Oracle® Cloud

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

Version 3.3
F61239-01
October 2022

ORACLE®

Oracle Cloud Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel, Version 3.3

F61239-01

Contents

5   Manage Catalogs and Business Objects

6   Configure Search Options for Download

7   Custom Actions

8   Use Lists of Values in an Excel Workbook

9   Appearance of an Integrated Excel Workbook

10   Upload Changes

11   Use Multiple Layouts for Multi-level Business Objects

12   Use Macros in an Integrated Excel Workbook

13   Publish an Integrated Excel Workbook

14 REST Service Support

15 Internationalization

16 Security

17 Troubleshoot Excel Workbooks

18 Migration

19 About Expressions

20 Embedded Browsers

21 Third Party Licenses

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
- Documentation Accessibility
- Diversity and Inclusion
- Related Resources
- Conventions

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 `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

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

Diversity and Inclusion

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

Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud

- About Oracle Visual Builder in *Developing Applications with Oracle Visual Builder*

- Introduction to Accessing Business Objects in *Accessing Business Objects Using REST APIs*

- 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 3.3.0

Here's an overview of new features and enhancements added to Oracle Visual Builder Add-in for Excel in Release 3.3.0. Take note also of some limitations when using the add-in.

New and Changed Features

- A new **Send Only Changed Data for Updates** check box that determines which values in a table row marked for update are included in the payload during upload. When this check box is enabled, only values in cells that have changed since the last download or upload are included. Send Only Changed Data During Upload.

- The **Refresh Field Titles** command is now available for all supported REST services. See Refresh All Field Titles .

- Support for using fields in searches that have lists of values (LOV) that use dynamic filters. You can now create a search in the Search Editor that includes two fields and then filter the values in the second field's list based on the value selected for the first. See Use Search to Limit Downloaded Data.

- A new Workbook Info viewer that provides status information for a workbook. If a workbook is unusable, the business user can open this viewer to determine the cause and troubleshooting steps. See Resolve Workbook Issues.

- Improvements to the **Refresh Catalog** command. You can now refresh a catalog that includes business objects that you have imported separately and that have their own service description documents. When you refresh a catalog, Oracle Visual Builder Add-in for Excel reads the metadata path for each business object and issues a request that returns the business object details from each service description document. See Refresh a Business Object Catalog.

- A new **Metadata Path** field in the Business Object Editor that displays the relative path to the business object's service description document. See Service Descriptions.

- Support for adding ancestor columns from the form part of the root Form-over-Table layout to a descendant layout. See Add Ancestor Columns to a Dependent Layout.

- An update to the log-in page to display the URL of the service the business user is logging in to. By showing the URL, business users can decide whether they want to provide their log-in credentials to proceed.

- Improvements to the user experience such as freezing header rows in new table layouts When you create a new table layout, the add-in freezes the top row so you always see the column headers when you scroll through your table. See Create a Table Layout in an Excel Workbook. The add-in also now allows business users to navigate through editors and wizards by double-clicking.

Known Limitations

- Review REST Service Support Limitations to understand the limitations around data type support.

- Review Notes and Limitations for Lists of Values to understand limitations around using lists of values.

- Review Custom Action Limitations to understand the limitations of custom action support.

- Workbooks created or modified with version 3.3 are generally considered to be incompatible with prior versions of the add-in.

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.
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.
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 description 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.

Installation

Install the latest version of Oracle Visual Builder Add-in for Excel available from the Downloads page on Oracle.com.

Before you begin, ensure that your environment meets the requirements for installation. See Software Dependencies and Supported Platforms for more details.
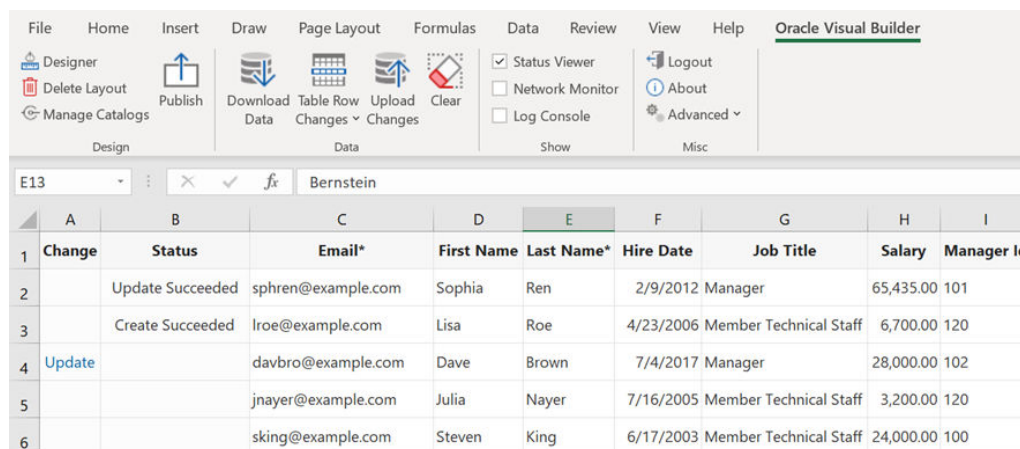
When you are ready, install the add-in as described in Install Oracle Visual Builder Add-in for Excel.

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:

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 description 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 pane on the ribbon.
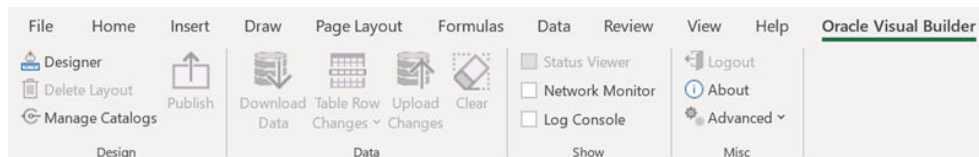
> ✎ **Note:**
>
> If you are unable to install the add-in, refer to the installation log. See Installation Logs.

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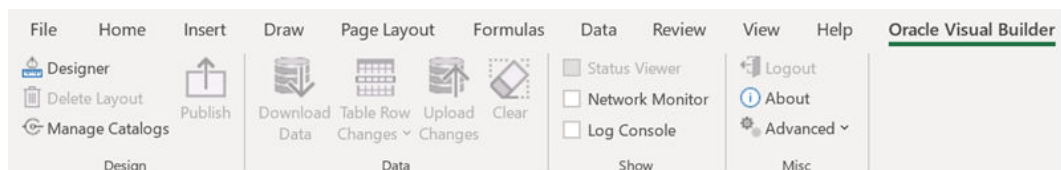
   When you first run Excel after the current-user installation, you may be prompted to confirm the installation of the "Office customization". This prompt generally appears the first time for each profile and whenever the digital certificate (used to sign the add-in) is changed.

   > **Note:**
   >
   > If you are unable to install the add-in, refer to the installation log. See Installation Logs.

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.

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

To check for new versions, select **Check for Updates** from the **Advanced** drop-down on the Oracle Visual Builder ribbon. If a newer version is available, you'll be prompted to open the Oracle Downloads page in your browser.

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** drop-down 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.

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 the add-in from the Windows Settings app.

If you're uninstalling the Current User version, sign in to the Windows user profile where the add-in is installed.

1. From the Start Menu, select **Settings**, then **Apps**.

2. 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.

3. Click **Uninstall** and follow the instructions.

4. If you have performed a current user installation for multiple Windows user profiles and you want to uninstall them all, log in to each profile and repeat these steps.

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
Microsoft Visual Studio 2010 Tools for Office (VSTO) Runtime	The VSTO runtime is included with most recent versions of Excel
Microsoft Edge WebView2	The add-in supports the use of Microsoft Edge WebView2 as an embedded browser for displaying log-in pages. See Embedded Browsers.

The installer checks your system for required software and quits without installing the add-in if any required software is missing. Run the Health Check tool to ensure you have all required software and versions. If you are missing these required components, follow the instructions in the Health Check tool to install the required software.

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.

> **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 are determined by Microsoft support policies. See Microsoft Lifecycle Policy.

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.

Layouts

Oracle Visual Builder Add-in for Excel lets you create different layouts to work with data in an Excel worksheet. Layouts are a way to display a business object in an Excel worksheet. Each worksheet supports one of two layouts: **Table** or **Form-over-Table**.

- Use a Table layout to view and edit data from a REST service in a tabular format. Here's a worksheet showing employee data in a Table layout:

- Use a Form-over-Table layout when a parent-child relationship exists in the business objects used by your web application. Here's a worksheet 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:

You can create one layout per worksheet in your Excel workbook. Layouts are created by workbook developers and are visible to data entry users in their workbooks.

Service Descriptions

To create a layout in the workbook, the REST services that you use must provide a service description that complies with the OpenAPI specification. The service description can be a URL or a local file. For Oracle business object REST API services, the URL typically includes a describe path segment, as in `https://my-service-host/fscmRestApi/resources/latest/invoices/describe`. For Oracle REST Data Services (ORDS), the URL may be similar to `https://host/ords/great_app/open-api-catalog/employees/`.

You can provide the service description document when you create a layout by clicking **Designer** in the Oracle Visual Builder tab, as described in subsequent sections. You can also provide the service description document by clicking **Manage Catalogs** in the Oracle Visual Builder tab (see Manage Catalogs and Business Objects).

The service description 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.

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 description document. The service description document can be stored on your local drive or accessed remotely using an 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.

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. In the Excel ribbon, select the **Oracle Visual Builder** tab.

3. Click the cell where you want to locate the data table.

4. In the Oracle Visual Builder tab, click **Designer** to launch the **New Layout Setup** wizard.

5. From the first screen, provide the service description document using one of these options:

   • **Web Address** option (the default) if you access the service description 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 description document is a local file on your computer.

   > 💡 **Tip:**
   >
   > If you are working with Oracle business object REST API services, the URL for the service description usually ends with `/describe`, for example, `https://my-service-host/fscmRestApi/resources/<version>/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 for descriptions of the fields.

