
<YOUR_ACCOUNT>.suitetalk.api.netsuite.com/app/login/oauth2/authorize.nl",
    "redirectionEndpoint": "<value provided by your account admin>",
    "accessTokenScope": "rest_webservices",
    "tokenEndpoint": "https://<YOUR_ACCOUNT>.suitetalk.api.netsuite.com/
services/rest/auth/oauth2/v1/token"
  }
}
```

Copy this sample, paste it into a plain text file, then save it with a file name such as `NetSuite-OAuth2-Config.json`. You can then ask your NetSuite account admin to fill in the missing values and return it to you.

> **Note:**
>
> If your NetSuite account admin needs some pointers on the missing values, see #GUID-1FDF1442-3A22-462B-8FFE-12B072995DAC/GUID-1390C97B-5BE9-4E69-9523-67BA8802F2A1.

Integration Records

If you have not yet created an appropriate NetSuite Integration Record to use with your integrated workbook, refer to Integration Record Overview and Create Integration Records for Applications to Use OAuth 2.0 in the *NetSuite Applications Suite* documentation.

To support OAuth 2.0 authentication with the add-in, the integration record must have these configuration settings enabled:

- **Public Client** check box. This is required since the add-in is considered a public client.
- **REST Web Services** check box

> **Note:**
>
> Make sure to capture the client ID during this process.

Once you have an appropriate integration record, it can be used with multiple different integrated workbooks that are integrated with NetSuite services that use this account.

For more information on how the add-in supports OAuth2, see OAuth 2.0 Authorization Code Flow with PKCE.

Configure a NetSuite Catalog for Parent-Child Business Objects

Before you can create a layout or layouts from a NetSuite service that references parent and child business objects, you'll first need to import the catalog and manually add the required child business objects. The NetSuite service metadata document does not provide child paths.

You'll need to complete this task if you want to create a Form-over-Table layout or a Table layout with one or more dependent layouts at the child level.

Suppose you want to create a Form-over-Table layout with a "Sales Order" business object in the form and the child business object, "Items", in the table. Because the NetSuite service metadata document doesn't provide the child path, you won't see the Items child object in the New Layout Setup wizard when you try to create your Form-over-Table layout.

Instead, you need to start by importing the catalog, then adding the child object. When you do this, you'll provide the collection and item paths as well as configure path parameters. You'll also need to add all required fields for the child object.

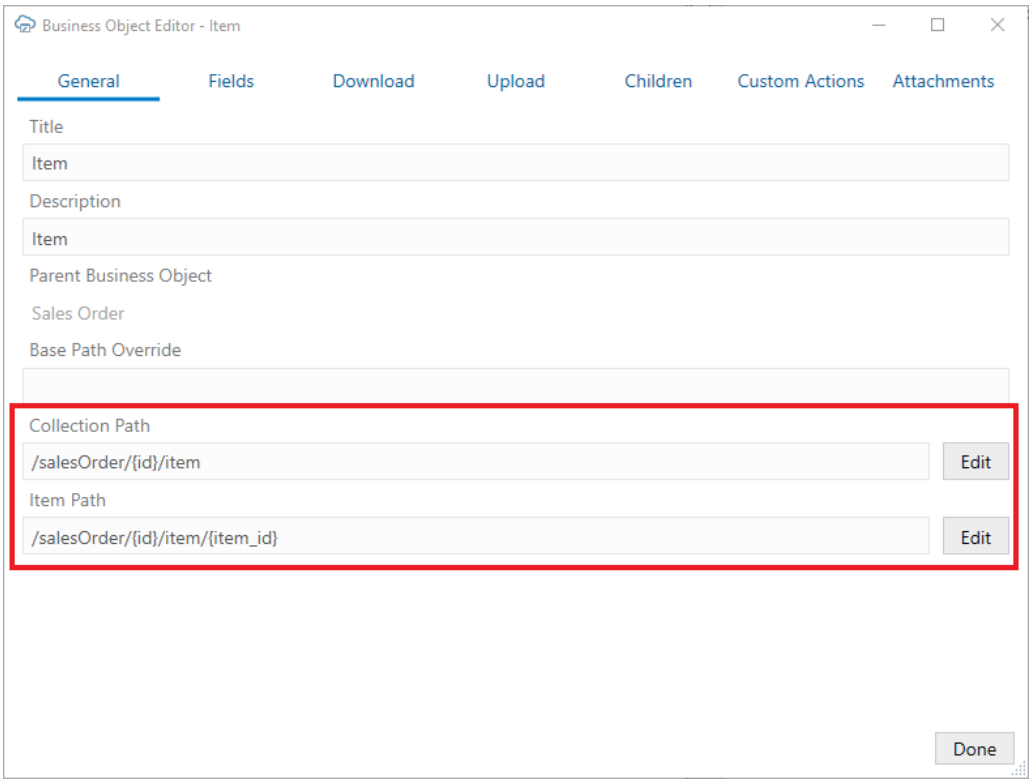
For more information about parent and child path requirements, see Requirements for Dependent Layouts.

> **Note:**
>
> This task covers adding a child business object, configuring it, and then adding fields. Before you begin, import your catalog as described in Import a Business Object Catalog.

To configure a child business object for a NetSuite catalog:

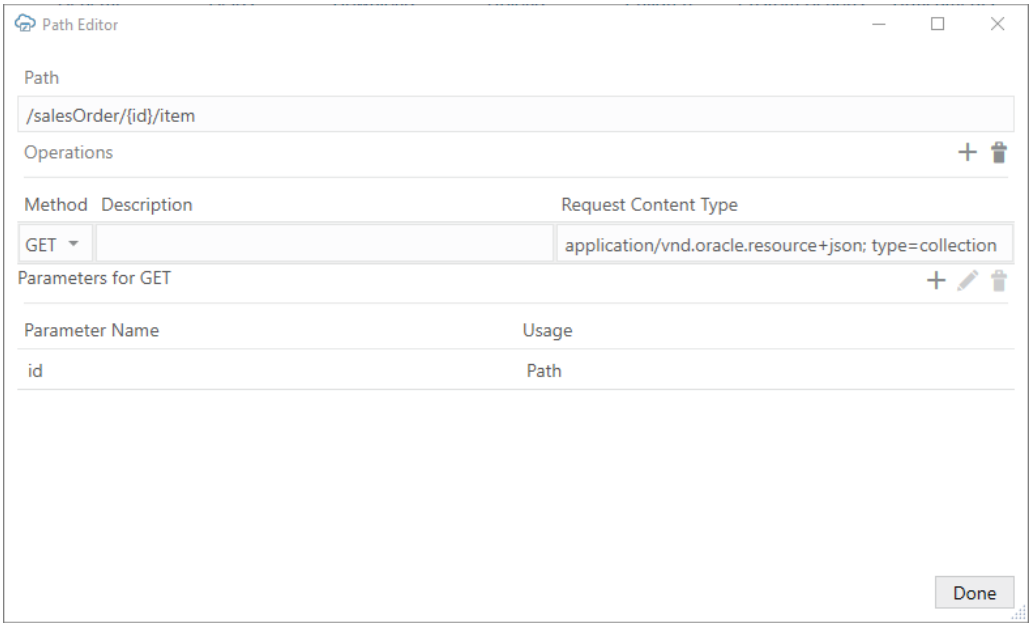
1. Add a child business object to the catalog:
   a. Open your new catalog, then open the **Business Object Editor** for the parent object, "Sales Order".
   b. From the **Children** tab, click **Add Child Business Object** (+).
2. Configure your new child object:
   a. Open the new child business object in the **Business Object Editor**, then provide details such as a title and description for the business object.
   b. Enter the collection and item paths for the child object.

      The collection and item paths may look like this for an Item object.



   c. Click the **Edit** buttons for each path, then edit the path parameters from the Path Editor.

      Add all REST request methods that are supported by the services. This image shows the GET method configured for the collection path.

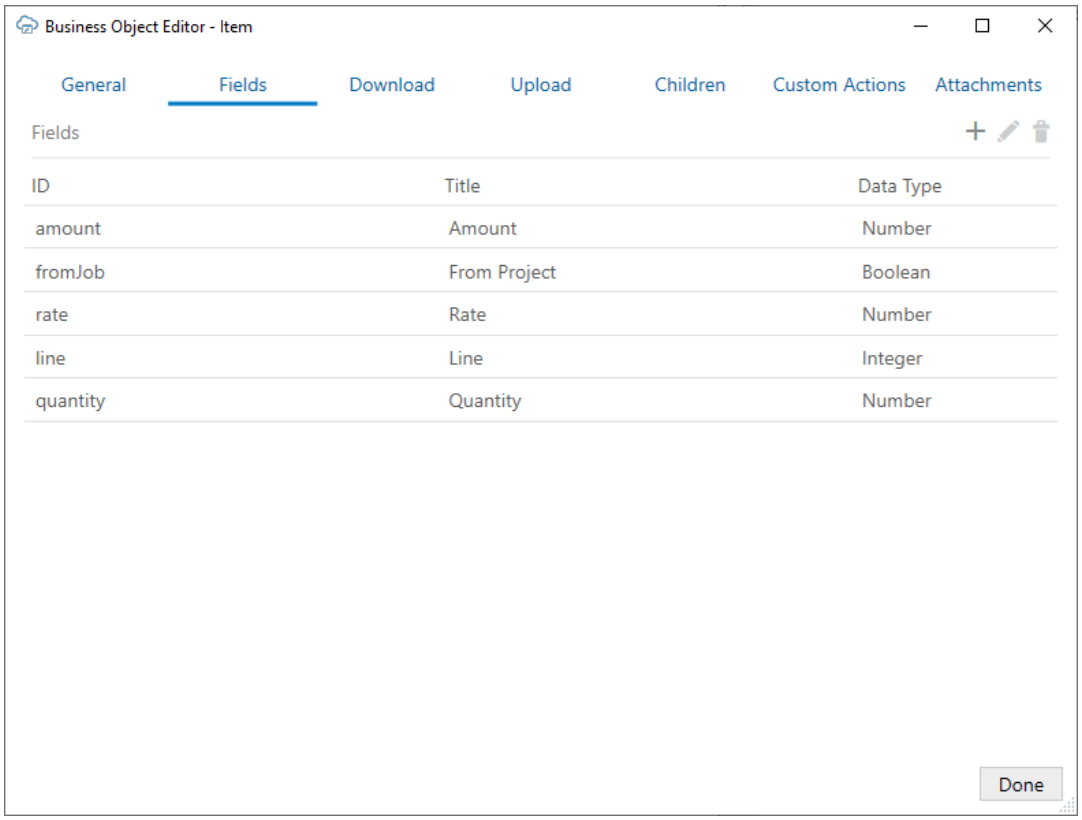


d. From **Download** tab, make these changes:

- Leave the **Offset Parameter Name** and **Limit Parameter Name** fields empty.
- Enter "items" in the **Response Payload Items Member Name**" field.

3. Add fields for the child object from the Fields tab of the editor. For information about available fields, see NetSuite REST API Browser.

   This image shows configured fields for the Items child object.



When you are finished, create your layouts in the usual way. You'll see the nearly-added child business object in the New Layout Setup wizard when prompted to choose a child business object for the table part of a Form-over-Table layout or for a dependent layout. See Create a Table Layout in an Excel Workbook or Create a Form-over-Table Layout in an Excel Workbook.

> **Note:**
>
> When the business user edits the form, the downloaded data may be marked as invalid. If these two conditions are met, then clear the value and upload changes:
>
> - The cell value is not changed after download but is still marked as invalid.
> - The field's **Omit from payload if value is empty** check box is selected.

Add NetSuite Reference Fields for a Table Layout

When you create a Table layout for a NetSuite business object, some fields may be missing due to limitations in the service metadata. Here are the steps to add a "Company" field for a "contact" business object.

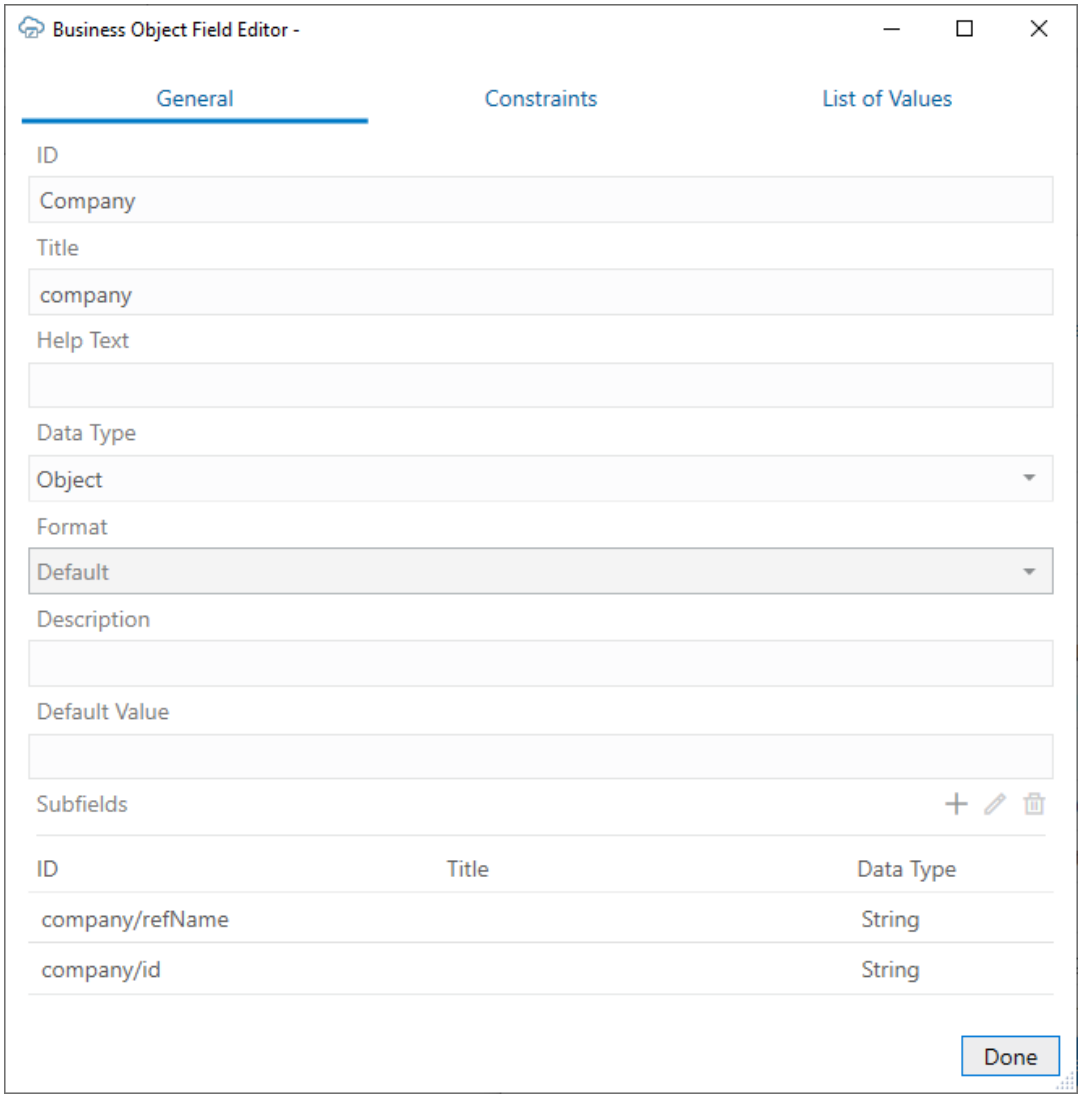
Before you begin, create a Table layout for the contact business object. You'll be prompted to provide the URL to a service metadata document as well as an OAuth2 configuration file. See About NetSuite Services.

To add a NetSuite reference field:

1. Open the Business Object Editor for the "contact" business object, then click the **Fields** tab.
2. Click **Add Field** (+), then provide details for the Company field:
   - **ID**: company
   - **Type**: Object
   - **Subfields**: Add two subfields of type string with ID values of "company/refName" and "company/id".

     > **Note:**
     >
     > Keep in mind that IDs are case sensitive and must match the NetSuite JSON member names exactly.

   - Provide **Title** and **Help Text** values as desired.



3. Return to the table layout.
4. From the Layout Designer, click the **Columns** tab, then click **Manage Columns** (+).
5. From the Table Column Manager, add the subfields to the layout as needed. For example, you might find it useful to add the "company/refName" to your contact layout.

> **Note:**
>
> There is no support for including additional fields from the referred record in the referring table. So, using the example above, you cannot include additional company (customer, partner, vendor, etc.) fields in the contact table.

For more information about the contact record, see contact in *NetSuite REST API Browser: Record API v1*. See also Format of Selects and References in the *NetSuite Applications Suite* documentation.

NetSuite Support Limitations and Known Issues

Here are some things to keep in mind when using creating layouts for NetSuite records.

Limitations

These NetSuite features are not supported by the add-in:

- Record actions
- Transforming records
- Upsert operation

These add-in features are not currently available for integrated workbooks connected to NetSuite services:

- Send only changed data

Known Issues

Feature Area	Issue	Description
Mandatory Properties	The add-in cannot properly set default values for the "Required for Update" and the "Required for Create" properties for a field.	The OA3 service metadata document does not provide the "Required" property for schema members (fields). You can manually configure these properties using the Business Object Field Editor. See Configure Business Object Fields.
Data Download	Download performance is suboptimal.	A separate GET request is needed for each row because there is no way to get all fields for several rows in a single GET request. This limitation causes performance to suffer.
	The GET request for a single item does not include some of the expected fields in the response payload.	For example, "Last Sale Date" and "Opening Balance" in a "customer" business object.
	Query by Internal ID field returns an error	
Data Upload	Some fields in some NetSuite services do not accept a null value in a POST/PATCH request even though the service metadata document describes them as "nullable".	Workaround: Review the error message details to identify the fields causing the error, then configure these fields to omit empty values from the payload. See Omit Empty Values During Upload.
	After a successful upload to add or update a row, the row's values in the layout does not reflect changes made on the server.	In this case, the response does not include the updated fields. The user will need to re-download to see any changes.

Feature Area	Issue	Description
	When uploading Create/Update rows, the error message returned by the service is missing specific details.	You instead see a generic message such as "Error while accessing a resource. Field must contain a value." is returned without indicating which field it is.
	Conflict detection is not supported	If a user uploads changes to the service that overwrite another user's recent changes, the upload succeeds with no notification of the conflict.
Lists of Values	List of values are not configured automatically on business object fields.	Configure fields with list of values using a local data source. See Create a Local Data Source for a List of Values and Configure a List of Values with a Local Data Source.

Other Services

Oracle Visual Builder Add-in for Excel can also be used with other service types as long as the service behaves as the add-in expects.

When using another REST service, provide an OpenAPI service metadata document as you would for other service types. See REST Service Support Limitations for more information on the add-in's expectations for any service.

> **Note:**
>
> The Search Editor is not available for these services but download parameters can be used. See Use Download Parameters to Limit Downloaded Data.

Supported Data Types

The add-in supports a variety of OpenAPI data types available from REST services. These data types are derived from the service's JSON Schema Specification.

This table shows how JSON data types are mapped to the Data Type property for Business Object fields as shown in the Business Object Field Editor.

JSON Type	Business Object Field Data Type	Notes
Boolean	Boolean	
integer	Integer	`integer` is defined as a JSON number without a fraction or exponent part.
object	Object	See the subfields description in Configure Business Object Fields.
number	Number	
string	String	

JSON Type	Business Object Field Data Type	Notes
string	Date-time	When the OA3 `format` property is "date-time" See Date-Time Value Handling.
string	Date(no time)	When the OA3 `format` property is "date"
array	n/a	Not supported

For OpenAPI properties with type "string", the following format values are explicitly unsupported:

- time
- binary
- byte
- long-text

Other format values are ignored and the field is mapped to the String data type.

The add-in ignores fields with unsupported data types when you create a Table layout or Form-over-Table layout in the Excel workbook. If, for example, a service that you use to retrieve data includes the `binary` attribute data type, the add-in ignores it and does not create a column in the data table for this attribute type.

> **Note:**
>
> File, text, and web page type attachments are supported. See Create Layouts for Attachment Business Objects.

For more information, refer to the following OpenAPI and JSON resources:

- OpenAPI: https://swagger.io/specification/#dataTypes and https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.0.md#dataTypes
- JSON: https://datatracker.ietf.org/doc/html/draft-wright-json-schema-00#section-4.2

Business Objects Harvested from OpenAPI Metadata

Oracle Visual Builder Add-in for Excel identifies business objects in an OpenAPI document primarily by examining the paths defined in the service metadata.

The add-in creates a business object for each collection path defined in the OpenAPI document. A valid collection path:

- Does not end with a path parameter replacement token like `{department_id}`
- Defines `GET` and/or `POST` operations

Examples of valid collection paths are:

- `/Departments`
- `/Departments/{department_id}/child/Employees`

The add-in associates the collection path with any related paths. For example, the add-in associates an item path with a collection path if the path value is identical with the addition of a path parameter replacement token at the end. This comparison is case sensitive. An item path value of `/Departments/{department_id}` would be correctly associated with a collection path value of `/Departments`. In contrast, `/Departments` and `/departments/{department_id}` would not be associated with each other.

A business object is then defined by its collection path, item path (if present), and any other associated paths (custom actions, for example).

> **Note:**
>
> Certain paths may be ignored by the add-in depending on the service type.

Relationships between business objects are also determined by comparing the path information. A business object is considered a descendant of another business object if the collection path value starts with the collection and/or item path of another business object. This comparison is case sensitive. A business object with collection path `/Departments/{department_id}/child/Employees` is a valid child of a business object with collection path `/Departments` and item path `/Departments/{department_id}`.

Required Fields

When a business object field is created from service metadata, the initial values of the **Required for update** and **Required for create** properties are set based on the following OpenAPI property:

- The **Required for update** value is determined by the request body schema of the PUT (or PATCH) operation for the item path, along with the OpenAPI Required array property.
- The **Required for create** value is determined by the request body schema of the POST operation for the collection path, along with the OpenAPI Required array property.

When these PUT/PATCH or POST operations are not available, the Required properties are set using the response body schema of the collection GET operation and the OpenAPI Required array property. If the OpenAPI Required array property is not present in a schema, the corresponding Required property defaults to `false`.

You can always edit the **Required for update** and **Required for create** values in the Business Object Field Editor. See Configure Business Object Fields.

REST Operations

Table and Form-over-Table capabilities are enabled for `GET`, `PUT`, `PATCH`, `POST`, and `DELETE` operations as follows:

- Download is enabled if there is a `GET` operation on the collection path. Note that you must first download rows before you can perform operations on the item path. Therefore, `GET` on the collection path is a prerequisite for operations on the item path such as `PUT`, `PATCH`, or `DELETE`.
- Existing row updates are enabled if the item path has either a `PUT` or `PATCH` operation.
- Create new rows is enabled if the collection path has a `POST` operation.

- Delete existing rows is enabled if the item path has a `DELETE` operation.

For more information about Download, see Download Data. For more information about Upload, see Upload Changes.

> **Note:**
>
> Even if the business object supports an operation, you can still choose to disable a layout's capability by deselecting it in the Layout Designer's Advanced tab.

REST Request Headers

If required, you can add HTTP header fields to REST requests used in an upload operation. You can do this with any REST framework unless noted otherwise.

You can only add request headers to `PUT`, `PATCH`, `POST`, and `DELETE` operations. Additional request headers are not supported at download (`GET` operation).

During an upload, the add-in prepares the REST request as usual with the standard HTTP header fields. If the corresponding operation has additional headers defined, these values are evaluated and the headers added.

Reserved headers are skipped with no error. See Notes on REST Request Headers.