8. Select a business object and click **Next**.
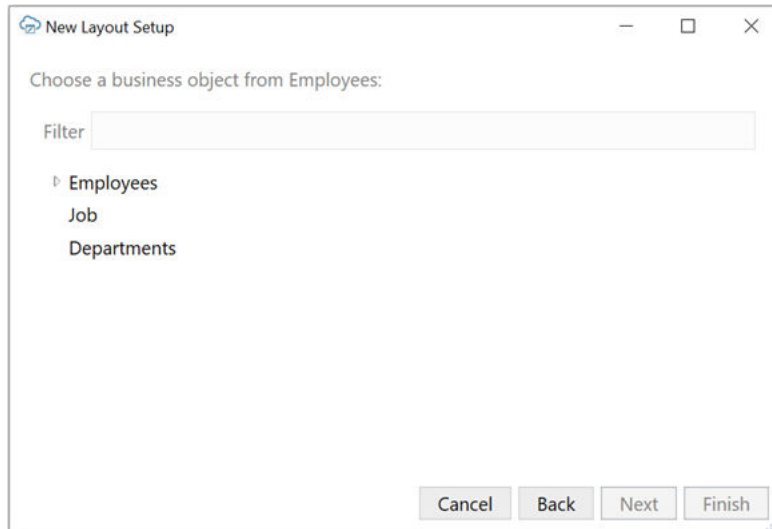
9. Select **Table Layout** and click **Next**.
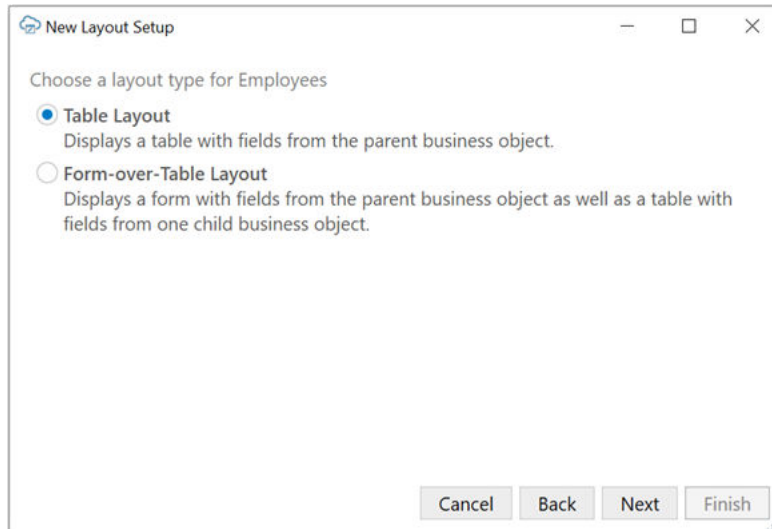
10. Review the Table layout details and then click **Finish**.
    The add-in creates a Table layout in the Excel workbook that includes column headers and a placeholder data row. The Layout Designer opens in the Excel Task Pane.

    > ✎ **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.
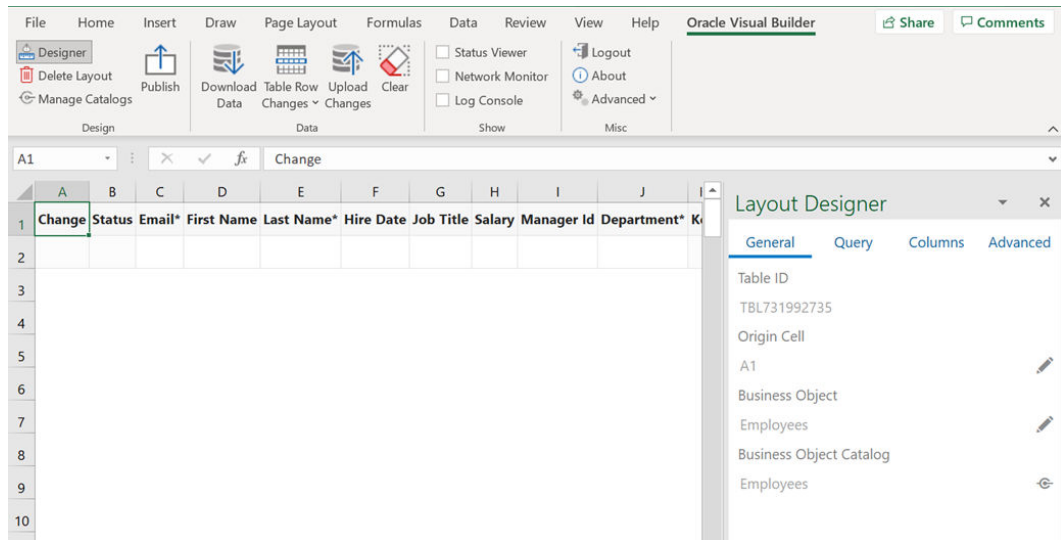
11. Click the **Columns** tab in the Layout Designer to add or remove columns, or change the order in which columns appear. When a data table is created, most (but not all) fields are added as columns. Click the Manage Columns button (➕) to add or remove columns as needed.
You can also modify a field associated with a column to, for example, show a tooltip or display a list of input values for your business users. To update a field, select it from the Table Columns table and click the Edit button (✎) to open the Business Object Field Editor. See Configure Business Object Fields.

To add a tooltip, click the **General** tab and type help text in the **Tooltip** property. The tooltip is displayed when the user selects the table column header. To create a list of values, see Configure a List of Values.

12. Optionally, configure the workbook further to limit the data that the add-in downloads, as described in Configure Search Options for Download.

The workbook is now complete and ready to be published. At this point, it's a good idea to perform various data operations, such as download, update, and upload, to test the workbook before you publish and 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*.

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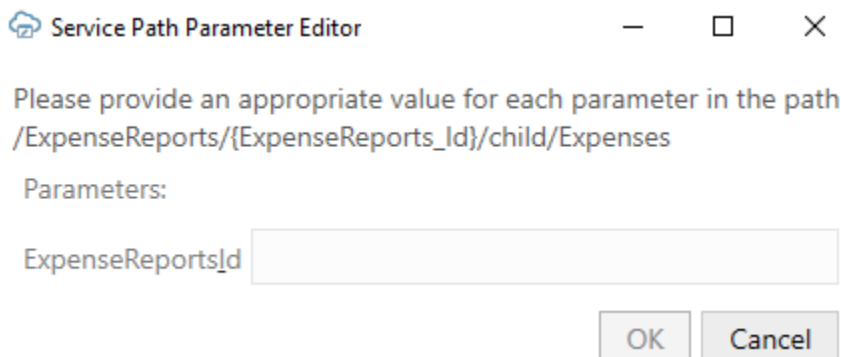
Some service paths include path parameters. The add-in provides support for configuring a Table layout using a parameterized service path. It automatically extracts the path parameters and prompts the user to provide the corresponding values at download time.

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

> 💡 **Tip:**
>
> When working with Oracle business object REST API services, you should start with the web address to the parent business object description (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` description (not `Expenses`). Oracle business object REST API services cannot provide OpenAPI service descriptions for parameterized service paths.

Complete the layout configuration. When users click **Download Data** in the Oracle Visual Builder tab, the add-in displays the Service Path Parameter Editor where users provide the required path parameter values that enables the download of data to complete.

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 values used on the URL path are not URL-encoded, so the values entered must be acceptable by the REST service.

Here is an example of a service path with an embedded parameter:

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

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

Using the Service Path Parameter Editor, you provide the proper value for `{ExpenseReports_Id}`, for example, `123456`, which results in the add-in using the following path:

`/ExpenseReports/123456/child/Expenses/`

Accessing this service path will provide all the expenses for expense report `123456`.

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.

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.

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 expose 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 description document. The service description document can be stored on your local drive or accessed remotely using an 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.

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. In the Excel ribbon, select the **Oracle Visual Builder** tab.
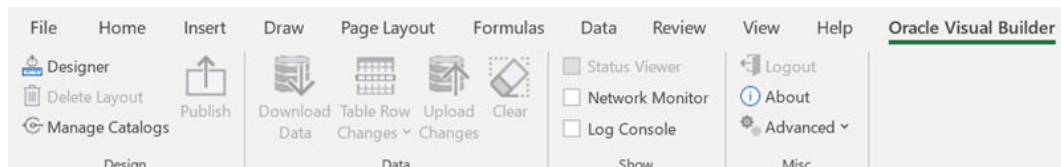
3. Click the cell where you want to locate the form and table.

4. In the Oracle Visual Builder tab, click **Designer** to launch the **New Layout Setup** wizard.

5. From the first screen, provide the service description document using one of these options:

- **Web Address** option (the default) if you access the service description 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 description document is a local file on your computer.

> **Tip:**
>
> If you are working with Oracle business object REST API services, the URL for the service description 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 for descriptions of the fields.

8. Choose the top-level business object for the form (in this case, `purchaseOrders`), and click **Next**.
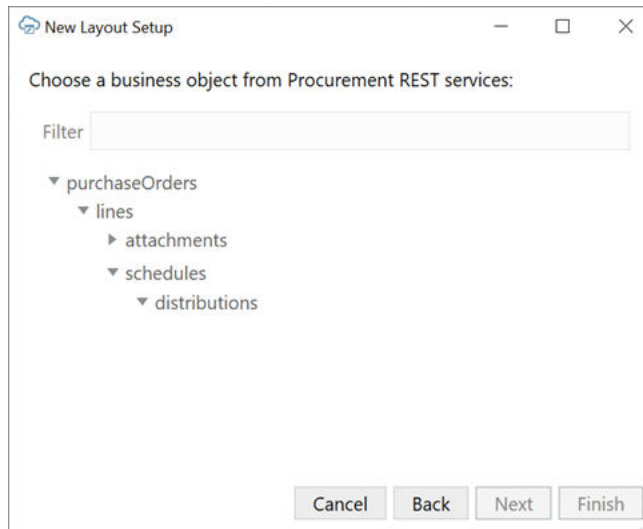
9. Choose **Form-over-Table Layout**, and click **Next**.
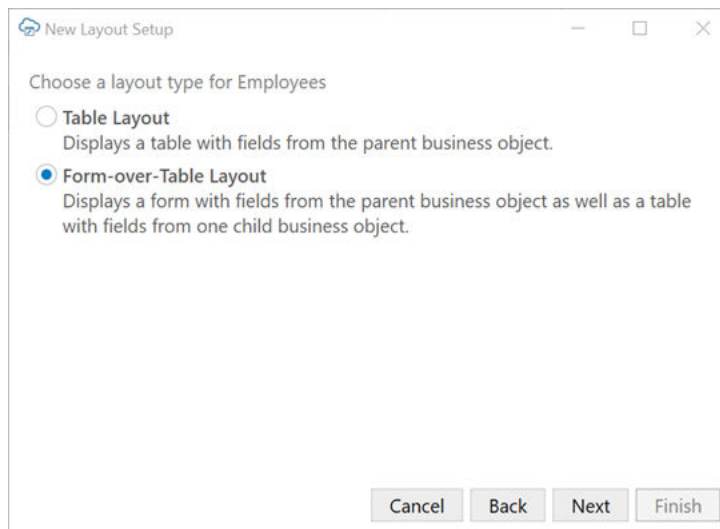
The Form-over-Table layout option is unavailable if you didn't select the top-level business object or there is no available child business object.

10. Choose a child business object (in this case, `lines`), and click **Next**.

11. 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:
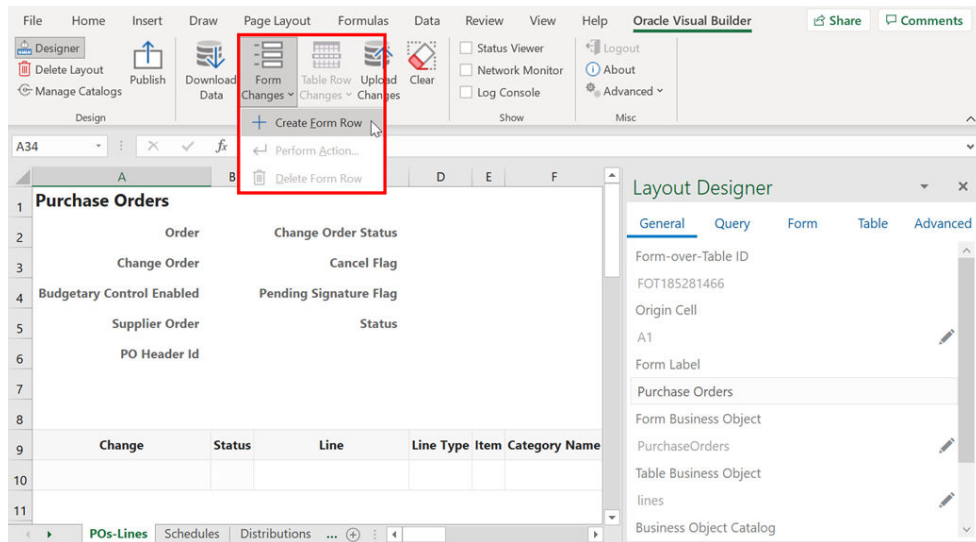
If the form business object (in this case, `purchaseOrders`) supports a create action, a **Create Form Row** option appears in the Form Changes menu, as shown in the image. Use this option to create a new form row during your next upload (see Create a Parent Row in a Form-over-Table Layout in *Managing Data Using Oracle Visual Builder Add-in for Excel*).

12. Customize the form and table by modifying the automatically populated properties in the Layout Designer.
    Use the Form page to add, remove, or rearrange fields on the form. The Table page includes the same functionality for the table under the form.

    > 💡 **Tip:**
    >
    > From the Form or Table page of the Layout Designer, right-click a field or column to see choices for changing the order.

    To modify a form field or table column, select it from the table and click the Edit button (✏) to open the Business Object Field Editor. From here, you can add a tooltip or a list of values to a field. To add a tooltip, click the **General** tab and type help text in the **Tooltip** field. The tooltip is displayed when the user selects a form field label or table column header. To create a list of values, see Configure a List of Values.

13. Configure 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.

Configuration is now complete and you can publish the workbook. At this point, 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*.

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.
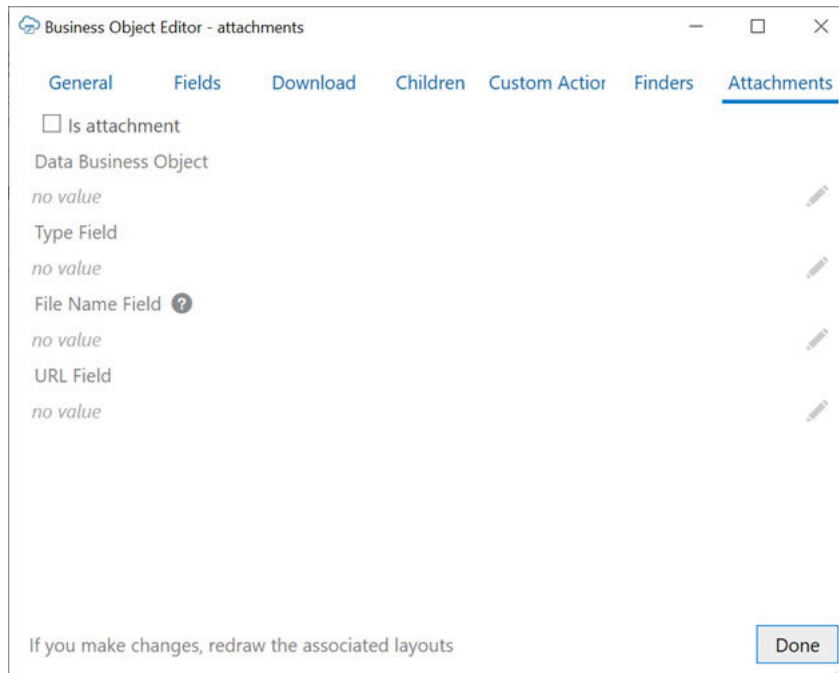
2. Select **Is attachment**.

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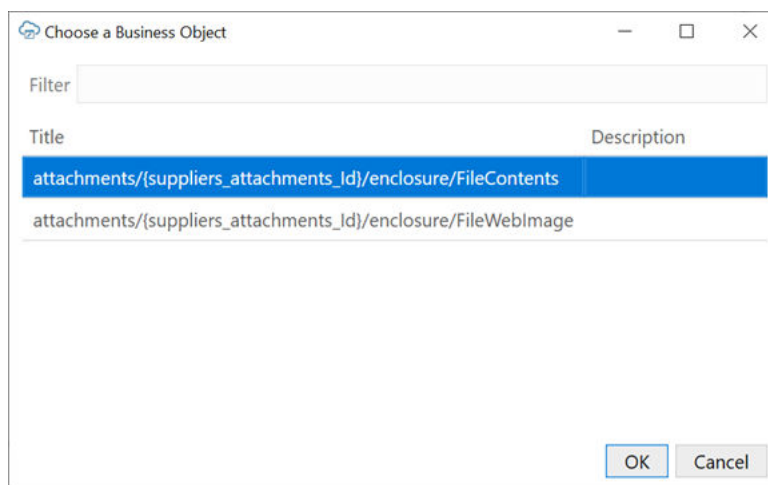
The child business object must support sending and receiving attachment data. This means that the child endpoint has a collection path that supports:

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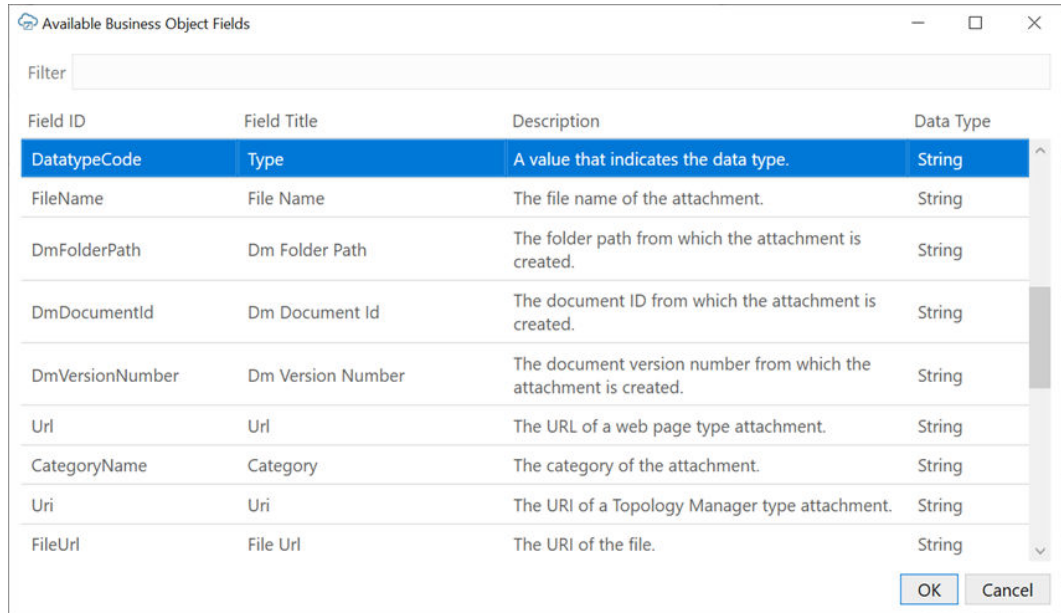
This field must be a string field with valid values of `TEXT` (text type attachment), `FILE` (file type attachment), and `WEB_PAGE` (web url-based attachment). Typically, the field ID for this field is `DatatypeCode`.

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*.

B	C	D	E	F	G	H	I
Status	Type	File Name	Url	Title	Description	Local File Path	Key
	TEXT	attachment.txt		Text attachment	Text attachment		
	WEB_PAGE		https://link/to/document.html	Web page attachment	Web page attachment		
	FILE	receipt.pdf		PDF attachment	PDF receipt attachment		

Attachment
Type Text
File name attachment.txt
Local file 📥 📤

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.

Use Polymorphic Business Objects and Fields

Oracle Visual Builder Add-in for Excel supports layouts for top-level polymorphic business objects as well as child business objects with a one-to-many relationship with its parent. For child objects in a one-to-one relationship, add descriptive flexfields (DFF) from the child to the parent layout.

Topics:

- About Polymorphic Business Objects
- Use a Polymorphic Business Object in a Layout
- Add Descriptive Flexfields to a Layout
- Refresh Polymorphic Business Object Metadata
- Polymorphic Support Limitations

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) are a type of polymorphic business object.

For example, an "Employees" REST resource may include a child polymorphic "Regional Information" resource 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.

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 the polymorphic business object is:

- A top-level business object or it has a one-to-many relationship with its parent business object, it can be used directly in a layout. See Use a Polymorphic Business Object in a Layout.

- In a one-to-one relationship with its parent business object, it can be added to an existing layout bound to its parent business object. See Add Descriptive Flexfields to a Layout.

Use a Polymorphic Business Object in a Layout

You can create Table or Form-over-Table layouts for top-level polymorphic business objects as well as child business objects with a one-to-many relationship with their parent business object.

Before you begin, you can verify that a child business object is in a one-to-many relationship with its parent by checking the "cardinality" setting for a child business object. If this setting isn't set to "Many", the child polymorphic business object won't appear in the New Layout Setup wizard when you try to create a layout.

> **✏️ Note:**
>
> For child polymorphic business objects in a one-to-one relationship, add descriptive flexfields (DFF) from the child business object to a Table or Form-over-Table layout bound to the parent. See Add Descriptive Flexfields to a Layout.

1. 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.
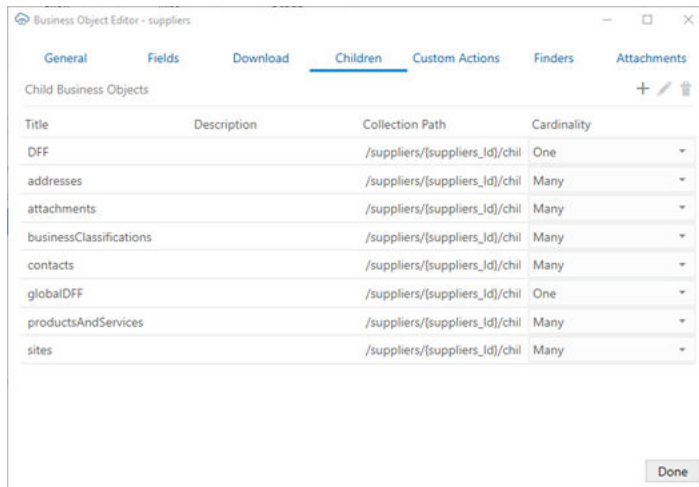
Cardinality options are "One", "Many", or "Unknown". Only child business objects with a cardinality of "Many" (one-to-many relationship) can be used in a layout.

> **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.

2. Create a layout for a polymorphic business object as you do for other business objects. See Create Layouts in an Excel Workbook.