The add-in does not validate the header names or values. Be sure to consult the documentation for your target service before attempting to use request headers. An incorrect header may result in "Bad Request" errors.

Configure a Request Header

You can add REST request headers using the **Operation Editor** available from a business object's **Business Object Editor**.

The value you enter should be a string. If you want to use a non-string value, such as a date value, consult the REST service documentation to determine how the header should be formatted.

Header values can also include an expression. If you use an expression, make sure to follow the Oracle Visual Builder Add-in for Excel expression rules, in particular the rules for escaping ({ }) and using literal values. Reserved words, other than `RowVariables`, are not supported. See About Expressions.

You can define headers for child business objects as well. But keep in mind that, if the child data is sent in the parent payload, any child headers are ignored since the add-in does not send requests at the descendant level. See Upload Parent and Child Changes in the Same Payload.

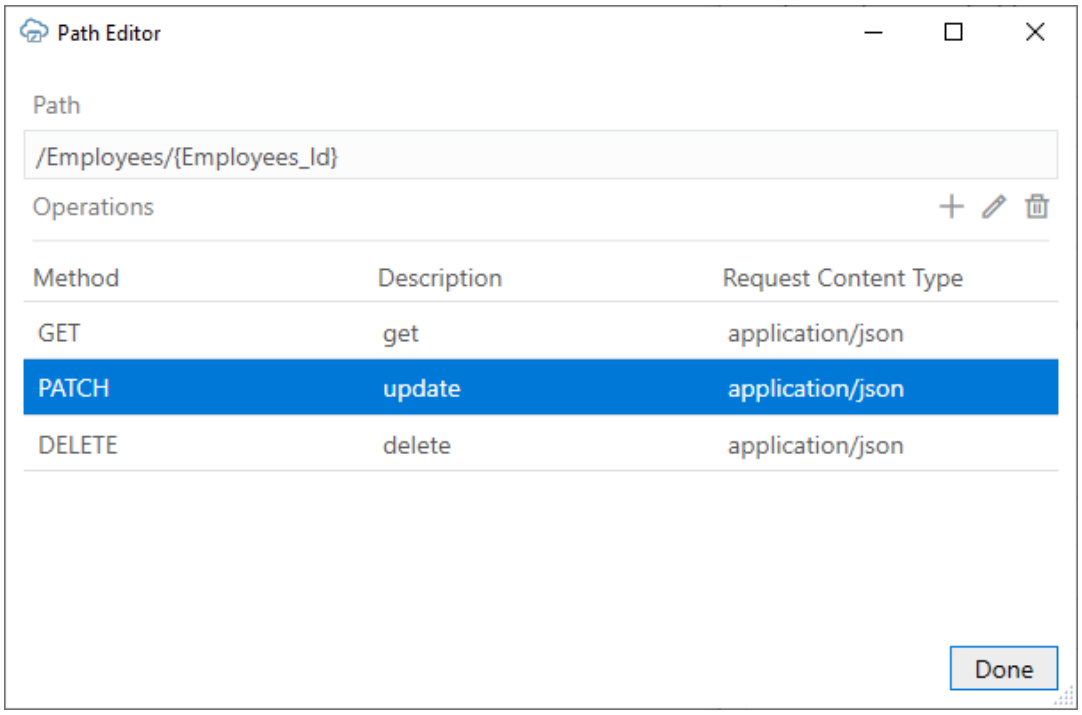
This feature supports the use of date effective objects in a workbook. Changes to date effective objects are uploaded to the service based on a range start date. To support this, you'll need to add a date field, called a "row variable", to your layout that your business users can use to enter the start date for a row. See Use Row Variables for a Business Object.

You can use Request Headers support to include the `Effective-Of` header, if desired. See EffectiveOf Headers in Multi-Row Requests.

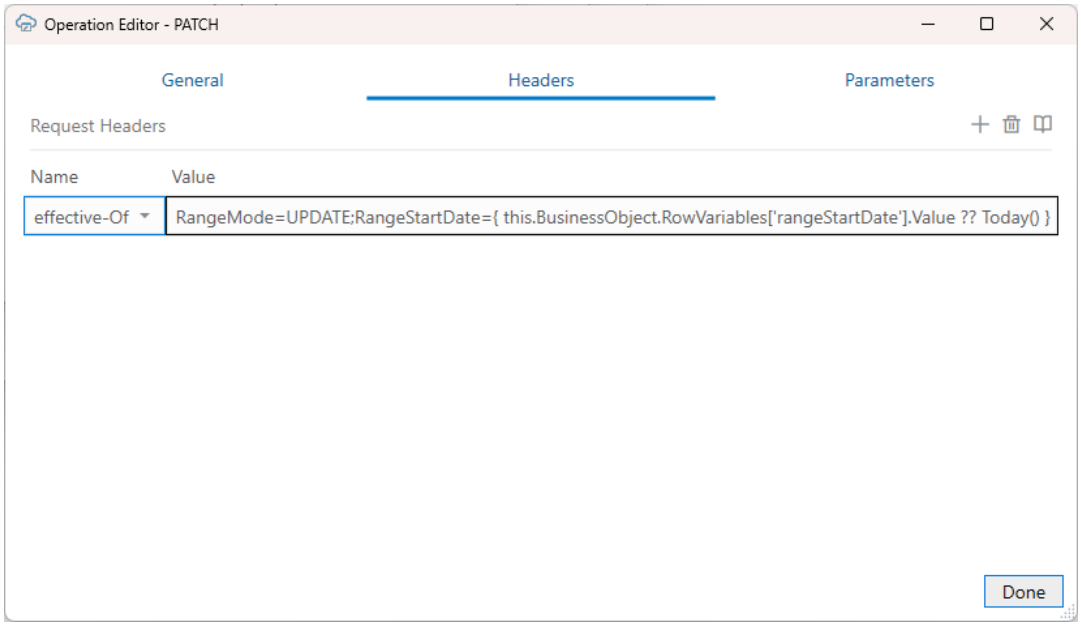
This task uses the example of an `Effective-Of` header on the `PATCH` operation that refers to a row's row variable, `rangeStartDate`.

To add an HTTP Request header to a business object:

1. Open the **Business Object Editor** for the business object. You can open the editor from the **Layout Designer** or by clicking **Manage Catalogs** from the Visual Builder ribbon.
2. From the **General** tab, click the **Edit** button next to the **Collection Path** or the **Item Path** to open the **Path Editor**. See REST Operations for more information on paths and operations.
3. From the **Path Editor**, select the desired operation and click the **Edit Operation** icon ( ).



4. From the **Headers** tab of the **Operation Editor**, click **Add a new header** ( ).
5. Select the desired header name (or type the name if it is not in the list); for example, `effective-Of`. Note that header names are case-insensitive.
6. Type the desired value; for example, `RangeMode=UPDATE;RangeStartDate={ this.BusinessObject.RowVariables['rangeStartDate'].Value ?? Today() }`.



In this example, the `RangeStartDate` attribute is set to the value of the row variable, `rangeStartDate`, in the selected layout. If the row variable is empty, today's date is used instead.

7. Click **Done**.

Once you add a header, the workbook can no longer be used with versions 3.6 or earlier.

Before you distribute the workbook, it is recommended that you test the defined headers by opening the Network Monitor and upload a change using the Upload Changes Data button from the Oracle Visual Builder ribbon. The Network Monitor shows the add-in's requests and the service's responses. See Network Monitor.

If you experience issues, try modifying the request header value and then clicking Upload Changes again.

Notes on REST Request Headers

Review this section for more information on reserved headers and limitations.

Reserved Headers

Some HTTP header fields are reserved by the add-in. If you use one of these headers for a REST request, the add-in will ignore it.

The reserved HTTP header fields are:

- `User-Agent`
- `Authorization`
- `Accept`
- `Content-Encoding`
- `Host`
- `Content-Length`
- `REST-Framework-Version`

- `Accept-language`
- `Accept-Encoding`
- `Content-Type`

Limitations

- Headers defined for GET operations are ignored. These include operations for download, lists of values (LOV), describe, and so on.
- Request headers are not supported for ADF REST multi-row requests except for the `Effective-Of` header. See EffectiveOf Headers in Multi-Row Requests.
- Headers are not supported for attachment business objects.
- Headers are not supported for custom actions.
- For ADF REST services, do not add the `Upsert-Mode` header on the `POST` operation for the collection path. Instead, enable Upsert Mode from the **Upload** tab of the **Business Object Editor**. See Upload Changes Using Upsert Mode.

Natural Language Support

For every request that the add-in makes to the REST service, Oracle Visual Builder Add-in for Excel automatically adds the `accept-language` header.

By default, the value sent with the `accept-language` header is the language/culture code that Excel is currently configured to use. You can change the language as described in Change the Add-in's Language.

Each REST service determines how and whether it will react to the `accept-language` header.

Object-typed Fields and Subfields

Oracle Visual Builder Add-in for Excel supports fields of type `Object`. These fields may expose **subfields**, also known as "nested" fields.

Consider, for example, an Employee business object with the following fields:

- First Name (type: String)
- Last Name (String)
- Address: (Object)
  - Street (String)
  - City (String)
  - State (String)
  - Zip (String)
  - GPS Coordinates (Object)
    - Latitude (Number)
    - Longitude (Number)
- Hire Date (Date)

In this example, the type of the Address field is `Object` and it contains subfields. Object fields should not be confused with arrays. In this example, an Employee has only one Address. The add-in does not support fields that are typed as arrays.

The add-in handles Object fields and their subfields in the following manner:

1. First, in the Business Object editor **Fields** tab, only the top-level fields are listed. In this example, the top-level fields are: First Name, Last Name, Address, and Hire Date. To edit the properties for subfields of Address, edit Address and find the Subfields list on the Field Editor window. The direct subfields for Address are Street, City, State, Zip, and GPS Coordinates. Since GPS Coordinates is of type `Object`, its field editor will show its subfields (Latitude, Longitude).
2. Next, when creating a Table layout from a business object's fields, the add-in promotes the subfields and creates columns for each (leaf) subfield. This maintains a regular, rectangular structure for the table in the worksheet. So, the above example generates a table with these columns:
   - First Name
   - Last Name
   - Address / Street
   - Address / City
   - Address / State
   - Address / Zip
   - Address / GPS Coords / Latitude
   - Address / GPS Coords / Longitude
   - Hire Date

REST Service Support Limitations

Refer to these limitations when planning to integrate a workbook with a REST service using Oracle Visual Builder Add-in for Excel.

> **Caution:**
>
> Many different request and response schema types are possible and Oracle cannot list all that are compatible with the add-in. If a particular structure is not listed explicitly as supported, it may not work.

- The add-in supports REST response payloads up to 1 billion characters in size.
- Only REST services that return `application/json` media types as response payloads are supported. The add-in supports `application/octet-stream` for attachments. Other media types such as XML are not supported.
- Asymmetrical field lists. Since download, editing, and upload all occurs in the same Excel rectangular grid, the add-in counts on having a single set of field IDs (JSON member names) for both download and upload. If the REST service uses different field IDs for the same information when completing different operations, it cannot be used effectively with the add-in.
- Fields with forward slash ('/') in the member name:

  - OpenAPI documents contain schema properties that are represented in JSON as something like `"memberName" : { . . . properties describing the field ... }`
  - When creating the business object field from the JSON member, the add-in uses the member name as the field ID.
  - Field IDs that include the / character are incompatible with the add-in, so such members will not be represented as fields in the business object.
- URLs longer than 8000 bytes may fail due to limitations in various network devices between the add-in and the server.

If a REST service owner makes significant changes to the service after the workbook is configured to integrate with the service, the integration may not function as expected. In such cases, you can either re-import the service metadata and create a new layout, or refresh the business object catalog. If the change is minor, you can update the business object details to match the change in the service. See Manage Catalogs and Business Objects.

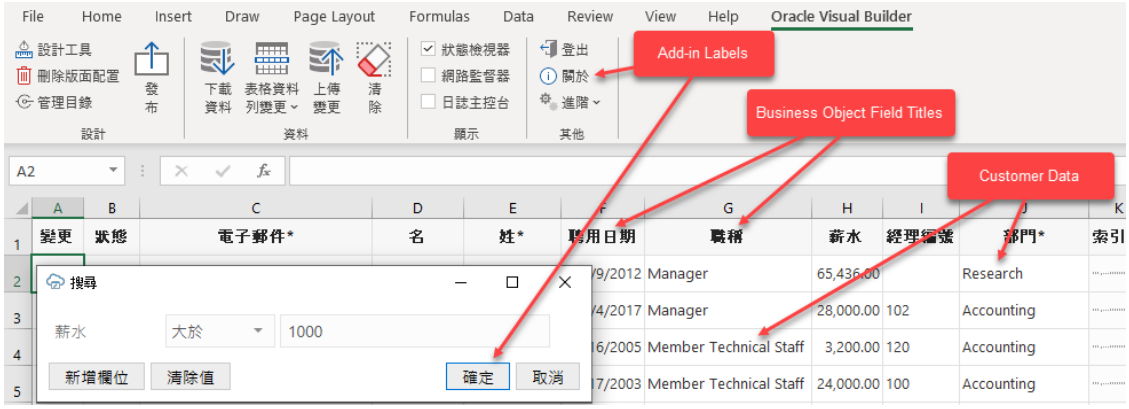
18
Internationalization

Oracle Visual Builder Add-in for Excel includes several features that support distributing your integrated workbook to other locales and in other languages.

These features include:

- A localized add-in
- Support for a user's preferred regional data formats
- Support for translating the integrated workbook using resource bundle files

If a business user opens an integrated workbook in the Chinese edition of Excel, the add-in displays all add-in labels such as icon labels, menu items, and buttons in the Chinese language.



If the workbook itself has been translated, the add-in can display Chinese text for strings such as business object field titles, the form label in a Form-over-Table layout, and workbook's help text.

Languages and Formats

Refer to this table for the settings used for Add-in, workbook, and REST languages and formats.

Area	Description	Controlled By	Notes
Add-in labels: language		Preferred Add-in Language	
Add-in windows: text orientation	Add-in windows include designer windows, task panes, and pop-ups displayed to business users such as Search. Text orientation is left-to-right or right-to-left.	Preferred Add-in Language	

Area	Description	Controlled By	Notes
Workbook labels: language	Workbook labels include column headers, help text, and so on. This does not include data from the services.	Preferred Add-in Language	If translations are available. See Manage Workbook Translations.
REST requests: accept-language header		Preferred Add-in Language	Results depend on the service.
Worksheet Cells: non-string formats		Cell styles and Windows Regional formats	See Appearance of an Integrated Excel Workbook.
Add-in windows: non-string formats	Add-in windows include designer windows, task panes, and pop-ups displayed to business users such as Search	Windows Regional formats	
Add-in installer: language		Windows display language	
Login web browser: language		Preferred Add-in Language	You may need to clear the browser cache when the preferred language is changed.

Refer to the following table for internationalization/localization terms and definitions.

Language or Format	Definition
Preferred Add-in Language	The preferred Add-in Language defaults to Excel's display language. Users may change the preferred language. See Change the Add-in's Language.
Windows Regional formats	Defined by Microsoft Windows system settings. Refer to your Microsoft documentation for the steps to adjust these settings.
Windows display language	Windows display language is set from the Language page. Open **Settings** and search for "Language".

Add-in Localization

The add-in is available in over 30 languages. By default, it automatically detects the user's preferred language from Microsoft Excel and uses that language where possible. When a business user opens an integrated workbook in Excel, everything in the add-in is displayed in the desired language—from the Oracle Visual Builder ribbon's icon labels and menu commands to add-in windows such as the Status Viewer and the Network Monitor.

The date, date-time, and number formats used by the add-in are also culture-sensitive. These data format types depend on the business user's preferences as defined in the Windows region settings. So a US user will see US dates and a French user will see French dates. See Appearance of an Integrated Excel Workbook.

REST Service Localization

The add-in also supports retrieving localized text from your REST service if the service supports it.

Every request the add-in makes to the service includes the `accept-language` header. The language setting specified for Excel is used for requests, including the describe requests that fetch the initial business object field titles. See Natural Language Support.

Workbook Translations

The add-in also supports localizing text strings in the integrated workbook, such as business object field titles, variable names, and help text for fields and finders. See Manage Workbook Translations.

Date-Time Value Handling

The add-in considers all date-time values in Excel to be in the local (machine) time zone. Date-time values are converted to UTC and sent in canonical ISO 8601 format to the REST service.

Date-time values retrieved from the REST service are expected to be in UTC with canonical ISO 8601 format. The add-in converts them to the local time zone before inserting them into the spreadsheet.

Manage Workbook Translations

If you plan to distribute your integrated workbook in another language, you can extract the base language file from the workbook and send it for translation. This translation file includes workbook-specific text strings such as field titles and help text for you to translate into your target language.
Once translated, simply import translation files for each required language through the add-in. When you distribute your localized workbook to your business users, the add-in displays the translated strings if the preferred language is available.

Translate Your Integrated Workbook

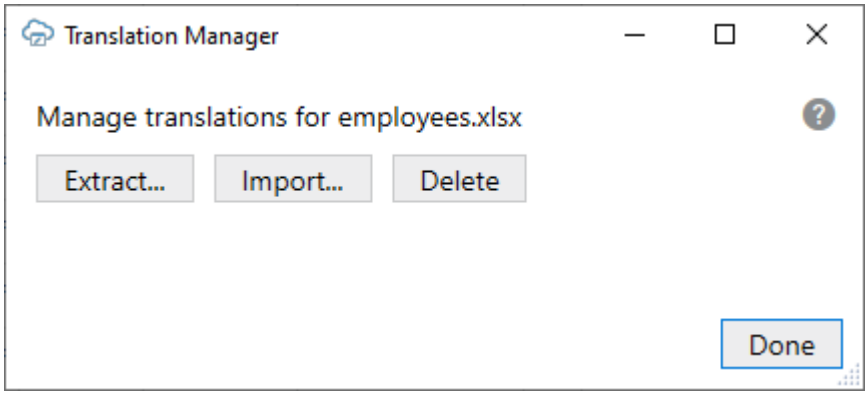
To translate your integrated workbook into another language, extract the workbook's translation file using Oracle Visual Builder Add-in for Excel and send it for translation. When you get the translated files back, import them into your workbook.

The extracted translation file only includes text strings that are currently used in your workbook's layouts. Field titles, variable names, and help text for fields and finders that are not in use are not included in the translation file. If you change the configuration of the workbook, extract the translation file again to pick up any new strings.

> **Note:**
>
> Fields included dynamically from polymorphic business objects are not included in the translation file.

To translate your workbook:

1. Open the workbook you want to translate.
2. Click the **Oracle Visual Builder** tab from the Excel ribbon.
3. Choose **Manage Translations** from the **Advanced** menu to open the Translation Manager.



4. From the Translation Manager, click **Extract** and save the translation JSON file to your local drive.
5. Submit the file to your translation team for translation.

   When creating a translation file for a specific locale, your translators will translate the strings and set the locale for the file before sending it back.
6. When you get the translated files back, open the Translation Manager and click **Import**.
7. From the **Import Translations** dialog, navigate to the location of the translated file or files, then select them.

   You can use the Shift and Ctrl keys to select multiple files for import.
8. Click **Open** to import the files.
9. Save and publish the workbook before distributing it to your business users.

About Translation Files

A translation file is a JSON file in Application Resource Bundle (ARB) format that stores the workbook's translatable strings for a given language. Each translatable string is stored in a key-value pair and includes additional attributes that help you and your translators understand the context.

Here's an example of an entry for a Business Object field, `"First Name"`, in an Employees workbook:

```
    "50689d80-c95c-4353-9032-cf4251d4abec.Title": "First Name",
    "@50689d80-c95c-4353-9032-cf4251d4abec.Title": {
        "context": "layout: TBL731992735, business object: Employees, field:
firstName",
        "description": "The field title used as a column header or a form
label.",
        "source_text": "First Name"
    },
```

The first line contains the translation key (`50689d80-c95c-4353-9032-cf4251d4abec.Title`) and translatable string value (`First Name`) separated by a colon. The key is a unique ID for the string in the workbook. The value is the string that your translator will translate into the target language, for example, "Prénom" for French.

Keep in mind that translation keys are different for each workbook. Two workbooks may have the same translatable string—say, “First Name”—but these strings will have different keys. For this reason, you can’t share translation files between workbooks.

Each key-value pair also includes some additional attributes that help translators understand how to translate the text string: `context`, `description`, and `source_text`. The context and description provide information about where in the workbook the string is used.

The source text is the original value for the string in the base language. In the base language file, the source text is always the same as the value. In the translated files, the value is translated but the source value remains unchanged. The source text value allows someone to inspect the translations and compare the strings easily.

The translation file also includes some global attributes (prefixed with `@@`) that apply to the translation file as a whole. Here is an example of the global attributes (`locale`, `context`, and `last_modified`) for an Employees workbook:

```
{
    "@@locale": "en-US",
    "@@context": "Integrated Excel Workbook: employees.xlsx",
    "@@last_modified": "2022-12-07T15:19:16.8664548-05:00",
```

The locale attribute provides the language code for the text strings stored in this file; in this case U.S. English (`en-US`). This is the value that you’ll need to change to indicate the new language when you translate the file. For Brazilian Portuguese, you’d use the value “pt-BR”. The locale attribute requires an IETF BCP 47 language tag.

On import, the add-in relies on the value of `@@locale` to identify the language for the translation file. The file name is not used for this purpose.

Change the Add-in's Language

You can change the language that the Excel add-in uses. Do this if you want to evaluate your integrated workbook with different languages.

To change the add-in language:

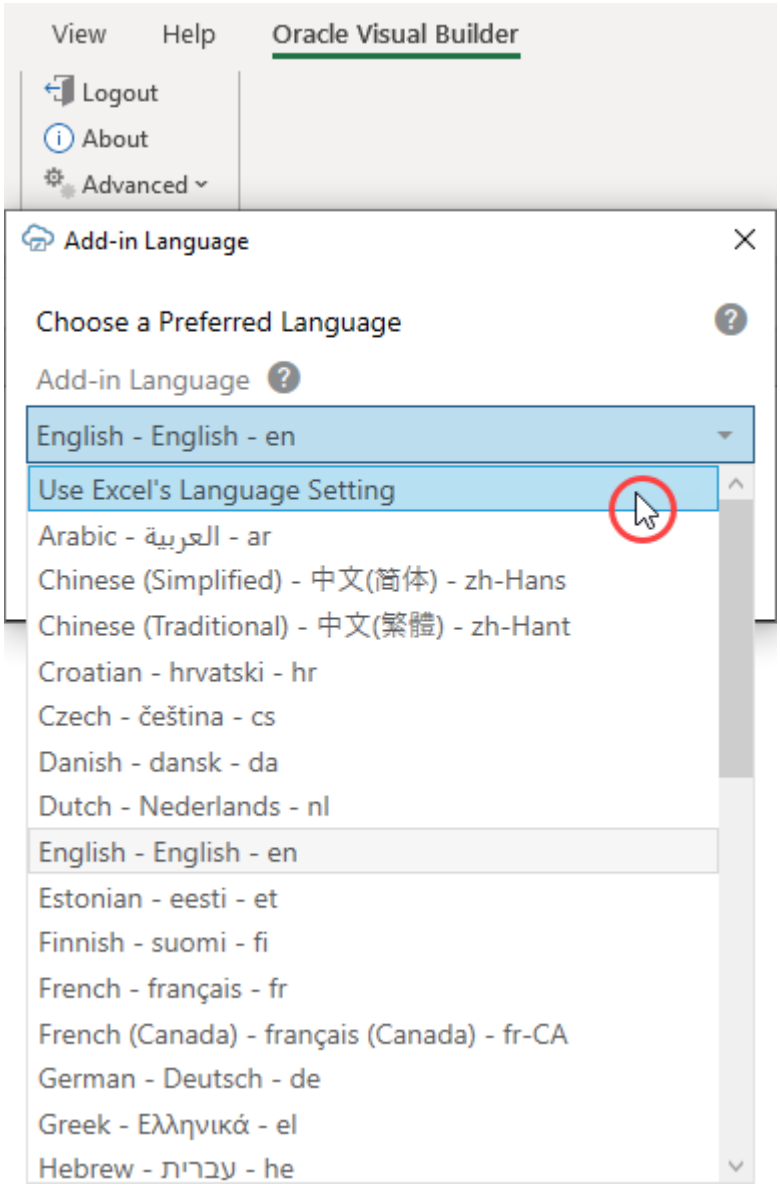
1. In Excel, click the **Oracle Visual Builder** tab.
2. Choose **Select Language** from the **Advanced** menu.
3. From the **Add-in Language** list that appears, select the language you want to use. The list displays the languages that the add-in supports.
4. Click **OK**.
5. Clear the embedded browser cache. See Clear the Embedded Browser Cache.
6. Restart Excel to make your changes take effect.

The add-in's user interface elements (**Download Data** and so on) now use the language you selected. If the workbook has been localized for the preferred language, the add-in uses that localization as well.

If the preferred language uses a right-to-left writing system, the add-in's windows appear in right-to-left mode. The language that Excel uses remains unchanged, as does the format used for dates, times, and numbers. See Excel or Windows options to change Excel’s language and formats for dates, times, and numbers. See also Natural Language Support.

During development, the add-in displays localized text for read-only property values in the design editors and dialogs. If a property is editable, the add-in displays the default language (or "base" language) value instead. When working in a language other than the base language, you may see localized values in some places and base values in others. When editing a property in the designers, you edit the base value. There is no change to the translated value already imported regardless of the current preferred culture.

The language that you choose for the add-in language is stored in a local file in the Windows user profile. You can select the **Use Excel's Language Setting** option in the Add-in Language drop-down list to remove this setting for the current user.



Language Change Detection

If business users have a different add-in language setting than the one used for the integrated workbook, they are prompted to redraw all layouts when they open the workbook for the first time. If they choose to redraw the workbook, any data and changes to the layouts are discarded.

Clearing all layouts when the language changes is recommended since some data, such as lists of values, may be language-sensitive. Downloading in one language and uploading in a different language may not succeed.

If they choose to skip the redraw, they can manually redraw the workbook later using either the **Clear Layout** or **Download Data** icons from the **Oracle Visual Builder** tab.

Set the Worksheet Title

If you plan to translate your integrated workbook, you can set the title of a worksheet in the Layout Designer so that the title is translated.

When you create a layout, Oracle Visual Builder Add-in for Excel uses the business object name (or parent-child names for a Form-over-Table layout) as the worksheet tab name. It does not, however, set the Worksheet Title property in the General tab of the Layout Designer.

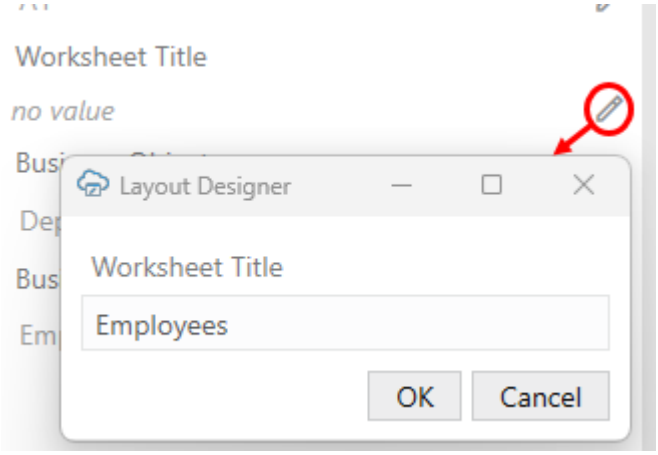
You'll need to set the worksheet title in the Layout Designer to ensure the worksheet title key and value are included in the translation file.

If you decide to set a value for a worksheet title, keep in mind that Microsoft Excel imposes a number of limitations on worksheet tab names:

- The length of worksheet tab name must not exceed 31 characters
- It must be unique within the workbook
- It cannot contain of the following characters: *[ ] \ / ?
- It cannot start or end with an apostrophe
- It cannot match any of the undocumented reserved words defined by Excel (example: "history")
- It cannot be blank/empty

To set the worksheet title:

1. Select the worksheet you want to update.
2. From the Layout Designer's General tab, click the Edit icon (✎) next to Worksheet Title and type in a title for the worksheet.



The add-in resets the worksheet tab name to the value you provided.

> **Note:**
>
> If the add-in fails to update the worksheet tab name, try a different value.
>
> You can also check the add-in logs for details on the failure. See Logging.

You can now translate your integrated workbook as described in Translate Your Integrated Workbook.

Once translated, the add-in updates the worksheet tab name to the translated value when the add-in detects a language change.

> **Note:**
>
> If you or a business user modifies the worksheet tab name, the workbook displays this value in the tab unless and until either the Worksheet Title property value or the language setting changes.

Internationalization Notes

- Some login pages used by some services may include a language selector. The language selector may have some influence over the login page for that service. However, it has no influence over the language choices for the add-in in general.
- If you change Windows Regional formats, you must restart Excel for the add-in to use the new settings.

19

Security

When using Oracle Visual Builder Add-in for Excel, refer to this topic for security information including security-related best practices and recommendations.

Security Guidelines

Follow these best practices:

- Update the add-in to the latest version available.
- Restrict access to Excel documents containing sensitive data.
- Consider adding passwords to workbooks to further reduce exposure.
- Always use HTTPS endpoints instead of HTTP.
- Do not use basic authentication.
- Ensure that the latest Windows updates and security patches have been applied to the computers where you install the add-in.
- Disable deprecated transport layer protocols, such as SSL, TLS 1.0, and TLS 1.1. Refer to KB5017811—Manage Transport Layer Security (TLS) 1.0 and 1.1 after default behavior change on September 20, 2022 on the Microsoft Support site.
- Consider using Excel's Inspect Workbook feature (available on Excel's File menu) to review and remove personal information from the workbook before you distribute it. When you use the Document Inspector, make sure the Hidden Worksheets check box is not selected. You must not remove hidden worksheets, because the add-in uses hidden worksheets to integrate a workbook with the REST service.

Microsoft Components

Oracle Visual Builder Add-in for Excel relies on a number of Microsoft technologies. These Microsoft technologies are subject to Microsoft's privacy policies and other Microsoft terms.

By installing and using this add-in, you are agreeing to those policies and terms and this add-in's direct or indirect usage of these technologies. See the Microsoft Privacy Statement.

See also Software Dependencies.

Authentication Options

At log in, the add-in uses the catalog's authentication setting to determine how to log in.

The add-in supports five authentication options:

- **Default:** At login, the add-in pings an Oracle Cloud Application anti-CSRF servlet endpoint. If the ping succeeds, Oracle Fusion Applications Token Relay is used. If the ping fails, Basic authentication is used instead.
- **Basic Access Authentication:** See Basic Authentication.
- **Oracle Fusion Applications Token Relay:** See Oracle Fusion Applications Token Relay Authentication.
- **OAuth 2.0 Authorization Code (PKCE):** See OAuth 2.0 Authorization Code Flow with PKCE.
- **No Authentication:** There is no prompt for credentials. No authentication-related headers are added to requests.

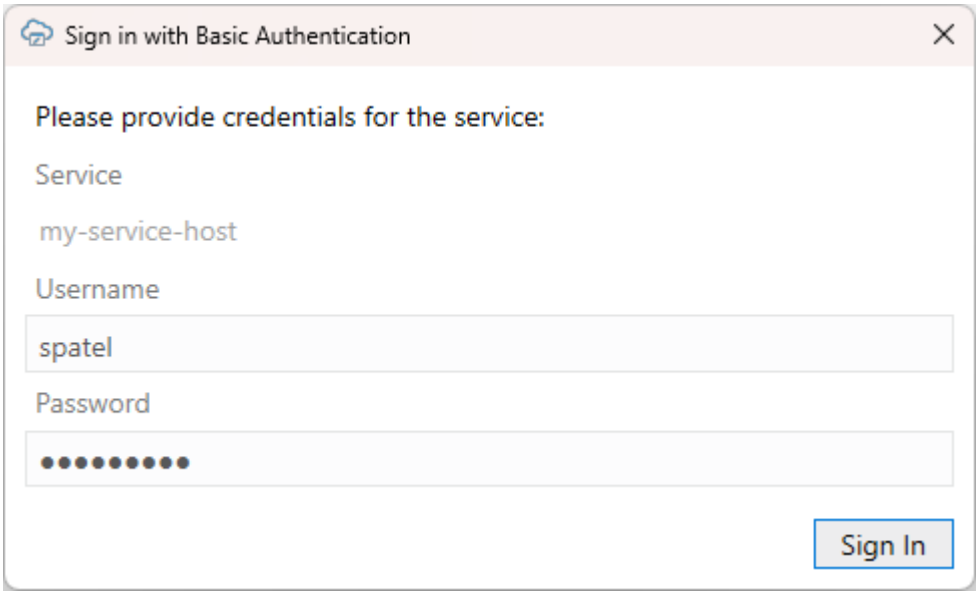
You can choose an authentication method when creating a new catalog. You can also change it later.

For information about how to configure your authentication settings, see Set an Authentication Method for a REST Service.

Basic Authentication

Oracle Visual Builder Add-in for Excel supports basic authentication.

- When a catalog is configured to use Basic Access Authentication, the business user is prompted for basic credentials before the first request is sent to that catalog's endpoints.



  See Authentication Options.
- The add-in sends the user credentials in the Authorization header for REST requests to the endpoint. See RFC 7617 for more information.

- When used with HTTP, basic authentication is not secure. Basic authentication should only be used with HTTPS, and preferably only in non-production environments.
- Valid credentials for a given service using Basic authentication may be different from valid credentials for Token Relay.

Oracle Fusion Applications Token Relay Authentication

Oracle Visual Builder Add-in for Excel supports authentication for REST services available through Oracle Cloud applications that use the Oracle Fusion Applications Token Relay servlet. Refer to this section for technical details on the Token Relay authentication mechanism.

Topics:

What Happens During the Login Sequence?

The add-in performs the authentication process just prior to any business operation initiated by the business user that requires access to the REST service. The add-in uses an embedded web browser control to host the login interaction sequence between the workbook user and the Fusion Application (FA) authentication provider.

During a successful login, the add-in captures the necessary authentication tokens and then uses them with subsequent REST requests. In particular, the access token is sent using the Bearer authentication scheme in the Authorization header. For more information, see RFC 6750.

The add-in uses these endpoints to get the authentication token:

Endpoint	Example
UI home page	`https://<my-service-host>/fscmUI/faces/FuseWelcome`
Anti-CSRF endpoint	`https://<my-service-host>/fscmRestApi/anticsrf`
Token relay servlet endpoint	`https://<my-service-host>/fscmRestApi/tokenrelay.`

> **Note:**
>
> In most cases, *<my-service-host>* is the catalog host. If a Token Relay Host is configured, it is used instead. See Configure Token Relay Authentication for a Catalog.

The login sequence:

1. If the catalog is configured to use the "Default" authentication method, the add-in makes an initial ping request to the anti-CSRF endpoint.

- If the request returns a `404` or other error, then the add-in abandons the Token Relay authentication mechanism and falls back to using Basic authentication.
- If the request returns a `200`, with content-type `application/json`, and the payload contains a non-empty `xsrftoken` member, then the add-in proceeds with the login sequence.

> **Note:**
>
> If the catalog is configured to use the "Oracle Fusion Application Token Relay" authentication method, the initial ping to the anti-CSRF endpoint is skipped. The ping occurs later during page navigation.

2. The add-in displays a modal pop-up login window that contains an embedded browser control, and directs the browser control to navigate to the UI Home Page URL. See The Embedded Browser.
3. Since the business user has not yet successfully logged in, the browser is redirected to a page with a login form. This form typically contains user name and password fields and a Submit button.
   The login user interface pages are controlled by the FA environment. There could be multiple steps involved, such as SSO, multi-factor authentication, and so on.
4. Each time a page navigation occurs in the embedded browser, the add-in performs an authentication test to see whether the user has logged in. See Token Relay Authentication Test.
   - If the user has successfully logged in, the login sequence is complete. The add-in automatically closes the login window and can continue with the REST request that triggered the login sequence using the harvested authentication tokens.
   - If the user has not yet logged in, the login window remains visible and the add-in continues listening for page navigation events.
   - If the user closes the login window before logging in successfully, the operation that initiated the authentication process is canceled.

Token Relay Authentication Test

The add-in watches for page transitions in the embedded browser and performs a Token Relay authentication test to see if the user has successfully logged in. If the add-in can get a token from the Token Relay servlet using cookies from the embedded browser, then the user has successfully authenticated.

Details about the test:

1. When each page navigation event is raised from the browser, the add-in sends a GET request to the anti-CSRF endpoint in order to get an anti-CSRF token.
   Once an anti-CSRF token is obtained, the add-in skips this step on subsequent page navigation events.
2. The add-in then makes a GET request to the Token Relay servlet endpoint. This request includes the ScopesFQ value, if configured. If the request is denied, the test fails. Otherwise, on a 200 OK with a response payload of content-type `application/json`:
   a. The payload is parsed and validated. The `access_token` member must be present and non-empty.

b. The value for `access_token` is captured in memory and used for subsequent REST requests.

c. If the parsing succeeds, the test is considered to have passed. The login window is closed, and the target REST request continues.

d. Otherwise, the test is considered to have failed and the login window remains open.

Configure Token Relay Authentication for a Catalog

When you configure token relay authentication, you're prompted to provide values for two optional configuration properties: Token Relay Host and ScopesFQ. If your REST service doesn't require values for these settings, you can leave them blank.

Provide a Token Relay Host value if the Fusion Applications token relay service is located on a different host than your REST service. If these services are both hosted on the same pod, you can leave this field blank and Oracle Visual Builder Add-in for Excel will use the catalog's host instead.

Provide a ScopesFQ value if you want to append a `scopesfq` query parameter value when calling the token relay service. If you leave this field blank, the add-in does not include the `scopesfq` query parameter.

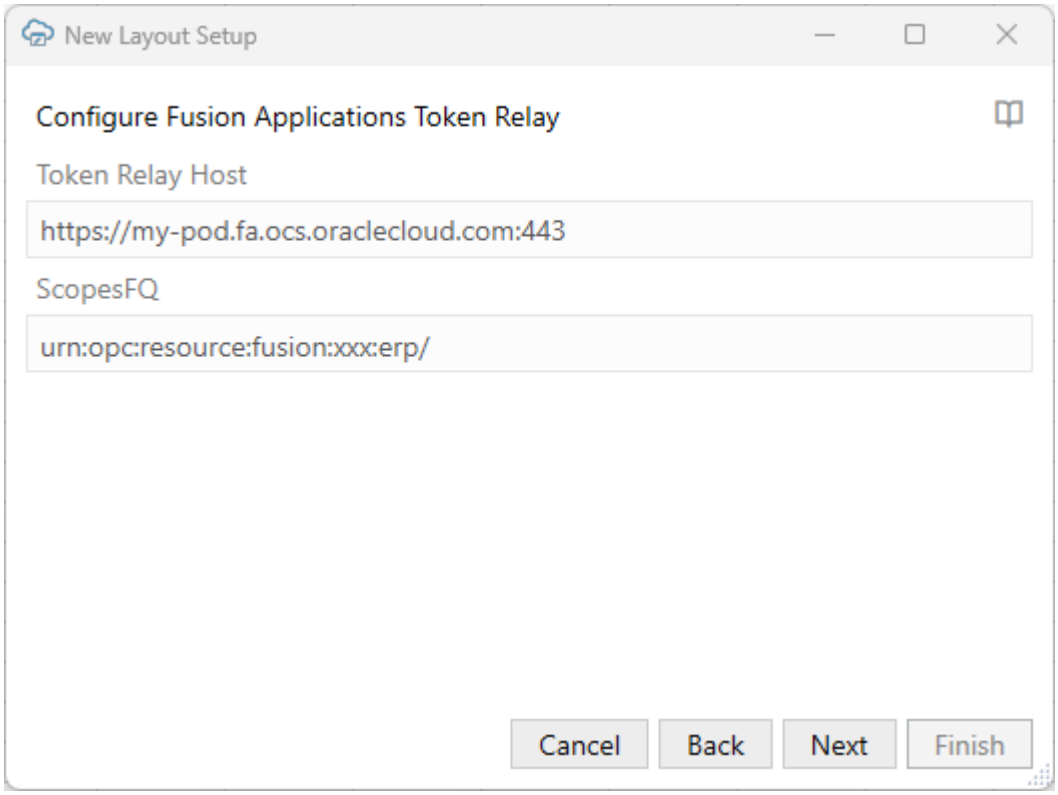
Consult with your REST service owner or refer to the documentation for your target REST service to determine the appropriate values for these properties.

These optional authentication properties can be configured when you create a catalog during layout creation. They can also be configured for an existing catalog from the Business Object Catalog Editor.

To configure token relay authentication during new layout creation:

1. Launch the New Layout Setup wizard as described in either Create a Table Layout in an Excel Workbook or Create a Form-over-Table Layout in an Excel Workbook.
2. From the first screen of the wizard, select **Oracle Fusion Applications Token Relay** from the Authentication list, then click **Next**.

   The wizard displays a screen for entering token relay properties. This image shows the screen with sample values in the screen's fields.



> **Note:**
>
> You can also access this screen for an existing catalog from the Advanced page of the Business Object Catalog Editor. To open this screen, select **Oracle Fusion Applications Token Relay** from the Authentication list, then click **Edit Authentication Flow Properties**. See Set an Authentication Method for a REST Service.

3. If the Fusion Applications token relay service is located on a different host than the REST service, type the token relay host in the **Token Relay Host** field.

   The host should use this format: `<protocol>://<host>:<port>`. For example:

   ```
   https://my-pod.fa.ocs.oraclecloud.com:443
   ```

> **Note:**
>
> Do not include the path to the Fusion Applications token relay service.

4. If you want to include a `scopesfq` query parameter, type it in the **ScopesFQ** field.

   If you include a value here, the add-in URL-encodes this value and adds it as the `scopesfq` query parameter value when calling the token relay service.

5. Click **Next** to proceed through the New Layout Setup wizard.

The sample values in the image results in this URL (with URL encoding omitted for readability):