3. If no polymorphic fields appear in the resulting layout, add the discriminator field to the layout from the Layout Designer.

   See Add Descriptive Flexfields to a Layout for the steps.

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 Use a Polymorphic Business Object in a Layout 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.
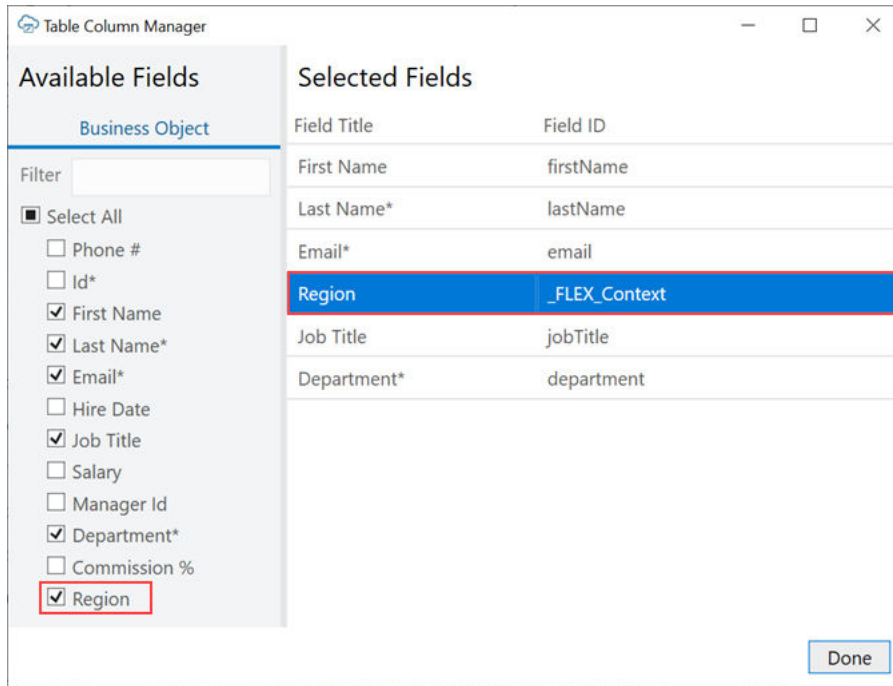
You can also change the order of fields in the form or table by dragging and dropping fields in the Selected Fields list.

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).

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.

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.

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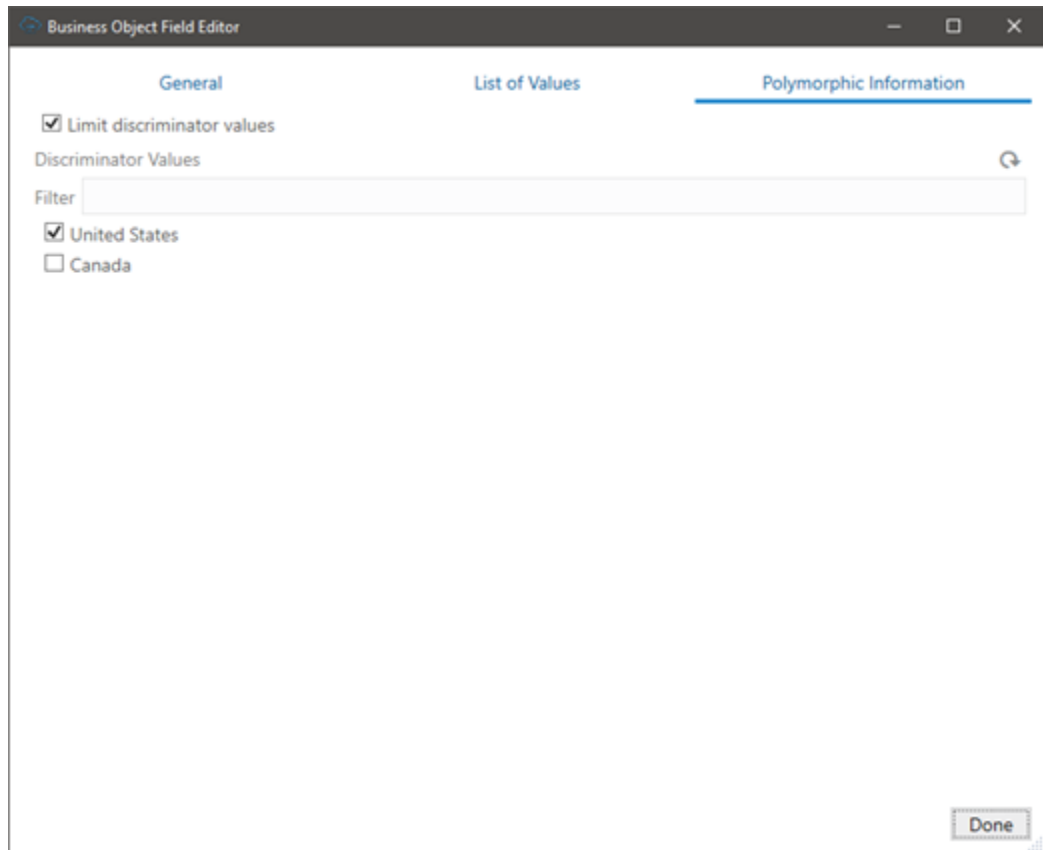
8. After you select or deselect discriminator values, close the open editors.

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

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.

Polymorphic Support Limitations

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

- A polymorphic REST resource 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 resource to a layout, it must expose polymorphic business object metadata. In an OpenApi3 describe document, this means that it must contain a "discriminator" and "oneOf" syntax in the schema for the business object. "anyOf" polymorphic business object syntax is not supported.

- When a layout contains a polymorphic resource, the "fields" and "expand" query parameters should not be manually configured in the "Search Parameters" in the "Query" tab of the designer.

- The add-in assumes that polymorphic fields appear in the metadata in the following order: global segments, discriminator, context-sensitive segments. If that is not the case, the add-in may fail to correctly determine which segments are global and which are context-sensitive.

- 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

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:
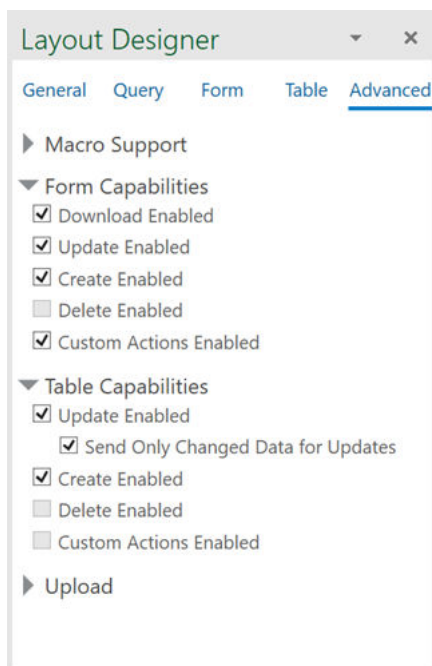
3. Select or deselect options as required. If the business object doesn't support an action, it won't be available for selection (like **Custom Actions Enabled** in the example). See REST Operations for more information about the REST support required for these options.

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 description 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 tooltip help 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:
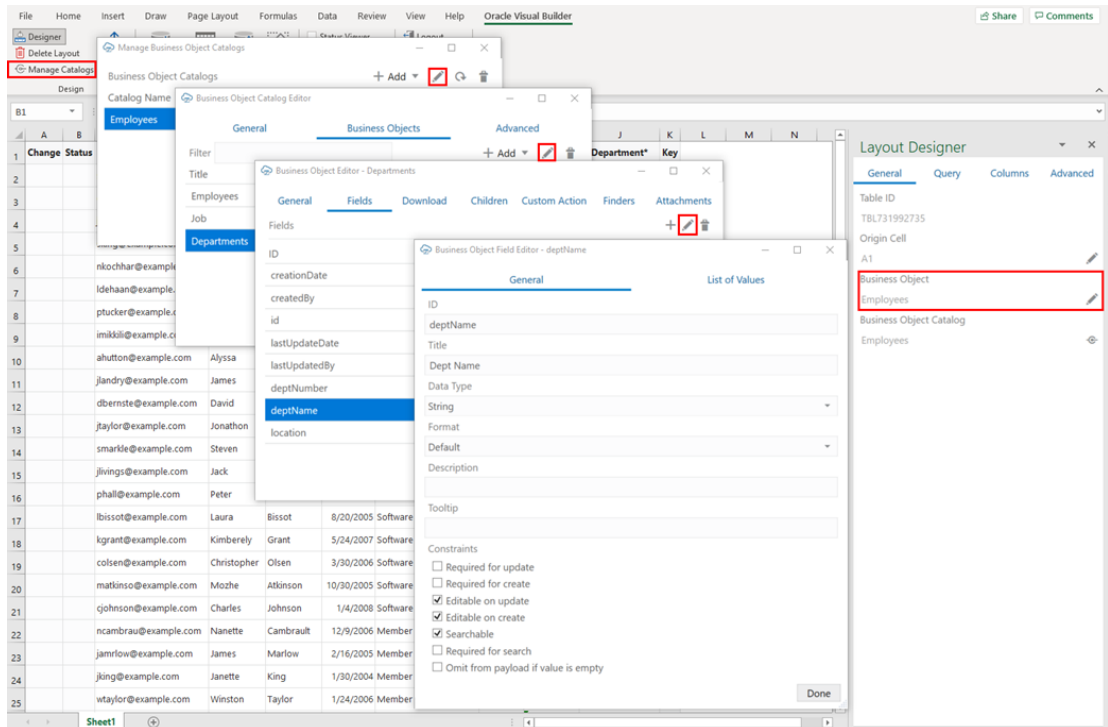
The add-in includes these editors:

- Manage Business Object Catalogs: Let's you import, open, and refresh a catalog

- Business Object Catalog Editor: Let's you 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: Let's you override the base path of the object, view service description path details, configure pagination, and configure row finders
- Business Object Field Editor: Let's you edit field titles, change when a field is editable, adjust the field data types (Advanced), and configure tooltips. You can also configure the list of values for a field.

Add a Business Object to an Existing Catalog

If your catalog is missing a business object, you can add a new business object description to the catalog using the Business Object Catalog Editor. The new business object will be located at the same host/base path as the other business objects in that 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 a Business Object from a Service Description** from the menu.
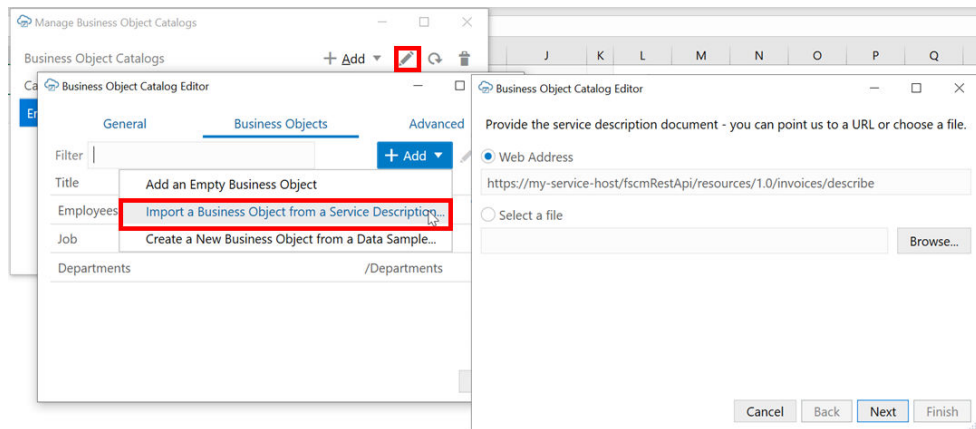
5. Provide the URL or a file that contains the OpenAPI description of the new business object and click **Next**.

6. Review the details of the new business object, then click **Finish**.

7. Click **Done** first in the **Business Object Catalog Editor**, then in the **Manage Business Object Catalogs** window.

> **Note:**
>
> If Oracle Visual Builder Add-in for Excel finds more than one business object in the service description you provide, only the first business object is added to the selected catalog. All other business objects are ignored.

Import a Business Object Catalog

Although most integrated workbooks will only ever need a single catalog, you can import another business object catalog by providing Oracle Visual Builder Add-in for Excel the catalog's service description document.

> ⚠ **Caution:**
>
> If you want to add an additional business object to your workbook, you should generally add it to your existing catalog. See Add a Business Object to an Existing Catalog for the steps.
>
> If you need to use a different service at a different host or framework type, you can create an additional catalog instead by following the steps in this topic.

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 Description** from the menu.
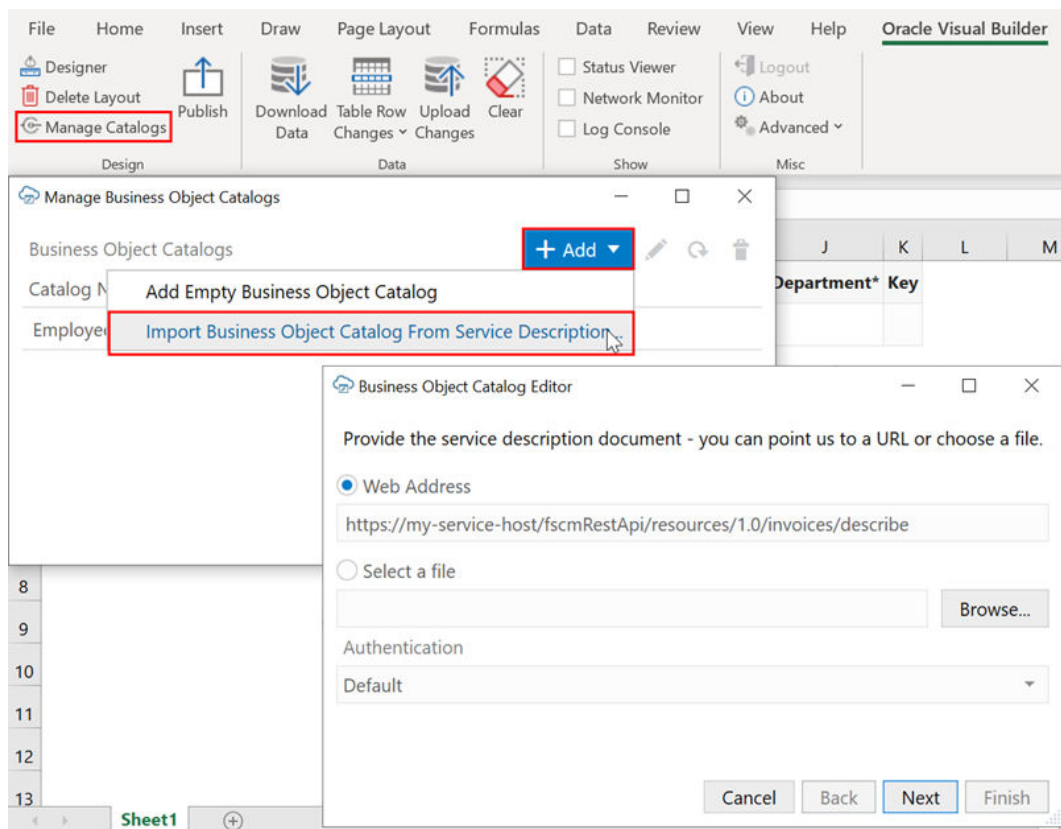
3. Follow the instructions in the wizard to import the catalog.

After you import a catalog into the Excel workbook, consider editing it in the Business Object Catalog Editor so that the catalog has a descriptive name. 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`.
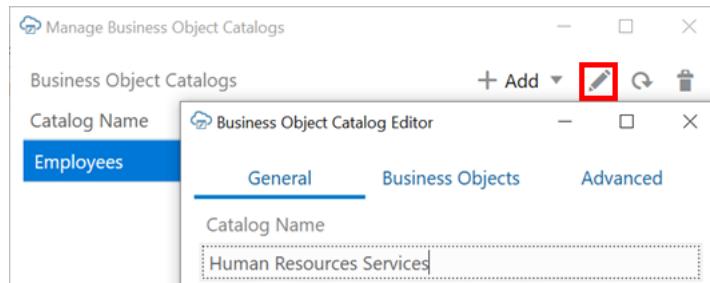
> **Note:**
>
> Certain OpenAPI document properties, such as Description, can contain formatting hints. The add-in displays the description text as is, with no interpretation of such hints.

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 description.

If you have the host, base path, and resource name for the service description, 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**.
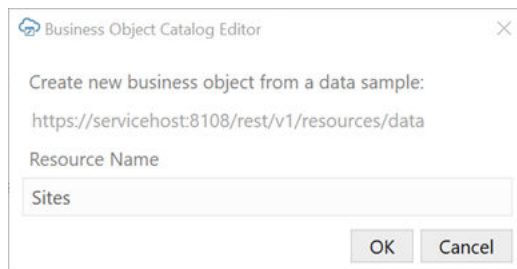
5. Review the business object definition to ensure it matches the service.
   Here are some things to watch for:

   • 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 Field 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

• Adding a tooltip to help your business users understand the purpose of the field

• 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 description document 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 value used by the add-in as the form field label or column header in a layout. Update the title as required to make the label or header more understandable to your audience.

Setting	Description
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` `String`
	⚠️ **WARNING:** Do not modify the data type without consulting the service owner.
Formatting	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 renders "7/15/2022" as "Friday, July 15, 2022".
	See Choose Field Formats.
Description	A description for the field.
	Add an optional description for the field. This value is only shown in the Business Object Field Editor.
Tooltip	Help text for the field.
	This text is displayed in a popup when the user selects the form field label or table column header.

Setting	Description
Required for update and Required for create	Enforces data validation on the field during create or update: • If 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 Understanding Data Validation in *Managing Data Using Oracle Visual Builder Add-in for Excel*. • If unselected, the add-in doesn't perform data validation on these field's cells on create or update.

> ⚠️ **WARNING:**
>
> Don't deselect these check boxes without first consulting with the REST service owner. These settings come from the service description document and reflect requirements of the REST service.
>
> If you deselect these check boxes, the add-in attempts to upload the new or changed row. If the field is required and no value is provided, the service returns a "(400) Bad Request" error.
>
> See Required Fields.

Setting	Description
Editable on update and Editable on create	Allows or prevents write operations on the field during create or update: • If selected, the business user can provide a value during create or update. • If unselected, the field cells are set to read-only. See Understanding Read-Only Behavior in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Setting	Description
Searchable	Determines if a field can be used in a search.
	If selected, the field is available when creating a search for the workbook.
	 ⚠️ **WARNING:** The default value depends on the service description document. Make sure that the service supports searching on the field before selecting this check box.
Required for search	Determines if a value is required for the field during a search: • If selected, a value must be provided for the field on download. Required fields are indicated by an asterisk (*) in the Search Editor. • If unselected, no value is required. If none is provided, the add-in ignores the field when retrieving data. See Use Search to Limit Downloaded Data.
Omit from payload if value is empty	Determines if the field is omitted from the payload if the value is null.
	If selected, null values are omitted. See Omit Empty Values During Upload.

5. 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.

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 supports five authentication methods: 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.

- **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. For example, Oracle Fusion Applications Token Relay is not supported with Oracle REST Data Services (ORDS). If you pick token relay for an ORDS catalog, authentication fails.

5. If you selected OAuth 2.0 Authorization Code (PKCE), click **Edit Authentication Flow Properties** and enter the required authentication properties.
   You can also use the **Import** button to upload a JSON file containing the authentication properties. See OAuth 2.0 Authorization Code Flow.

Override a Business Object's Base Path

You can configure a business object to use a specific base path. 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, `/abcRestApi-context-root/resources/v1`). But if a business object needs to use a different base path (for example, your list of values come from `/123RestApi-context-root/resources/v1`), 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 a Business Object 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 **Referenced Business Object** for a list of values if required. See Configure a List of Values.

Manage Metadata Path Information

When you provide a service description document, Oracle Visual Builder Add-in for Excel captures the path to the service description 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**. It also captures path information such as REST API methods and parameters and displays these details in the **Path Editor**.

View these settings by clicking the **Edit** button next to the **Metadata Path** field.
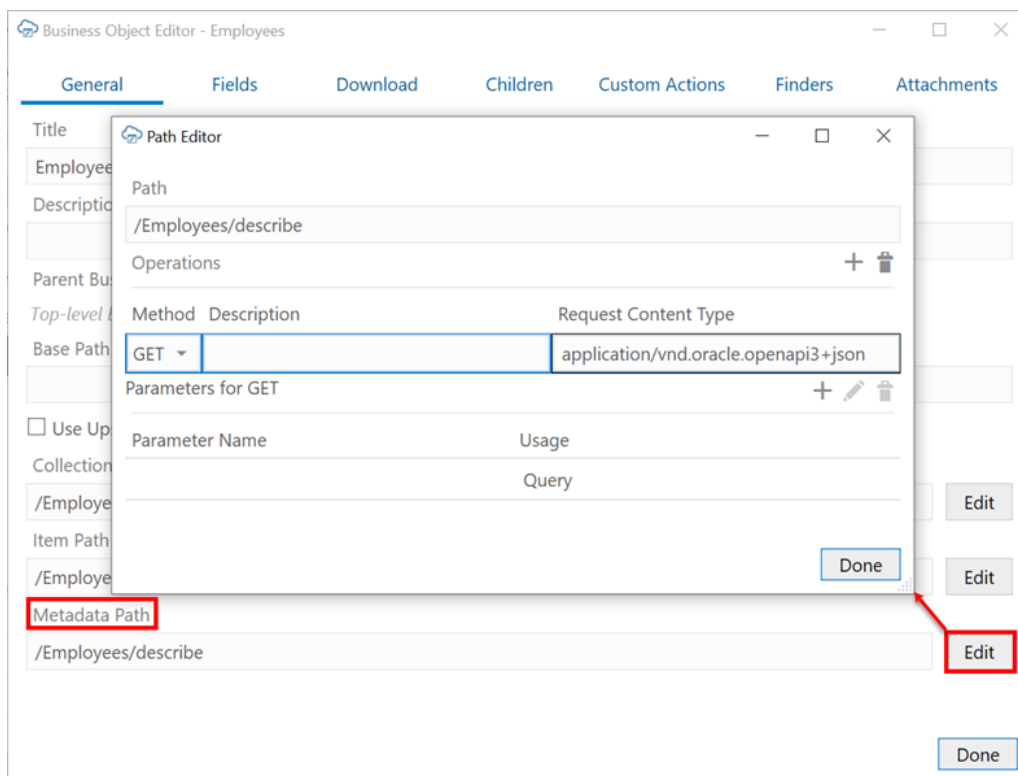
> ⚠️ **Caution:**
>
> The metadata path should include a GET method that specifies the correct request content type that the service expects for a metadata request. If you need to modify these settings, be sure to provide values that will result in the service returning a proper OpenApi version 3.x service description.
>
> Consult the REST API owner for the required values.