```
https://my-pod.fa.ocs.oraclecloud.com:443/fscmRestApi/tokenrelay?
scopesfq=urn:opc:resource:fusion:xxx:erp/
```

Requirements for Token Relay Authentication

Before configuring Token Relay Authentication for an integrated workbook's catalog, review these requirements:

- The UI Home Page must be protected in such a way that an unauthenticated request in a browser redirects the browser and initiates a login sequence by redirecting to a login page.
- The `/anticsrf` endpoint should allow anonymous ("unauthenticated") access. The response payload must contain the token in the `xsrftoken` member.
- The `/tokenrelay` endpoint must be protected so that only authenticated users, identified by cookies issued during the browser login sequence, may access it.
- Oracle Fusion Applications Token Relay is only supported for Oracle Cloud Application deployments that include standardized `/anticsrf` and `/tokenrelay` endpoints with standardized payloads.

OAuth 2.0 Authorization Code Flow with PKCE

Oracle Visual Builder Add-in for Excel supports authentication for REST services using OAuth 2.0 Authorization Code flow with Proof Key for Code Exchange (PKCE). This authentication method allows clients like the add-in to authenticate and get an access token which can then be used to make REST requests to service endpoints.
This authentication method is required by some services such as NetSuite.

The easiest way to configure OAuth is to fill in and import a JSON configuration file with the required OAuth properties. See Configure OAuth 2.0 Authorization for a Catalog for the steps to configure OAuth for your workbook.

To configure your workbook to use OAuth 2.0 Authorization Code flow, you'll need to obtain some authentication details from your REST service owner. See OAuth 2.0 Authorization Properties.

OAuth 2.0 Authorization Properties

In order to configure your workbook to use OAuth 2.0 Authorization Code flow, you'll need to obtain a client identifier from the security administrator for the service you are using. You'll also need to provide other details such as required endpoint URLs. Consult with the REST service owner for help.

Here are the required OAuth properties:

- **Client Identifier:** The identifier set up for the add-in to use when executing the authorization flow. Obtain this value from the security administrator for your service.
- **Authorization Endpoint:** The authorization server endpoint used by the client to obtain authorization from the resource owner via user-agent redirection.
- **Redirection Endpoint:** The client endpoint used by the authorization server to return responses containing the authorization code to the client using the resource owner user-agent.
- **Access Token Scope:** The authorization and token endpoints allow the client to specify the scope of the access request using the "scope" request parameter.

- **Token Endpoint:** The authorization server endpoint used by the client to exchange an authorization grant for an access token, typically with client authentication.

> **Note:**
>
> The add-in does not require the Client Secret.

For more information on these properties, see OAuth 2.0 Authorization Framework.

OAuth 2.0 Authorization Code Flow Steps

The authorization flow follows these steps:

1. The add-in starts the login sequence by validating the OAuth2 configuration properties. If the properties are valid, the flow proceeds.
   If the properties are missing or otherwise invalid, then the login attempt is aborted and an error is reported. For example, an endpoint property that is not an absolute URL is invalid and results in an aborted login.
2. The add-in constructs a Uniform Resource Identifier (URI) using the Authorization Endpoint property, along with Client Id and other values saved in the **OAuth 2.0 Authorization Code (PKCE)** screen.
3. The add-in displays the login browser window and instructs the browser to navigate to that authorization Uri. See The Embedded Browser.
4. The add-in watches for page transitions and redirects in the browser. All other browser and user interactions are governed by the logic and configuration of the authorization server. There could be multiple pages and steps necessary for the user to provide credentials, get consent, and so on.
5. When the authorization server redirects the browser back to the Redirection Endpoint, the add-in closes the browser.
   If the Redirect indicates an error, the add-in reports it and aborts the login flow. On a successful redirect, the add-in performs some validation on the returned values. If that succeeds, the add-in proceeds with the flow.
6. After a successful redirect, the add-in harvests the authorization code and sends it, along with other key values, in a POST request to the Token Endpoint.
   The Token Endpoint returns an access token, which the add-in then includes in the Authorization header using the Bearer scheme, when making subsequent REST requests.

Refer to Authorization Code Grant for information on authorization code flow. See also Proof Key for Code Exchange by OAuth Public Clients for information on PKCE.

Configure OAuth 2.0 Authorization for a Catalog

You can configure your integrated workbook to authenticate with the REST service using OAuth 2.0 Authorization code flow. You can provide the required authentication properties when you create a catalog during the creation of a layout.

You can also configure an existing catalog to use this authentication method from the Business Object Catalog Editor.

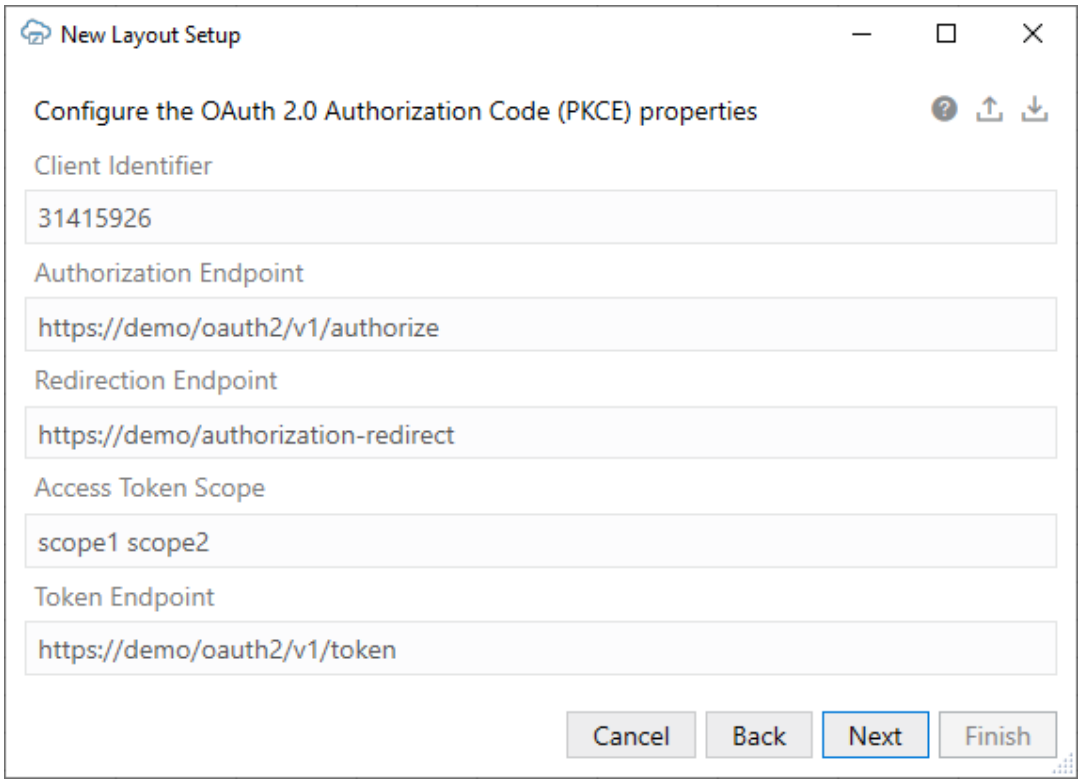
The easiest way to configure this authentication method is to export a blank JSON file from Oracle Visual Builder Add-in for Excel, fill in the required values, and import the completed configuration file.

Before you proceed, obtain the required properties from the REST service owner. See OAuth 2.0 Authorization Properties.

To import a configuration file during new layout creation:

1. Launch the New Layout Setup wizard as described in either Create a Table Layout in an Excel Workbook or Create a Form-over-Table Layout in an Excel Workbook.
2. From the first screen of the wizard, select "OAuth 2.0 Authorization Code (PKCE)" from the **Authentication** list, then click **Next**.

   The wizard displays a screen for entering OAuth 2.0 properties. This image shows the screen with sample values in the screen's fields.



> **Note:**
>
> You can also access this screen for an existing catalog from the **Advanced** page of the **Business Object Catalog Editor**. To open this screen, select "OAuth 2.0 Authorization Code (PKCE)" from the **Authentication** list, then click **Edit Authentication Flow Properties**. See Set an Authentication Method for a REST Service.

3. Click the **Export** icon ( ) to save a blank JSON file with the required properties to your local drive.
4. Provide this file to the service owner to fill in.

Here is an example of the JSON file showing sample values:

```
{
  "type": "oauth2",
  "authorizationCode":
  {
    "clientId": "31415926",
    "authorizationEndpoint": "https://demo/oauth2/v1/authorize",
    "redirectionEndpoint": "https://demo/authorization-redirect",
    "accessTokenScope": "scope1 scope2",
    "tokenEndpoint": "https://demo/oauth2/v1/token"
  }
}
```

5. When you get the JSON file back with the required values, click the **Import** icon () to import the file to the add-in.
6. Click **Next** to proceed through the New Layout Setup wizard.

OAuth Limitations and Known Issues

Before configuring OAuth 2.0 for an integrated workbook's catalog, review the limitations here:

- PKCE support is required. See RFC 7636.
- There is currently no support for token Refresh logic.
- For the first step in the OAuth2 flow, the `code_challenge` is set using `code_challenge_method=S256`. "Plain" is not supported.

Service Authorization and User Privileges

In addition to authentication ("who am I"), REST services often enforce authorization ("what am I allowed to do") on every service request (GET, POST, PATCH, and so on). REST services may require business users to have specific privileges in order for them to view and update data.
Oracle Visual Builder Add-in for Excel does not have access to the user identity or any kind of privilege/role/authorization information and can't check authorization prior to an operation. Instead, it is the service that enforces authorization and returns an error if the current user is not authorized to perform the requested operation.

If the workbook is configured to allow an operation the business user is not authorized for, the user may see an error such as an `403 Forbidden` error. For these cases, the business user should follow up with the REST service owner or system administrator in order to be granted the appropriate privileges.

Transport Layer Security

When the add-in connects to a REST endpoint using HTTPS, the add-in relies on the system default behavior for Transport Layer Security (TLS) to determine which TLS protocol is to be used.

Because the add-in runs within the Excel process, it cannot rely entirely on the .NET Framework 4.8 default setting to do this. To ensure that the system default behavior is in effect,

the add-in sets the *AppContext.DontEnableSystemDefaultTlsVersions* property to false for the current app domain.

See the following Microsoft documentation:

- If your app targets .NET Framework 4.7 or later versions
- Configuring security via AppContext switches (for .NET Framework 4.6 or later versions)

The Digital Certificate

The artifacts that make up the Oracle Visual Builder Add-in for Excel are signed with a digital certificate. The digital signature proves the authenticity of these artifacts and verifies the identity of the publisher, Oracle. Digital signatures are created using certificates issued from trusted certificate authorities.

Certificates are used to sign artifacts during the product build process. All "sign-able" artifacts are signed starting with the installer (MSI) file and including all the DLLs that make up the add-in.

> **Note:**
>
> This topic provides the procedures in Windows Explorer to view and install the certificate as well as copy the certificate's public key. Be aware that the steps may be different for different editions and versions of Windows. Check the documentation for your version of Windows for more information.

Can I inspect the certificate?

You can inspect these certificates before and after installation to verify the authenticity of the add-in's artifacts.

To do so, navigate to the installer file (`vbafe-installer-all-users.msi` for the all-users installer), open the Properties window, then select the Digital Signatures tab.

> **Caution:**
>
> If the Digital Signatures tab is missing on the installer, discard the file. It may not be authentic.

Expired Signatures

An expired certificate doesn't mean that the signature is invalid. A properly timestamped signature remains valid well after the "valid from/to" date range shown in the certificate.

To get the latest certificate, upgrade to the latest available version of the add-in.

Trusted Publishers

Microsoft Excel offers an optional trust center setting called **Require Application Add-ins to be signed by Trusted Publisher**.

To use this feature, install the certificate:

1. From the Digital Signatures tab, select the signature from the Signature list and click **Details**.
2. From the Digital Signature Details dialog, select the General tab, then click **View Certificate**.
3. From the General tab of the Certificate dialog, click **Install Certificate...**.
4. From the Certificate Import Wizard, choose either **Local Machine** for the all users installer or **Current User** for the current user installer.
5. Click **Next**.
6. Select **Place all certificates in the following store** and then click **Browse**.
7. From the Select Certificate Store dialog, select "Trusted Publisher" and then click **OK**.
8. Click **Next**, then **Finish** to close the wizard.

The certificate now appears in Excel's Trust Center.

Please consult Microsoft documentation for more information.

The Public Key

To get a copy of the public key associated with the add-in's digital certificate:

1. From the Digital Signatures tab, select the signature from the Signature list and click **Details**.
2. From the Digital Signature Details dialog, select the General tab, then click **View Certificate**.
3. From the Certificate dialog, select the Details tab, then select Public Key from the list.
4. Click **Copy to file...**.
5. Follow the instructions in the Certificate Export Wizard.

The Add-in's Certificate Update Cycle

Oracle acquires a new digital certificate approximately every two years. Once available, subsequent releases of the add-in are signed with the new certificate.

If you have installed the certificate and public key previously, you may need to repeat that process after you upgrade to a new version of the add-in signed with a new certificate.

20

Troubleshoot Excel Workbooks

If you experience issues with Oracle Visual Builder Add-in for Excel, follow the steps here to identify and resolve issues. If you still can't resolve your issue, contact Oracle Support.

- If you are having issues installing the add-in, see Troubleshooting the Installation.
- Review the documentation to make sure the desired operation is supported.
- Download and run the Client Health Check Tool.
- Make sure you're on a supported platform.
- Upgrade to the latest version of the add-in.
- Apply available Microsoft updates.
- Close all workbooks, exit Excel, and try again with simple steps.
- Generate an add-in log for review. See Logging. If you contact Oracle Support, you may be asked to provide this log.
- Generate a diagnostic report if required. See Diagnostic Report.
- If you are having an issue with a login page, try clearing the browser cache. See Clear the Embedded Browser Cache.
- If some **Oracle Visual Builder** ribbon commands are disabled after you open a workbook, check the Workbook Info window for details on the issue. See Resolve Workbook Issues.

> **Tip:**
>
> You can copy the content of some of the add-in's windows and task panes to the clipboard by right-clicking them and selecting the option from the context menu.

> **Note:**
>
> If you are experience network issues such as "bad request" errors during normal operations, there may be an issue with the REST service. Contact the owners of the REST service to determine whether the service is providing the expected response. You can use the Network Monitor to capture details of the request-response pairs to share with the REST service owners. See Network Monitor for the steps.

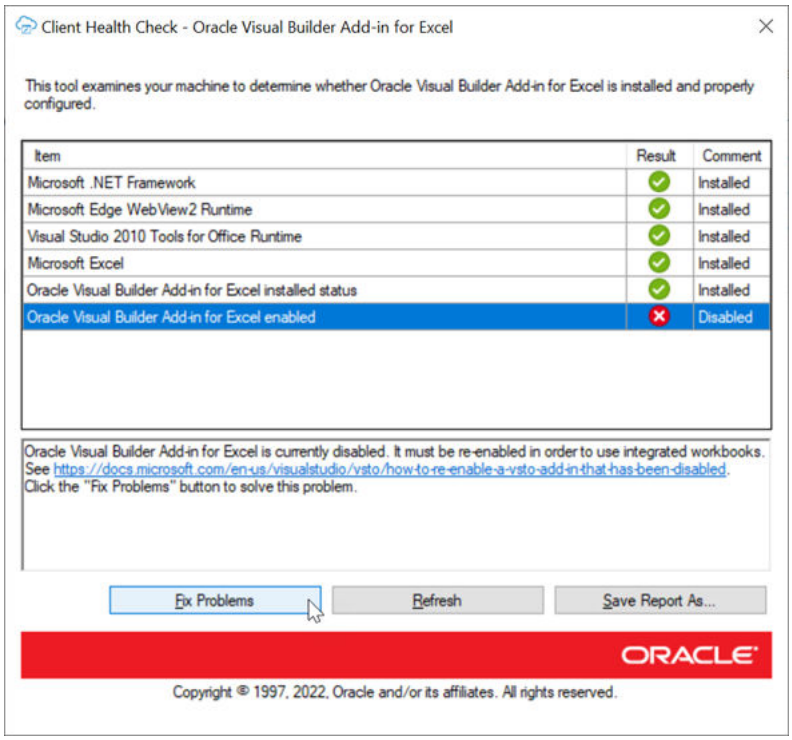
For more information about Excel, see Excel specifications and limits.

Check Your Environment

Run the Client Health Check Tool to check if the desktop configuration and environment are suitable for Oracle Visual Builder Add-in for Excel and to resolve issues.

Download the latest version of the Client Health Check tool (`vbafe-health-check.exe`) from the Oracle Downloads page.

1. Run `vbafe-health-check.exe` and review the result for each examined item.



Select each failed item (⊗) for more details, including the steps necessary to resolve the issue. If the add-in can resolve the issue, the **Fix Problems** button is enabled.

> **Note:**
>
> The Client Health Check tool may also show warnings (⚠) if it finds something that is not optimal. You don't have to resolve warnings in order to use the add-in but it is recommended. Select each item with a warning to display information on how to resolve the warning.

2. Click **Fix Problems** if available or follow the instructions to resolve the issue.
3. If Oracle Support requests a copy of the report, click **Save Report As...** and choose a name and location for the report.
4. Send the report to Oracle Support.

> **Caution:**
>
> This report may include personal information from your computer including, but not limited to, the computer's name and the end user's Windows profile name. Be sure to select the appropriate option when uploading files with personal information to a service request so that the file access can be restricted as needed.

Apply Microsoft Updates

When troubleshooting issues with Oracle Visual Builder Add-in for Excel, we recommend first applying all pending updates for Windows and Excel before reproducing the issue. A Microsoft patch may resolve your problem.

1. From the Windows Start menu, select **Settings**, **Update & Security**, and then **Windows Update**.
2. If updates are available on the Windows Update page, review the updates and click **Install Now**.

> **Note:**
>
> The details of applying Windows updates can vary from version to version and also according to your company's IT policy. Check with your system administrator for assistance, if needed.

Network Monitor

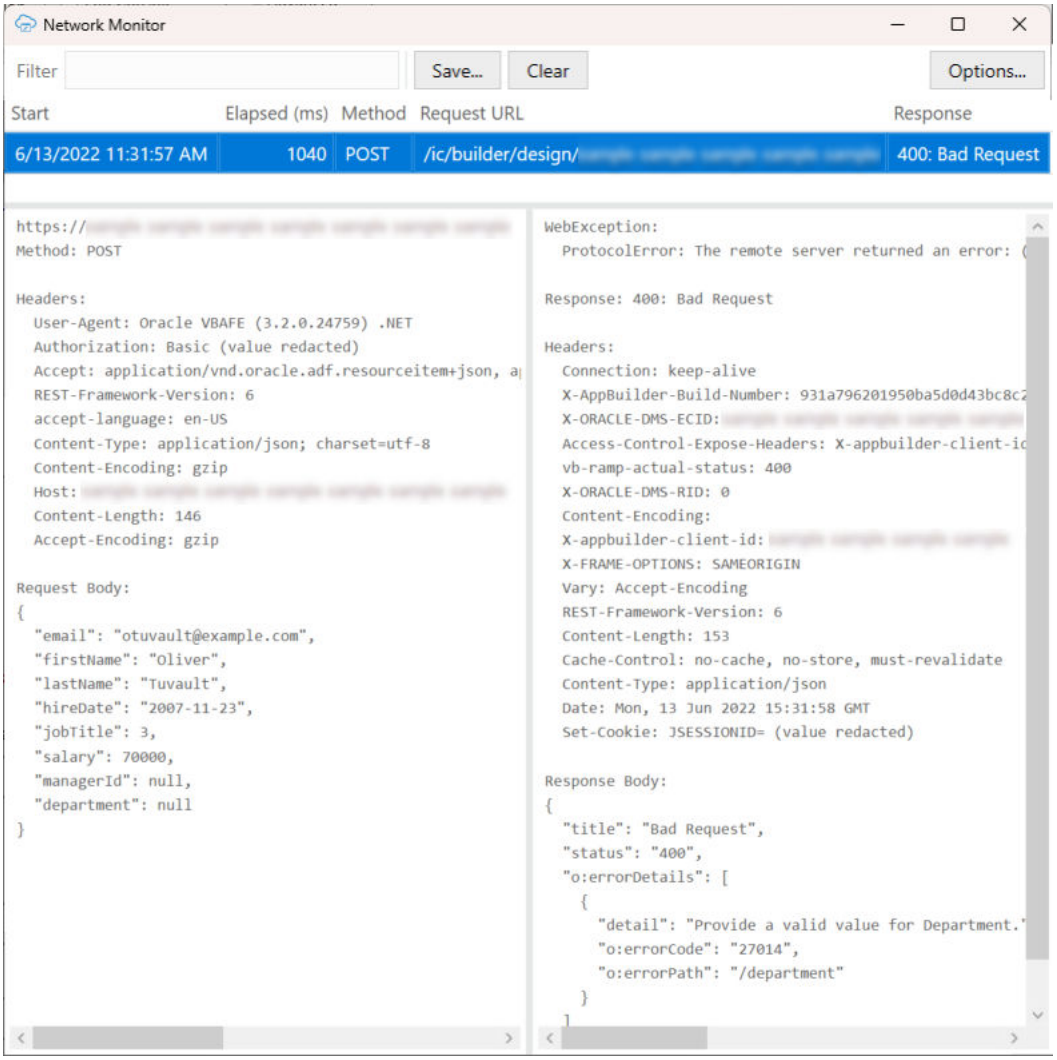
Use the Network Monitor window to inspect the content of REST service calls between your Excel workbook and the REST service if you encounter unexpected behavior.

The Network Monitor window provides information such as the start time, the elapsed time, and response for each REST call that originates from the workbook. In addition, it provides the headers and payloads of each request sent from Oracle Visual Builder Add-in for Excel and the corresponding response from the service. The window shows up to 100 request-response events by default. Older events are discarded as new ones are added.

If you encounter issues with the REST service such as "bad request" errors, you can capture information about the error in the Network Monitor window and share this information with the owner of the REST service.

The Network Monitor window generally goes to the background while you perform the steps of your use case. Bring the window forward to see the details of each request and response.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Network Monitor** to open the Network Monitor window.
3. Repeat the steps that led to the issue.



4. Select each request and response line from the upper table to display more details on the request and response in the panes below.
5. If you need to review more than 100 request-response events, click **Options**, increase the maximum number of events, then repeat the steps.
6. To save the details of a request and response, select an entry for a REST service call in the upper table, click **Save**.

> **Caution:**
>
> Request and response payloads may include sensitive information, including actual data and personally identifiable information. Be sure to handle these payloads with due care.

Installation Logs

The Oracle Visual Builder Add-in for Excel installer is a Windows installer that produces standard Microsoft Windows installer logs. If you are having trouble installing Oracle Visual Builder Add-in for Excel, you can generate and view an installation log file.

To generate an installation log file, run the installer from the command line and include `/log <log file path>`. See Run the Installer from the Command Line.

Add-in log files may include some personal information from the user's computer including, but not limited to, the computer's name and the end user's Windows profile name. When uploading files with personal information to a service request, be sure to select the appropriate option so that the file access can be restricted as needed.

Logging

When reporting an issue about the add-in, generate a detailed log file that captures the steps that lead to the problem you want to report.

The log file that you generate captures information about steps during an Excel session.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Log Activity** from the Advanced menu to specify a directory location and file name for the log file. This starts the logging session.
3. Repeat the steps that lead to the issue.
4. Exit Excel completely to stop the logging session and before you access the log file.

> **Note:**
>
> The next time you run Excel logging will no longer be enabled.

> **Caution:**
>
> Log files may include personal information from the user's computer including, but not limited to, the computer's name and the end user's Windows profile name. Be sure to select the appropriate option when uploading files with personal information to a service request so that file access can be restricted as needed.

Log Console

The Log Console displays log messages based on the actions performed. If you encounter any issues, view the logging messages to troubleshoot and diagnose issues.

> **Note:**
>
> If you are trying to provide a log file to support, refer to Logging to generate a log file, rather than trying to copy log entries from the Log Console.

To review logged messages in the Log Console:

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Log Console**.
   The Logging Console window displays.

3. Repeat the steps that led to the issue.
4. Review the logged messages.
5. Do one or more of the following as required:
   - Type a word or phrase in the **Filter** box to display matching log entries.
   - To display more details such as Time, Thread, and Level, click **Show Event Details**.
   - For verbose logging, click **Enable Verbose Output** and repeat the steps that led to the issue.
   - Click **Clear** to discard all log entries.

Diagnostic Report

The diagnostic report contains information that can help resolve issues. Provide a diagnostic report when reporting a problem with Oracle Visual Builder Add-in for Excel.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Diagnostic Report** from the Advanced menu.
3. Save the diagnostic report to a directory location with a file name of your choice.

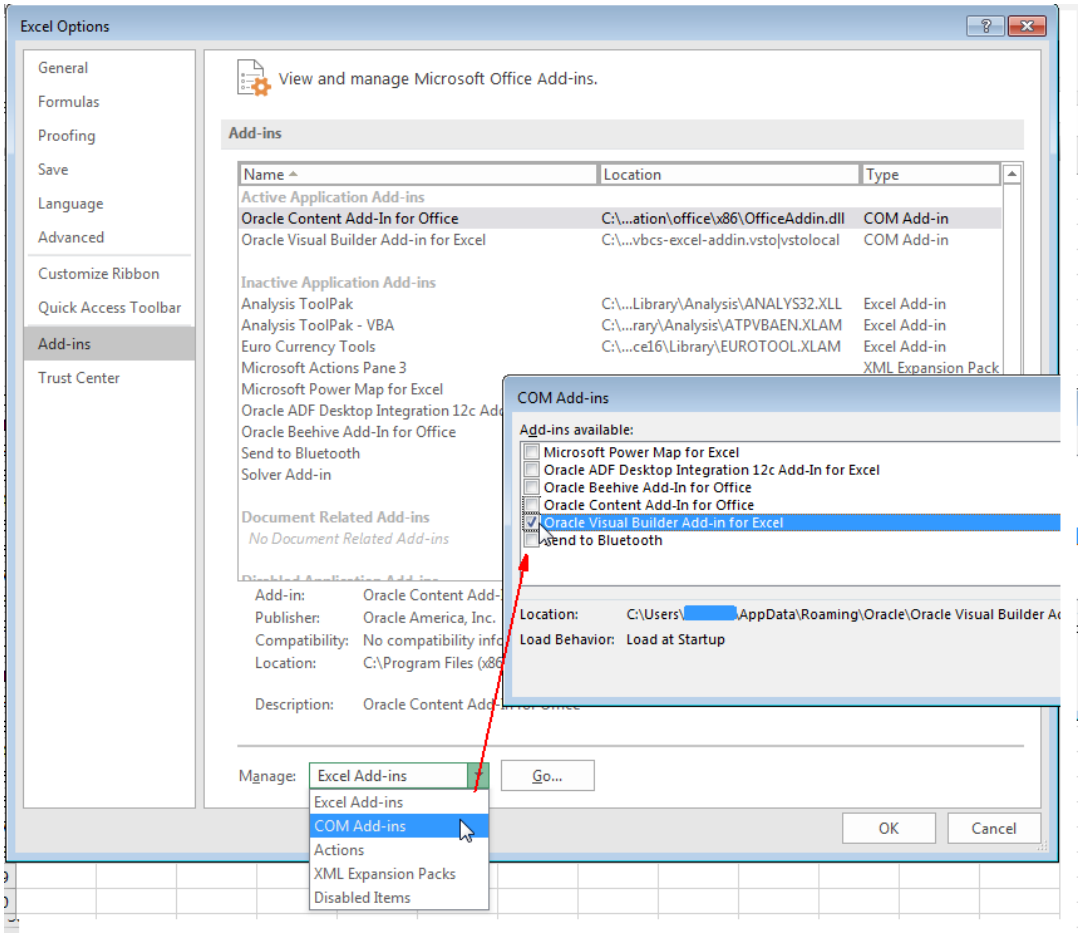
> **Note:**
>
> The report may include some personal information from the user's computer including, but not limited to, the computer's name and the end user's Windows profile name. Be sure to select the appropriate option when uploading files with personal information to a service request so that the file access can be restricted as needed.

Re-Enable Oracle Visual Builder Add-in for Excel

If your add-in becomes disabled and you are unable to use the client health check tool, you can re-enable Oracle Visual Builder Add-in for Excel through Microsoft Excel.

1. In Excel, click **File** > **Options** > **Add-Ins**.
2. Select **COM Add-ins** in the Manage drop-down list and click **Go**.
3. Deselect the **Oracle Visual Builder Add-in for Excel** check box and click **OK**.
4. Restart Excel.

5. Enable the add-in by repeating the steps and instead selecting the **Oracle Visual Builder Add-in for Excel** check box from the **Add-ins available** list in the COM Add-ins dialog.

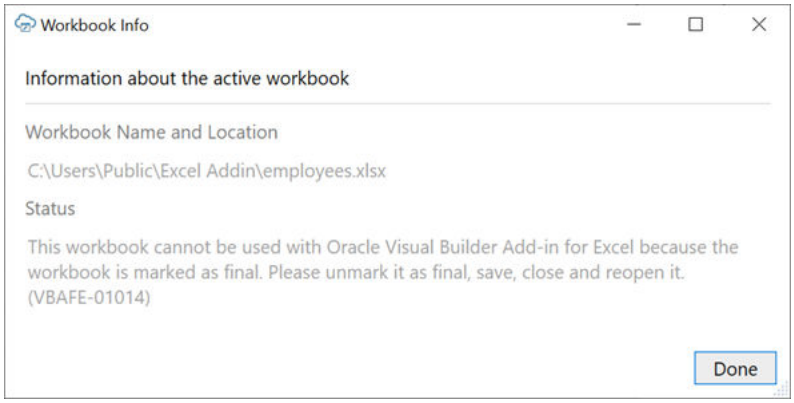


Resolve Workbook Issues

If you see an error message when you open a workbook or find that some **Oracle Visual Builder** ribbon commands are disabled, Oracle Visual Builder Add-in for Excel has detected an issue with your workbook. Use the Workbook Info window to troubleshoot these issues.

Your workbook may be unusable for a number of reasons such as if it is marked as final in Excel or has been saved to an incompatible file type.

To check the status of your workbook, open the Workbook Info viewer by choosing **Workbook Info** from the **Advanced** menu of the **Oracle Visual Builder** ribbon. This viewer shows information such as the name, location, and status of your workbook.



Check the status for the description of the issue and for any troubleshooting steps. Issue-free workbooks have a status of "Integrated". Workbooks that are not integrated with the add-in will show a status of "Not integrated".

In a scenario where a workbook is marked as "final", you'll need to clear the **Mark as Final** setting (under **File>Info>Protect Workbook**), then save and reopen the workbook.

> **Note:**
>
> Do not use Excel's **Edit Anyway** button in the yellow message bar to try to edit the workbook. This command will not re-enable the **Oracle Visual Builder** ribbon.

21

Migrating an Excel Workbook to Version 4.1

You can migrate an Excel workbook created or modified with version 4.0 or earlier of Oracle Visual Builder Add-in for Excel to use version 4.1.

This migration is seamless: No special steps are required, other than the usual upgrade recommendations (see Upgrade to the Latest Version).

In general, your workbook should continue to function after the upgrade as before. If you want to take advantage of new add-in features, you may need to make some changes to the workbook configuration.

> **Note:**
>
> Once you configure your workbook to use the latest features, it may no longer be compatible with the older version of the add-in. Before distributing your updated workbook, make sure your target audience has access to a compatible version of the add-in.

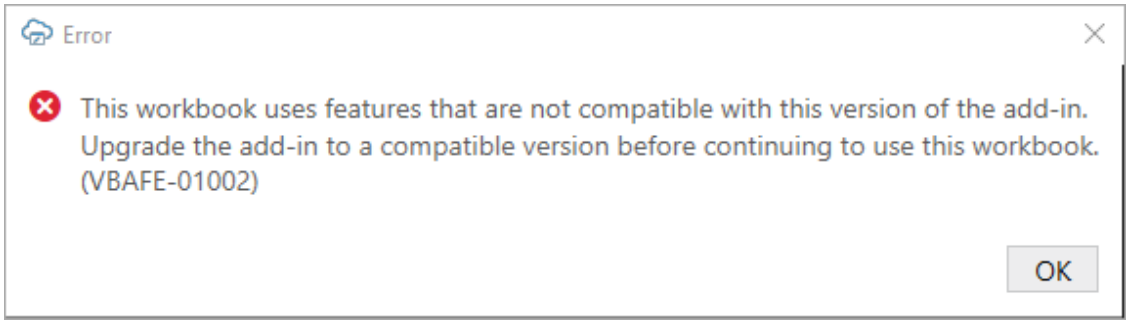
Backward Compatibility

Workbooks created with or modified by the latest version of Oracle Visual Builder Add-in for Excel may not be compatible with earlier versions of the add-in. Refer to this topic for compatibility restrictions.

A workbook created or modified in version 4.1 is:

- Compatible with versions 3.5 and later unless a particular incompatible feature was used in that workbook.
- Incompatible with versions 3.4 and earlier.

If a workbook uses a feature that is incompatible with the installed version, you'll see a message like this:



22

Use Expressions in an Integrated Workbook

You can use expressions in a number of places in your workbook configuration such as in search queries, REST request headers, lists of values, and so on.

A simple expression may compare a value in a field to a constant, like this: `{ this.Value <= 500 }`. This expression might be used in a custom validation rule to ensure an entered value doesn't exceed the given threshold. See Create Field Validation Rules.

Here are some other use cases for expressions:

- As a row finder variable's default value that references a workbook parameter. If a row finder is configured, then the user is presented with a Search prompt that displays the value of the referenced workbook parameter as the row finder variable value. See Use a Workbook Parameter Value for a Row Finder Variable.
- As part of a REST request header to include a range start date stored in a row variable. This feature is intended to provide support for workbooks with date effective objects. If you create a layout for a date effective object, you may need to create a REST request header of type `Effective-Of` that includes a range start date. See Configure a Request Header.
- As the default value for a search condition in the Search Editor or a download parameter in the Download Parameter Editor. See Configure a Search for a Layout and Use Download Parameters to Limit Downloaded Data.
- In the filter query parameter for a list of values. You can use an expression to filter the list of values based on the value of another cell in the row. See Configure a Cascading List of Values.

About Expressions

An expression is a string enclosed in curly braces (`{ }`) that can be evaluated to a single value at runtime. Expressions can reference configuration properties and dynamic runtime data.

The value of an operand or an intermediate result in an expression can be a Boolean value, string, or integer. However, the results of expressions may be converted to strings and concatenated if needed when resolving the entire property value. See String Representations.

For any given configuration property that supports expressions, you must escape any curly braces (`\{` or `\}`) that you wish to use literally.

Let's consider an example of a list of values for an employee `JobId` field that displays all job titles from the `jobId` field from the Jobs business object. To list only the job titles for a given department based on the `DepartmentId` of the current row, you could use a query parameter with the following expression:

```
DepartmentId={ this.BusinessObject.Fields['DepartmentId'].Value }
```

where:

- `this` represents the currently selected field;
- `BusinessObject` represents the business object to which this field belongs;

- `Fields['DepartmentId']` is the field (DepartmentId) associated with the business object; and
- `Value` is the value of the field.

You can also use `Parent` in an expression to refer to an ancestor business object ("parent" or higher) in a business object hierarchy. For example, to refer to a field in the parent business object you might use something like this:

```
ProjectNumber={ this.BusinessObject.Parent.Fields['ProjectNumber'].Value }
```

`Parent` can appear multiple times in the expression depending on the level you want to refer to in your business object hierarchy. To refer to the current business object's great grandparent business object, you'd use it three times:

```
ProjectNumber={ this.BusinessObject.Parent.Parent.Parent.Fields['ProjectNumber
'].Value }
```

Expressions can refer to a:

- Business object field (`BusinessObject.Fields['DepartmentId']`)
- Row finder variable (`Finder.Variables['CountryId']`)
- Workbook parameter (`Workbook.Parameters['Dept']`)
- Row variable (`BusinessObject.RowVariables['rangeStartDate']`)

Please note when using a collection element like `Fields['<fieldid>']` that if the field with the specified ID is not found the result is an error. The only exception is `Workbook.Parameters`. If the named workbook parameter is not found, the result is null.

> **Note:**
>
> Some workbook configuration properties support expressions and others do not. Those properties that support expressions may support all reserved words or only a subset of them. Consult the documentation for each property to determine what is, and is not, supported.

A Note on Spaces in Expressions

Keep in mind that spaces outside curly braces in an expression are included in the final string. So, for example, `DepartmentId= { this.BusinessObject.Fields['DepartmentId'].Value }` (space after the equals sign) would, given a department ID of `100`, yield `DepartmentId= 100`.

The extra space may or may not be important to the service.

Spaces immediately inside curly braces are not significant. For example, `{ this.BusinessObject.Fields['DepartmentId'].Value }` is equivalent to `{this.BusinessObject.Fields['DepartmentId'].Value}`.

Literal Values in Expressions

Literal values of certain data types are supported in expressions.

Support is included for these data types:

- Boolean
- String
- Integer or floating-point number
- Date
- DateTime

Boolean and number literal values must be in the form described here. For example, if you are in a country that uses a decimal comma (,), you must still use a decimal point or period in your expression.

Data Type	Description
Boolean	Supported values (case-sensitive, no quotes): • `TRUE` • `True` • `true` • `FALSE` • `False` • `false` **Note:** It's recommended that you use capital case only (`TRUE` and `FALSE`).
String	String literals inside expressions must be enclosed in single quotes ('). Single quotes inside string literals must be escaped (`\'`).

Data Type	Description
Integer or Floating-Point Number	Only the Western Arabic numerals (0-9) can be used. Other digits are not supported. These symbols are supported: • Leading negative sign (-) • Decimal separator (.) • Exponent (`E` or `e` followed by an optional sign and exponential digits) **Note:** This is allowed only when a decimal separator is present; for example, when the value is a floating-point number. These symbols are not supported: • Leading positive sign (+) **Note:** The plus sign when used as an operator is supported. • Thousand separator such as a space, period, comma, or underscore • No digit after the decimal separator
Date	A date type literal value must start with a prefix letter `d` followed by a single-quoted ISO 8601 date (`'yyyy-MM-dd'`), such as: `d'2020-10-24'` or `d'2000-01-01'`
DateTime	A date-time type literal value must start with the prefix letters, `dt`, followed by a single quoted ISO 8601 date-time in UTC (`'yyyy-MM-ddTHH:mm:ssZ'`), such as: `dt'2020-10-24T12:34:56Z'` **Note:** There is no millisecond component. Always include the `T` between date and time, and the `Z` at the end (indicates that it's UTC).

> **Note:**
>
> These rules only apply to literal values used in expressions. They do not apply to data formats used in an Excel cell. For example, you must write 135000 without thousand separator in this validation rule { this.Value > 135000 } but that validation rule can be used on an integer field that shows 150,000 or 150.000 in cell.

Operators in Expressions

The Oracle Visual Builder Add-in for Excel expression language supports a number of operators. Refer to this table for details.

Operator precedence is high to low.

Operator	Use
`[ ] . ?.`	Collection access, object member access, conditional object member access
`( )`	Grouping to change precedence
`- !`	Unary minus, negation
`* /`	Math (multiplicative)
`+ -`	Math (additive), also + for string concatenation
`< > <= >=`	Relational
`== !=`	Equality
`&&`	Logical AND
`||`	Logical OR
`??`	Null-coalescing
`? :`	Ternary conditional

String Representations

When Oracle Visual Builder Add-in for Excel evaluates an expression that includes non-string values—such as Boolean values, dates, and numbers—it may convert these values to strings when, for example, the configuration property expects strings.

Review this table for information on how each data type value is represented as a string.

Data Type	String Representation	Example
Boolean	true or false (note that Excel may capitalize the first letter or convert to other values)	false
Date-time	Full UTC string representation (ISO 8601)	2023-10-24T12:33:19Z
Date (no time)	Date-only string representation (ISO 8601)	2023-10-24
Integer	No thousand separator	12300000
Number	No thousand separator; period as decimal separator	0.123

Numbers in Expressions

Some number formats are not supported in expressions. Refer to this table for supported and unsupported formats.

Here are some examples of how numbers are supported in expressions:

Supported	Not Supported
0.123	0,123 .123
123 123.0	123.
-456	+456 (part of literal value instead of doing addition)
1234	1,234 1 234
1234.567	1,234.567
3.14E2	03.14E2
1.0e10	1e10

Dates in Expressions

Oracle Visual Builder Add-in for Excel supports the use of date type values in expressions. So, for example, you can pass date values to lists of values filters or REST request header fields with this support.

Literals

A date type literal value must start with a prefix letter `d` followed by a single-quoted ISO 8601 date (`'yyyy-MM-dd'`), such as:

`d'2020-10-24'` or `d'2000-01-01'`

Operations

The following operators are supported: `+`, `-`, `<`, `<=`, `>`, `>=`, `==`, and `!=`.

Supported +/- operations are:

- Date + Integer
- Integer + Date
- Date - Date (returns the difference in days as an integer value)
- Date - Integer

Here are some examples of expressions using supported operators. Resulting dates are in the default `yyyy-MM-dd` format.

Operation	Expression	Result
Date + Integer	`{d'2023-10-24' + 2}`	2023-10-26
Integer + Date	`{-2 + d'2023-10-24'}`	2023-10-22

Operation	Expression	Result
Date - Date	`{d'2023-10-24' - d'2023-10-20'}`	4
Date - Integer	`{d'2023-10-24' - 2}`	2023-10-22
Date >= Date	`{d'2023-10-24' >= d'2023-10-22'}`	True
Date != Date	`{d'2023-10-24' != d'2023-10-22'}`	True

Functions in Expressions

The add-in supports a couple of functions:

- `Today ()`
- `Format(object obj, string formatString)`

> **Note:**
>
> Function names are case-sensitive.

The `Today ()` function returns today's date.

Suppose you want to create an expression for a search parameter that returns rows with hire dates that are later than 90 days before today's date. You would create a search parameter with a parameter name "q" and a parameter value of:

```
HireDate > '{ Today() - 90 }'
```

> **Note:**
>
> In this example, you would select the **Allow expressions in Parameter Value** check box.

The Format function returns the string representation of the given date in a given format using "invariant" culture (a culture that is culture-insensitive). See InvariantCulture.

It includes two arguments:

- `obj` is a date. This can be a literal, a cell value from a field whose data type is `Date (no time)`, the result of `Today()`, or the result of `+/-` operations (see the Operations section in this topic).
- `formatString` is the date format. Supported formats are:
  - *yyyy-MM-dd* (default)
  - *MM-dd-yyyy*
  - *dd-MM-yyyy*

Take, for example, a service that requires a date in the *dd-MM-yyyy* format. You would use the Format function to provide this format rather than the default format, *yyyy-MM-dd*, like this:

```
HireDate > { Format(this.BusinessObject.Fields['HireDate'].Value, 'dd-MM-
yyyy') }
```

If the HireDate value for the current row is "2023-10-24", the final value of the line is:

```
HireDate > 24-10-2023
```

> **Note:**
>
> The resulting string does not include quotation marks. If the service requires quotes, add them explicitly. For example, to return a value with quotes, like `HireDate > '24-10-2023'`, use:
>
> ```
> HireDate > '{ Format(this.BusinessObject.Fields['HireDate'].Value,
> 'dd-MM-yyyy') }'
> ```

Notes on Dates in Expressions

- Once you configure a property with an expression that relies on date values or date operations, the workbook is no longer compatible with add-in versions prior to 3.8.
- Date values do not have a time part. Date values do not have a time zone.
- Literal input, operations, and Format () are NOT supported for date-time values (from fields whose data type is Date-time).
- There is no function that parses a string and returns a date.

Dates and Times in Expressions

Oracle Visual Builder Add-in for Excel supports the use of date-time type values in expressions. So, for example, you can pass date-time values to lists of values filters or REST request header fields with this support.

Time Values

The add-in doesn't store time zone information for date-time values. Time values are local. Note that:

- The add-in converts Coordinated Universal Time (UTC) values from the literal value to the local time value.
- During calculation, date-time values are processed as local values.
- The add-in converts local date-time to UTC date-time in its canonical string form when it's included in REST requests.

Literals

A date-time type literal value must start with the prefix letters, `dt`, followed by a single quoted ISO 8601 date-time in UTC (`'yyyy-MM-ddTHH:mm:ssZ'`), such as:

```
dt'2020-10-24T12:34:56Z'
```

> **Note:**
>
> There is no millisecond component. Always include the `T` between date and time, and the `Z` at the end (indicates that it's UTC).

Operations

The following operators are supported: `+`, `-`, `<`, `<=`, `>`, `>=`, `==`, and `!=`.

Supported +/- operations (order matters)

- DateTime + Integer
- Integer + DateTime
- DateTime - Integer
- DateTime - DateTime (returns the diff in seconds as an integer value)

Here are some examples of expressions using supported operators.

Operation	Expression	Result (local time is PDT or UTC-7:00)	Notes
DateTime + Integer	`{dt'2023-10-24T12:34:56Z' + 2}`	2023-10-24 05:34:58 (2023-10-24T12:34:58Z)	The integer (seconds) is added to the date-time value. Internally evaluating 2023-10-24 05:34:56 + 2
Integer + DateTime	`{2 + dt'2023-10-24T12:34:56Z'}`	2023-10-24 05:34:58 (2023-10-24T12:34:58Z)	Internally evaluating 2 + 2023-10-24 05:34:56
DateTime - Integer	`{dt'2023-10-24T12:34:56Z' - 2}`	2023-10-24 05:34:54 (2023-10-24T12:34:54Z)	The integer (seconds) is subtracted from the date-time value. Internally evaluating 2023-10-24 05:34:56 - 2
DateTime - DateTime	`{dt'2023-10-24T12:34:56Z' - dt'2023-10-24T12:34:50Z'}`	6	Internally evaluating 2023-10-24 05:34:56 - 2023-10-24 05:34:50

Operation	Expression	Result (local time is PDT or UTC-7:00)	Notes
DateTime > Date	`{dt'2023-10-24T12:34:56Z' > d'2023-10-22'}`	true	Internally evaluating 2023-10-24 05:34:56 > 2023-10-22 00:00:00

Note: When a date(notime) value is compared with a date-

Operation	Expression	Result (local time is PDT or UTC-7:00)	Notes

```
timevalue, it's first appended date time of day component of 00:00:0
```

Operation	Expression	Result (local time is PDT or UTC-7:00)	Notes
			0 (local).

Functions in Expressions

The add-in supports the Now function, `Now ()`, that returns the current date and time as a date-time value. There are no parameters for this function.

> **Note:**
>
> Function names are case-sensitive.

Suppose you want to create an expression that displays a `True` or a `False` for an "overdue" field if a deadline has passed. In this case, you might want to compare a date-time value in the same row ("deadline" field) with the current date and time, like this:

```
{ this.BusinessObject.Fields['deadline'].Value < Now () }
```

In this expression, if the date-time value in the "deadline" field is earlier than now, the expression evaluates to `True`. If the "deadline" field value is "2024-04-15 09:00:00" and the current date and time (provided by the `Now ()` function) is "2024-04-15 14:00:00" (both using local date-time), the expression result *is* `True` and the item is overdue.

Limitations

Setting the time zone is not supported.

Reserved Words and Properties Used in Expressions

The Oracle Visual Builder Add-in for Excel expression language includes some reserved words and properties. Refer to this table for how these are used in add-in expressions. Please note that this list is not exhaustive.

Reserved Word	Note
`this`	Represents the property owner depending on the configuration context. For example, when defining a field's configuration property, "this" represents the field. See specific configuration properties for details.

Reserved Word	Note
`BusinessObject`	Represents the business object to which the currently selected field belongs
`Parent`	Represents the parent business object of the currently selected field's business object in a business object hierarchy. Use additional instances in your expression to refer to higher level business objects, such as `Parent.Parent` for the grandparent business object and so on.
`Fields`	Represents fields of a business object
`Value`	The value of the referenced business object field, workbook parameter, row variable, and so on
`SelectWindow`	Search-and-select window in a list of values
`Finder`	Represents the row finder to which the currently selected variable belongs
`RowVariables`	Represents row variables configured for a business object
`Workbook`	Represents the integrated workbook
`Parameters`	Represents workbook parameters stored in the workbook

Handling Null Values in Expressions

Oracle Visual Builder Add-in for Excel supports the use of `null` in expressions to refer to null values. This allows you to write expressions that handle null values that may arise when the expression is evaluated.

You can use a number of approaches to handle a null value such as using a conditional statement to substitute a non-null value for the null. For more information on using null in expressions, see Null Values in Expressions.

Before version 4.1, the add-in automatically replaced null values with a fixed default value appropriate for the data type—a zero (0) for an integer field and a space (" ") for a string. By default, older workbooks keep this behavior. If you want to adopt the new behavior, you can turn off the legacy behavior from the Workbook Info window. See Disable the Legacy Null Handling Behavior.

Null Values in Expressions

The add-in includes `null` to refer to null values in fields and parameters referenced in expressions.

Literal

Use `null` in expressions to represent a null value, like this:

```
{ this.Value != null }
```

> **Note:**
>
> Use lower case only without quotation marks.

Operations

Null values can be used as operands for the `==` and `!=` operators, like this: `{ this.value == null }` and `{ Workbook.Parameters['param1'] != null }`.

Null handling supports the following additional operators:

Operator	Use	Description
?.	Conditional object member access	Performs member access only when the operand is non-null. Otherwise, it returns null.
??	Null-coalescing	Returns the value of its left-hand operand if it isn't null. Otherwise, it evaluates the right-hand operand and returns its result.

Expression Examples

Expression	Description
`{ Workbook.Parameters['maxSalaryParam']?.Value }`	If the workbook parameter `maxSalaryParam` is missing, the expression returns null. Otherwise, it returns the value of `maxSalaryParam`.
`{ this.Value ?? '' }`	If this field's value is null, the expression returns an empty string. Otherwise, it returns the field value.

Disable the Legacy Null Handling Behavior

If you have an old workbook that uses the add-in's "legacy" null handling behavior, you disable the behavior and use the new behavior instead.

There are two null handling behaviors the add-in supports when evaluating expressions:

- **Legacy** (version 4.0 and earlier): The add-in automatically replaced null values with a fixed default value appropriate for the data type—a zero (0) for an integer field and a space (" ") for a string. By default, older workbooks use this legacy behavior when opened by later versions of the add-in.
- **New** (version 4.1 and later): The add-in does not make this substitution by default. You must write expressions that handle null values that may arise.
  For new workbooks, the new behavior is enabled by default.

These properties are affected by this option:

- REST header values
- Field validation rules
- Default values for Search Editor search condition
- Default values for row finder variables
- Download parameter values

> **WARNING:**
>
> Before you change this setting, review all your expressions and update them to handle null values as required. If you do not update your expressions, you may encounter failures.

To disable the legacy null handling behavior:

1. Choose **Workbook Info** from the **Advanced** menu to open the Workbook Info window.
2. Click **Settings**, then deselect **Legacy null handling for expressions**.
3. Test your workbook thoroughly.

Workbook Parameters in Expressions

Oracle Visual Builder Add-in for Excel supports references to workbook parameter values in expressions to control search behavior during a download.

There are three cases where expressions can reference workbook parameter values:

- Comparison value for a Search Editor condition
- Default value expression for row finder variables
- Download parameter values

Missing Workbook Parameters

The expression you write may refer to a workbook parameter that for whatever reason is not present in the workbook. You can write your expression to guard against such cases.

Let's suppose you have a search condition on a Salary field that filters for items less than a maximum value specified by a workbook parameter named `maxSalaryParam`.

Your search field is Salary, the operator is "less than" (<) and the expression is:

```
{ Workbook.Parameters['maxSalaryParam']?.Value }
```

If `maxSalaryParam` is not present in the workbook, the expression evaluates to null, and the entire search condition will be omitted from the download request. See Null Values in Expressions for more information about the `?.` operator.

Empty Values in Workbook Parameters

The expression you write may refer to a workbook parameter that is present in the workbook but has no value specified.

Building off of the previous example, let's say that you want to use 5000 as the default value for `maxSalaryParam` if it is missing, or if it is present but has no value specified. In this case, the expression is:

```
{ Workbook.Parameters['maxSalaryParam']?.Value ?? '5000' }
```

See Null Values in Expressions for more information about the `??` operator.

Examples of Expressions

Here are some sample expressions and their uses.

Property Value	Use	Sample Value	Final Property Value
`DepartmentId={ this.BusinessObject.Fields['DepartmentId'].Value }`	This string sets the value of `DepartmentId` in the query to the current row item's department Id value.	Department Id is `101`	`DepartmentId=101`
`DepartmentId={ this.BusinessObject.Fields['DepartmentId'].Value } { SelectWindow.SearchTerm == '' ? '' : 'AND FirstName LIKE \'' + SelectWindow.SearchTerm + '*\'' }`	This string includes two expressions. The second expression uses the ternary operator. It returns results based on whether there is a search term in the search box. If there is no search term, the parameter returns values matching the current row item's department Id value. If there is a search term, the parameter returns results that match the department Id *and* the search term. The quotes are all single quotation marks. Note also the enclosed empty strings and escaped single quotes.	Department id is `101` and there is no search term	`DepartmentId=101`
		Department id is `101` and the search term is `Steve`	`DepartmentId=101 AND FirstName LIKE 'Steve*'`
`DepartmentId={ Workbook.Parameters['Dept'].Value } AND Salary >= { Workbook.Parameters['MinSal'].Value }`	This string sets the value of `DepartmentId` in the query to the value of the `Dept` workbook parameter and the value of `Salary` to the `MinSal` workbook parameter.	The workbook parameter `Dept` is `80` and `MinSal` is `7000`.	`DepartmentId=80 AND Salary >=7000`
`{ (this.Value ?? Today()) > d'2024-01-01' }`	This expression represents a Boolean value that is dependent on `this.Value`. It returns true if `this.Value` is later than date 2024-01-01 (ISO 8601 *yyyy-MM-dd*). If `this.Value` is null, it uses `Today()` for comparison.	`this.Value` is null	`true`

23
The Embedded Browser

Oracle Visual Builder Add-in for Excel uses the Microsoft WebView2 control as an embedded web browser to display web pages from inside Microsoft Excel. WebView2 is based on Edge/Chromium.

The embedded web browser is used to display the log-in web page when authenticating using Oracle Fusion Applications Token Relay or OAuth 2.0 Authorization Code Flow. See Authentication Options.

Your default web browser setting in Windows Settings has no effect on the add-in.

The WebView2 Control

Microsoft Edge WebView2 is an embedded web browser based on Edge/Chromium. In order for Oracle Visual Builder Add-in for Excel to use WebView2, the WebView2 runtime must be installed on each computer where the add-in runs.

Installation

You can download the runtime from here: https://developer.microsoft.com/en-us/microsoft-edge/webview2/consumer/.

> **Note:**
>
> The WebView2 runtime may already be present on your computer if you have Microsoft 365 Apps installed. See Microsoft Edge WebView2 and Microsoft 365 Apps.

Technical Notes

- When installing the WebView2 runtime, choose one of the "evergreen" installers. Do not choose the "Fixed Version" option.
- During the login sequence, using the WebView2 browser's Refresh function can cause issues, particularly if it is performed early in the login sequence. Users should avoid using Refresh.
- If you encounter a problem with the log-in web page, please contact the page owner. Such pages are outside the scope of the add-in. Let the page owner know that the page needs to be compatible with Edge/Chromium. Refer to Feature differences between Microsoft Edge and WebView2 on the Microsoft web site for more information.
- When the add-in uses the WebView2 browser control, the browser's SmartScreen feature is disabled. See the Microsoft Defender SmartScreen Frequently Asked Questions or the documentation for more information.
- The WebView2 browser control uses a user data folder on the local computer to store browser data, such as cookies, permissions, and cached resources. This folder can be found under `%LocalAppData%\Oracle\Visual Builder\`. For example, `C:\Users\username\AppData\Local\Oracle\Visual Builder\EBWebView`.

To clear the browser cache for the WebView2 browser control, refer to Clear the Embedded Browser Cache.

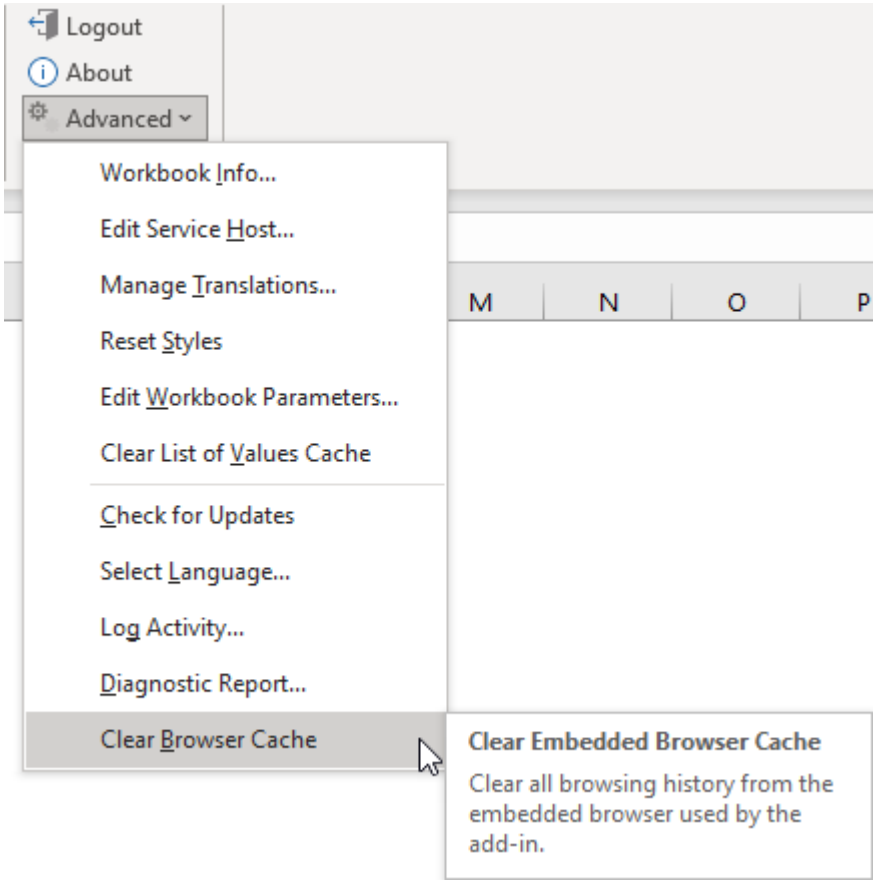
See also Manage the User Data Folder in the Microsoft Edge documentation.

- When the add-in uses the WebView2 browser control, it enables some Microsoft capabilities relating to single sign-on (SSO) with Azure Active Directory, such as the `AllowSingleSignOnUsingOSPrimaryAccount` property. See the AllowSingleSignOnUsingOSPrimaryAccount property in the *Microsoft API Reference*.

Clear the Embedded Browser Cache

If required, you can clear the cache for the embedded browser to get rid of all browser data including profile data such as history, bookmarks, and cookies. You may want to try clearing the browser cache if, for example, you are having an issue with a login page.

To clear the cache for the embedded browser, choose **Clear Embedded Browser Cache** from the **Advanced** menu.



Note:

You can also clear this cache by simply deleting the `EBWebView` folder at `%localappdata%\Oracle\Visual Builder\EBWebView`. The browser recreates the folder when the add-in next accesses the browser.

24
Third Party Licenses

Oracle Visual Builder Add-in for Excel includes third-party software which requires the user to reproduce all copyright notices, permission notices, conditions, and disclaimers. The following third-party software license information is reproduced here in compliance with the terms of these licenses.

Microsoft.OpenApi, Version 1.6.10

Copyright (c) Microsoft Corporation. All rights reserved.

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED *AS IS*, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Microsoft.Web.WebView2, Version: 1.0.1774.30

Copyright (c) Microsoft Corporation. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of Microsoft Corporation, or the names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,

DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NewtonSoft.Json, Version 13.0.3

The MIT License (MIT)

Copyright (c) 2007 James Newton-King

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SharpYaml, Version 2.1.0

Copyright (c) 2013-2022 SharpYaml - Alexandre Mutel

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SharpYaml is a fork of YamlDotNet https://github.com/aaubry/YamlDotNet published with the following license:

Copyright (c) 2008, 2009, 2010, 2011, 2012 Antoine Aubry

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute,

sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The following is additional License Text from the file CharHelper.cs at the root of SharpYaml folder in the source folder

==========
=

// Code from coreclr with MIT License // https://github.com/dotnet/coreclr/blob/e3eecaa56ec08d47941bc7191656a7559ac8b3c0/src/mscorlib/shared/System/Char.cs#L1018 // Licensed to the .NET Foundation under one or more agreements. // The .NET Foundation licenses this file to you under the MIT license. // See the LICENSE file in the project root for more information.

Content of License Text from the License.txt file at the root folder of the sources at https://github.com/dotnet/coreclr

==========

The MIT License (MIT)

Copyright (c) .NET Foundation and Contributors

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Content of Patents.txt file at the root folder of the sources at https://github.com/dotnet/coreclr

==========

Microsoft Patent Promise for .NET Libraries and Runtime Components

Microsoft Corporation and its affiliates ("Microsoft") promise not to assert any .NET Patents against you for making, using, selling, offering for sale, importing, or distributing Covered Code,

as part of either a .NET Runtime or as part of any application designed to run on a .NET Runtime.

If you file, maintain, or voluntarily participate in any claim in a lawsuit alleging direct or contributory patent infringement by any Covered Code, or inducement of patent infringement by any Covered Code, then your rights under this promise will automatically terminate.

This promise is not an assurance that (i) any .NET Patents are valid or enforceable, or (ii) Covered Code does not infringe patents or other intellectual property rights of any third party. No rights except those expressly stated in this promise are granted, waived, or received by Microsoft, whether by implication, exhaustion, estoppel, or otherwise. This is a personal promise directly from Microsoft to you, and you agree as a condition of benefiting from it that no Microsoft rights are received from suppliers, distributors, or otherwise from any other person in connection with this promise.

Definitions:

"Covered Code" means those Microsoft .NET libraries and runtime components as made available by Microsoft at https://github.com/dotnet/coreclr, https://github.com/dotnet/corefx and https://github.com/dotnet/corert.

".NET Patents" are those patent claims, both currently owned by Microsoft and acquired in the future, that are necessarily infringed by Covered Code. .NET Patents do not include any patent claims that are infringed by any Enabling Technology, that are infringed only as a consequence of modification of Covered Code, or that are infringed only by the combination of Covered Code with third party code.

".NET Runtime" means any compliant implementation in software of (a) all of the required parts of the mandatory provisions of Standard ECMA-335 – Common Language Infrastructure (CLI); and (b) if implemented, any additional functionality in Microsoft's .NET Framework, as described in Microsoft's API documentation on its MSDN website. For example, .NET Runtimes include Microsoft's .NET Framework and those portions of the Mono Project compliant with (a) and (b).

"Enabling Technology" means underlying or enabling technology that may be used, combined, or distributed in connection with Microsoft's .NET Framework or other .NET Runtimes, such as hardware, operating systems, and applications that run on .NET Framework or other .NET Runtimes.