Configure Pagination for a Business Object

If the REST service supports pagination, you can download pages of rows.

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 a Oracle business object REST API 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.

Configure Row Finders for a Business Object

Row finders are predefined filters available through Oracle business object REST API services that allow you to download a specific subset of records. If the service owner has included row finders with the service, you can view and configure them through the Finders tab of the Business Object Editor.

For example, if your service includes a row finder that filters expense reports by "unapproved" status and assigned to the "current user", you can use the row finder to download only those expense reports that require your approval.

Row finders are defined through the service. You can't add or configure them through Oracle Visual Builder Add-in for Excel. However, you can configure how they appear in the add-in. For example, you can remove unwanted row finder variables so that they don't appear in the prompt at download time. You may also want to give row finders or row finder variables a more readable title.

> **Note:**
>
> Any changes you make must be compatible with the service. If your changes are incompatible, you will likely see errors during a download.

1. In the Excel ribbon, click **Designer**.

2. In the Layout Designer, click the Edit icon next to the Business Object field.

3. Click the **Finders** tab in the Business Object Editor to view a list of available row finders. Note that the Finders tab shows only for Oracle business object REST API services (not ORDS or other REST services).

4. Remove row finders if required. You can also edit a row finder's details, like changing the title to be more readable:
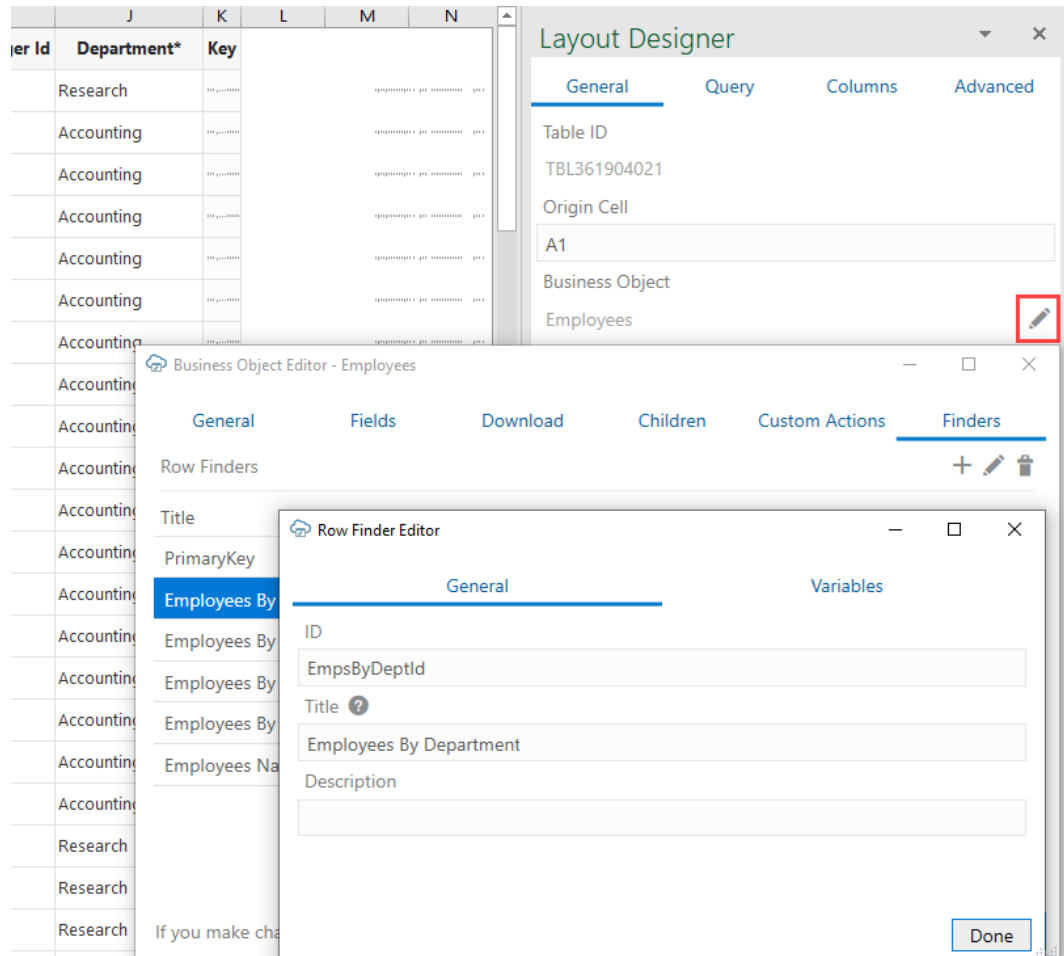
When editing a row finder, take care to not change the row finder's ID.

5. If a row finder supports variables that act as arguments or parameters for the finder, click the **Variables** tab to update or delete variables. For example, you can change the row finder variable's title (but not the ID) or set the variable as required to force the user to provide a value at download time, as shown here:
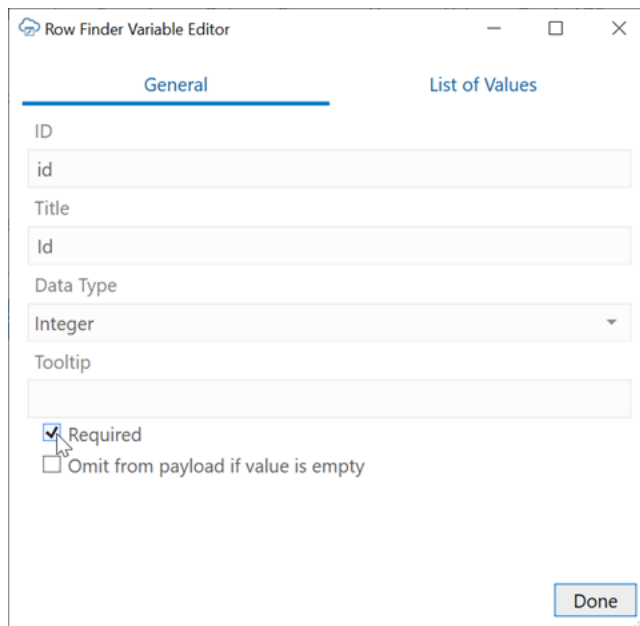
6. If required, configure a list of values for a row finder variable from the List of Values page in the Row Finder Variable Editor.

> **Note:**
>
> A list of values is a drop-down list populated with values from a referenced business object. Filters can also be defined to further limit the values to those required by the business user. These filters can contain expressions that are resolved at runtime. Some expressions like `{ this.BusinessObject.Fields['FieldId'].Value }` can't be used in filters for row finder variables. See Configure a List of Values and About Expressions.

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. For details on how your configuration takes effect, see Download Behavior in Layouts that Use Search and Row Finders.

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 Oracle business object REST API services.

Refresh a Business Object Catalog

If a service owner makes significant changes to the service description–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 a service description 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.

Your catalog might be based on a single service description 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 description documents. When you refresh a catalog, Oracle Visual Builder Add-in for Excel reads the metadata path for each business object and issues a request that returns the business object details from the service description document.

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

> ⚠️ **Caution:**
>
> A refresh overwrites any changes that you've made locally to the business object catalog. If you've spent a lot of time customizing the business object (especially field titles), consider whether it's worth redoing those changes; it might be simpler to manually add that new field or custom action. Before refreshing a catalog, back up the workbook in case you want to revert your changes.

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 Refresh Business Object Catalog icon (⟳).

4. 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.
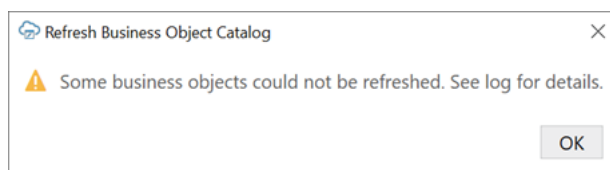
To troubleshoot the error, open the log console and repeat the process to discover the cause. See Logging Console.

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

5. 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 not 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. No existing items are removed.

• When an existing item is refreshed, the item's new properties replace the older properties. 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.

• A refresh does not change any field formats, as those are always set manually.

If you want a new field in your layout, use the Table Column Manager to add the field after refreshing the catalog:

1. In the Layout Designer, click **Columns**.

2. Click Manage Columns (✛) and select the new field. Click **Done**.

Configure the REST-Framework-Version

You can change the value of the REST-Framework-Version request header, set to version 6 by default for Oracle business object REST API services. The REST framework version determines functionality for accessing business objects.

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** field. This field is available only for Oracle business object REST API 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
Graphical Search Editor	Oracle business object REST API and Oracle REST Data 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	Oracle business object REST API	• Provides access to powerful, complicated searches defined by the service • Allows the business user to provide values	Only available for Oracle business object REST API services
Search 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 Search to Limit Downloaded Data

For workbooks that integrate with Oracle business object REST API services and Oracle REST Data Services, you can configure each layout of the workbook to enable a user to specify search values that limit 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 create a search for a layout by setting one or more search conditions. A search condition consists of a business object field, a query operator, and a value. A query 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.

The add-in supports lists of values (LOV), including ones with dynamic filters, in the Search Editor. 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.

Note that there are two business object field settings in the Business Object Field Editor related to search:

- **Searchable**: If **Searchable** is selected for a field, the field is available for you to choose from. You can adjust which fields show by selecting or deselecting **Searchable** in the Business Object Field Editor. Before you set a field as searchable, make sure that the service supports searching on that field.

- **Required for Search**: If you want to require your business users to provide a value for a field when downloading data, select **Required for Search** for the field in the Business Object Field Editor. Required fields are indicated by an asterisk (*) in the Search Editor.
  If a field is required, the business user must provide a value before proceeding. If a field isn't required and the business user leaves it blank, the corresponding condition is omitted from the search.

See Configure Business Object Fields for more information.

To create a search query for a layout:

1. Select the layout you want to create the search for, then click **Designer** from the Excel ribbon.

2. In the Layout Designer's Query 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.

3. To configure a search condition:

   a. From the **Available Business Object Fields** window, select the business object field that you want to enable users to enter search terms for. If this window is not open, click **Add field** from the **Search Editor** first.
   For example, select **Department Name** if you want to enable users to search on employees by department, as shown here:
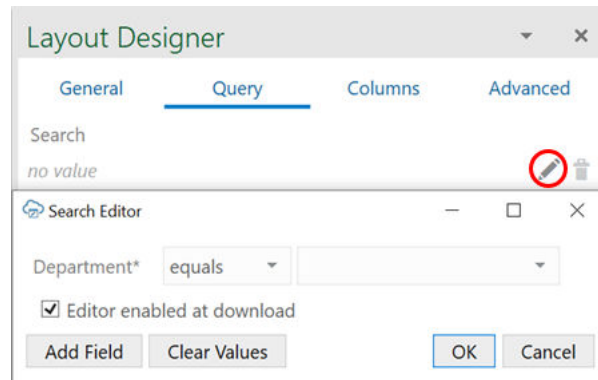
b. Select a query operator from the list beside the field name. For example, select "equals" to return employees from a department that matches the value in the value field or "not equals" to return only employees from other departments.
The available operators depend on the data type. For example, string fields have filters such as "starts with", "contains", and "ends with".

c. 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.

4. 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:
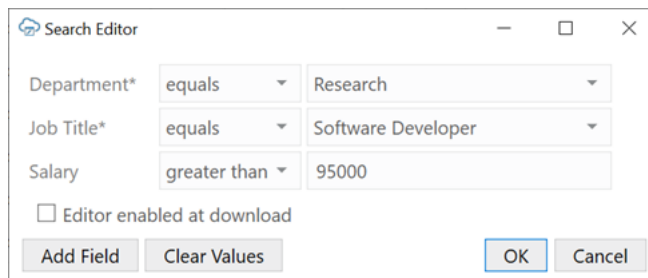
> **Note:**
>
> When configuring your search, make sure the field your dynamic list of values is based on is added before the field with the dynamic list. So, to filter the Job Title list based on the Department value, add the Department field before the Job Title field in the Search Editor. See Configure a Filter with a Dynamic Parameter.

5. 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:
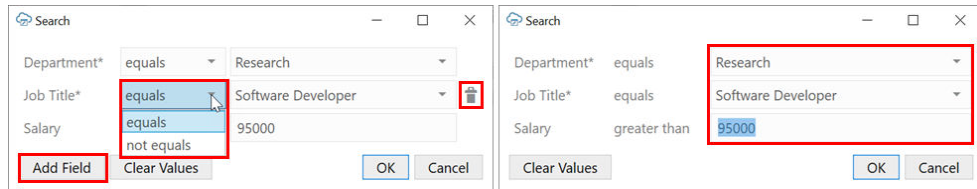
If the check box is unselected, only the values are editable, as shown on the right.

6. Click **OK**.

Use Row Finders to Limit Downloaded Data

For workbooks that integrate with Oracle business object REST API 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 Query 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:
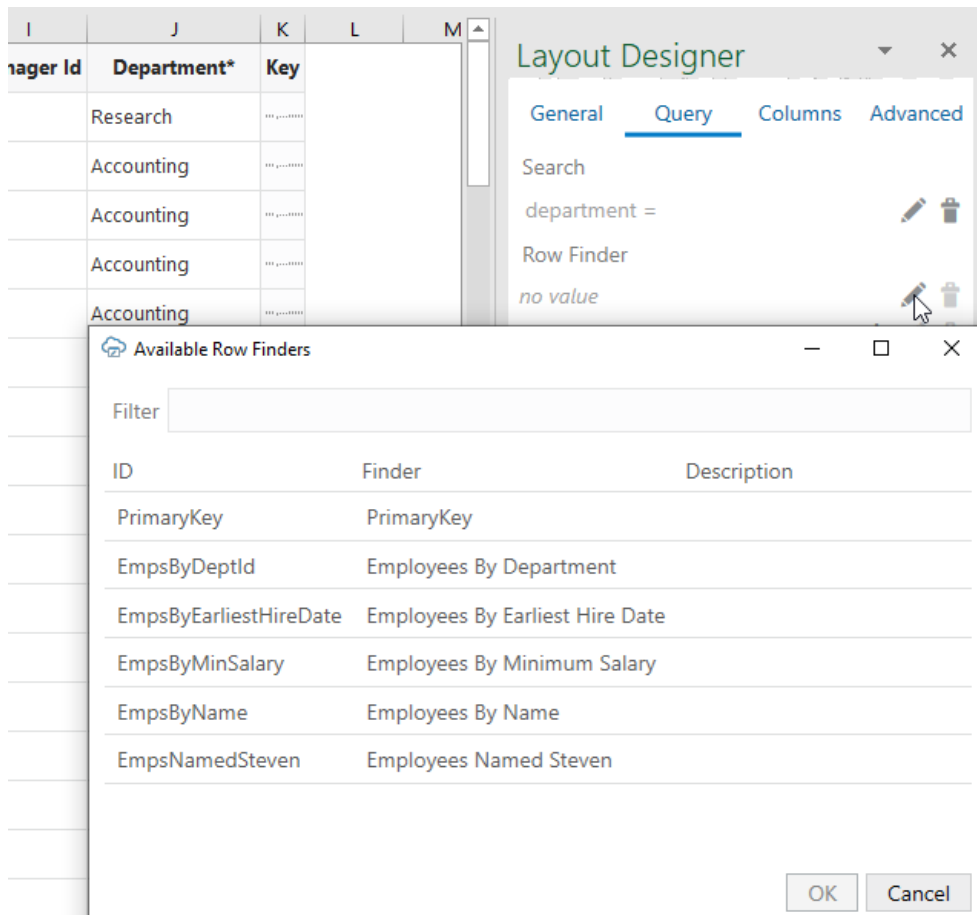
For details on modifying how row finders appear in the add-in, see Configure Row Finders for a Business Object.

3. Select a row finder and click **OK**.
   When you download data to your workbook, the row finder filters the data based on the filter criteria. If the row finder requires input from the user, you'll be prompted to provide a value for the row finder parameter.

For more information about configuring the Oracle business object REST API 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

Use Search Parameters to Limit Downloaded Data

Create a search query that determines which data get returned from the REST service when a business user invokes a download. For example, if a user only needs to access records for a specific location, you can create a query that includes a search parameter for that location.

For example, to retrieve employee items where the managerId field is empty, you would create a search on the **Employees** business object with a search 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

When the user clicks the **Download Data** button, the add-in appends the search query to the REST endpoint URL of the GET request. 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.

> **Note:**
>
> The Search Parameters property works in combination with the other two properties (where applicable). They are not mutually exclusive. However, some combinations may work where others may not. If you choose to configure multiple search options, you must ensure that the service supports that combination.
>
> For example, some services such as Oracle business object REST API services do not support using more than one "q" parameter in a REST call. If you define a "q" search parameter for a layout, it may not work if you already have a search defined using the Search Editor.

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)))
```

Each service defines which parameters can be used for search. Likewise, each service defines the required and supported syntax for the expression that appears on the right-hand side of the assignment operator (=).

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.

The add-in applies URL encoding to the parameter value at download time. Don't enter URL-encoded values. The search parameter name is not encoded.

> **Note:**
>
> You can create a separate query for each layout in a set of dependent layouts. See Create Search and Search Parameter Queries for Dependent Layouts.

To add search parameters to your layout:

1. In the Excel ribbon, click **Designer**.

2. In the Layout Designer's Query tab, click the Add or Edit Search Parameter icon next to the Search Parameters property to open the Search Parameter Editor where you add or edit search parameters.
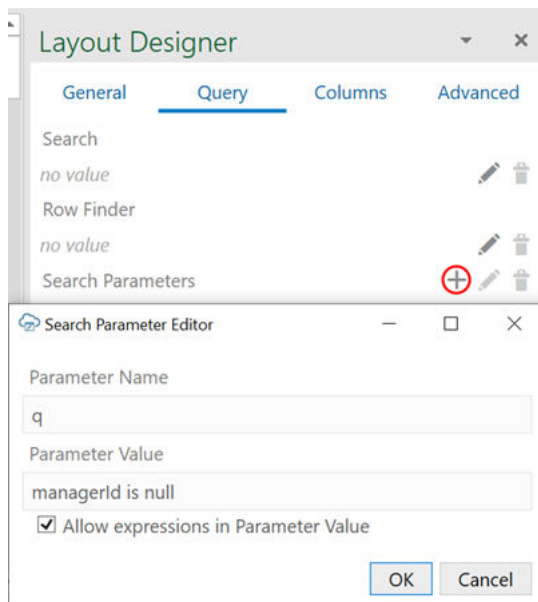
> **Note:**
>
> If you include an expression in the **Parameter Value** field, select **Allow expressions in Parameter Value** to ensure Oracle Visual Builder Add-in for Excel evaluates the value as an expression. See About Expressions.

3. Click **OK** to close the editor.

> **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.
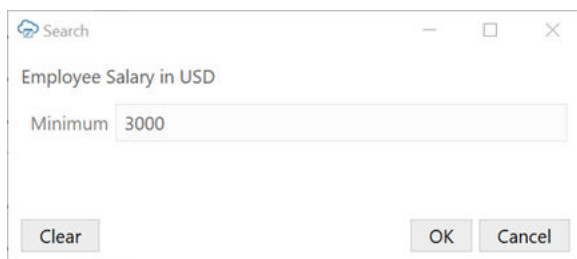
Download Behavior in Layouts that Use Search and Row Finders

If you configured a value for the Row Finder property, the behavior of the add-in depends on the configuration of the finder exposed by the REST service:

*   If there are no parameters, there is no prompt.

*   If there is exactly one parameter, a prompt appears that allows the user to specify a value for the parameter.

*   If the finder has more than one parameter, a prompt appears that allows the user to specify values for each parameter.

> **Note:**
>
> Only Oracle business object REST API services support row finders.

If you added search fields to the Search property, the add-in prompts users who download employee data using the **Download Data** button to enter values in the search field, as shown here:
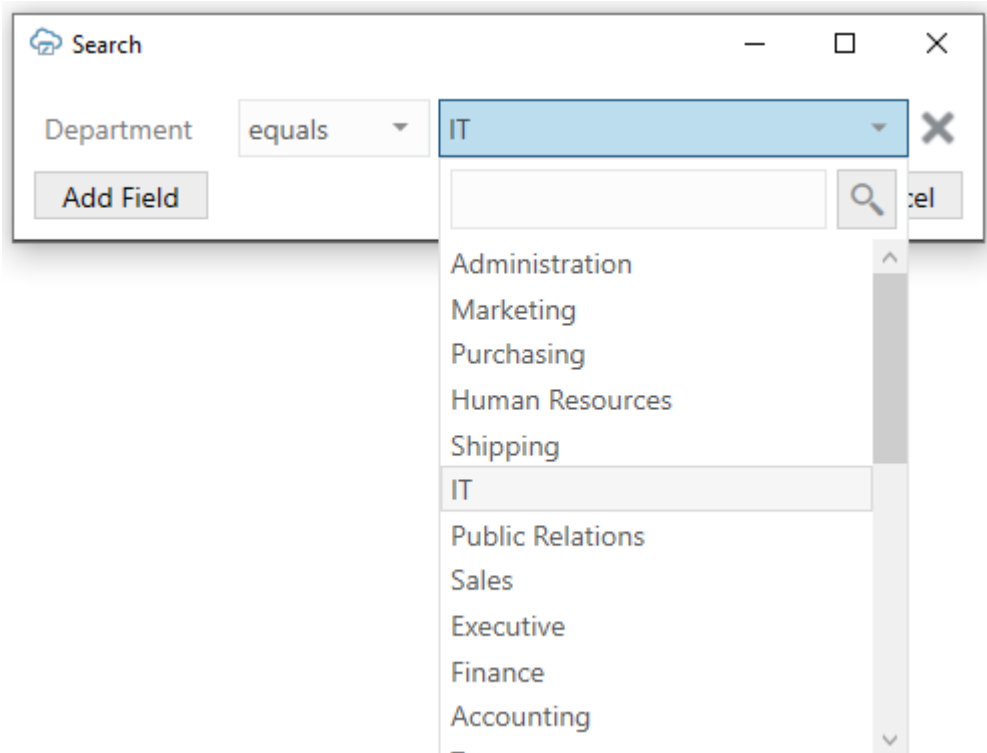
7
Custom Actions

Oracle Visual Builder Add-in for Excel supports custom actions defined by Oracle business object REST API services on an entire business object or on individual items (rows) from a business object.

A custom action is a function defined by the REST service that is callable by external systems using REST POST messages. 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. When that value is a string or other scalar value, the add-in shows the result to the business user.

What a custom action does when invoked by the add-in is determined by the service. The add-in simply sends a POST on the collection or item path that includes the action name, such as `/contextRoot/v1/myBusObj/{itemId}/action/doMyAction` for a business object item.

This post can include payload field values, known as "parameters", to the service. The service then takes the passed parameters and performs the custom action. This action may act on a single row or on all rows in the business object. It might also run a query and act on a subset of the business object.

Let's look at a couple of examples of custom actions at the business object (BO) level and BO item level. For example, a REST service may have a custom action defined at the BO level such as the one described here in *REST API for Oracle Project Portfolio Management Cloud*. When triggered, this custom action reprocesses all the project budget versions which are showing as failed and brings these back into a working state.

Examples of custom actions at the BO item level might be a couple of custom actions, "Approve" and "Reject", that operate on the items of a expense report business object. A business user can download one or more expense reports, approve or reject each one, and then update them all in a single upload.

If a given REST API supports custom actions, they are described in the service's OpenAPI v3 service description document. The add-in can parse this service description and capture custom action settings in your workbook. No further configuration is required.

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*.

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 Custom Action Limitations for information on other custom action limitations.

Edit Custom Actions

Custom actions exposed by the service that your workbook uses can be viewed and edited in the **Custom Actions** tab of the Business Object Editor.

Oracle Visual Builder Add-in for Excel populates this editor with the custom action properties as described in the service's OpenAPI v3 service description document.
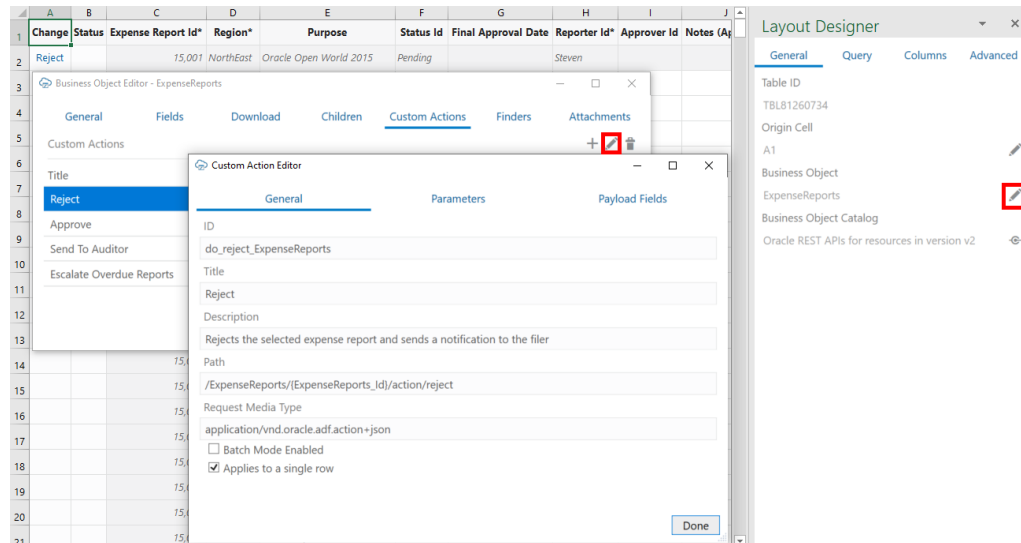
In this example, the add-in shows the properties for a "Reject" custom action on an ExpenseReports business object (BO) item path (`/ExpenseReports/{ExpenseReports_Id}/action/reject`). The **Applies to a single row** check box indicates that the custom action operates at the BO item level. If this check box is unselected, the custom action operates at the BO level.

For information on the **Batch Mode Enabled** check box, refer to Batch Mode for Custom Actions.

Properties such as **Title** (which appears in the UI that the business user sees) and **Description** can be edited as needed in the Custom Action Editor. Other properties reflect the service configuration and should be left as is.

Payload fields can be edited on the **Payload Fields** tab of the Custom Action Editor. Properties such as the **Title**, **Data Type**, and **Required** can be changed. For example, you can change the data type of a payload field from **String** to **Integer**, as long as the change is compatible with the service. You can also update the **Required** property to mark the payload field as required or not:
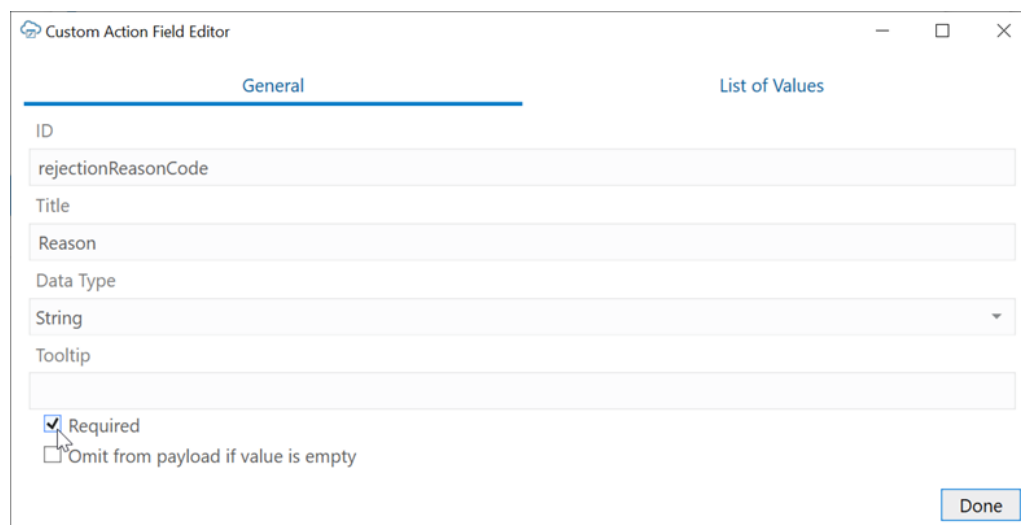
You can also use the **Tooltip** field to add a description of the payload field to help your users determine what information or values are expected. Users can display the tooltip in the worksheet by selecting the column header for a payload field.

You can also define a list of values for the payload field from the **List of Values** tab. See Configure a List of Values.

Add Custom Action Fields

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.
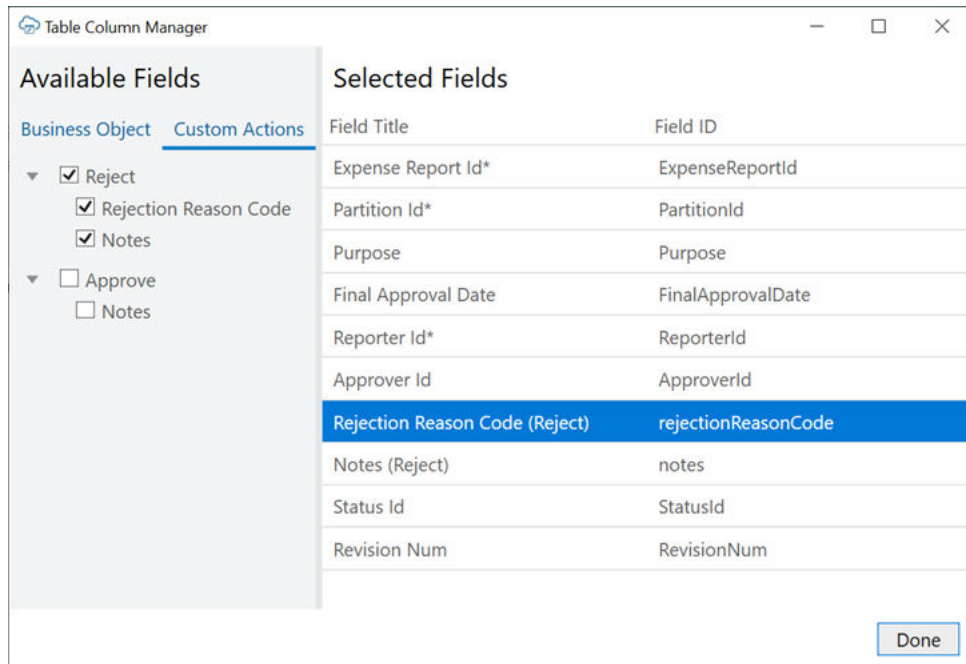
7. Click **Done**.

   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 sends their changes to the REST endpoint and invokes the custom action on the marked rows.

Service Description for Custom Actions

If a given REST API supports custom actions, they are described in the OpenAPI v3 service description document generated by the Oracle business object REST API service. For example, a `reject` 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 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 business object-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.

Batch Mode for Custom Actions

If a custom action supports batch mode, Oracle Visual Builder Add-in for Excel can process custom actions and other types of changes in batches.

Batch 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 (BO) may update multiple items in the BO but is accomplished using a single request.

To enable the add-in to process changes in batches, select the **Batch Mode Enabled** check box in the Custom Action Editor.

If the custom action doesn't work properly in batches, deselect the check box to disable batch processing. If the check box is unselected, the add-in invokes the custom action one row at a time.

> **Note:**
>
> The `"batchEnabled" : false` parameter in the service description only determines the default value of the **Batch Mode Enabled** check box in the Custom Action Editor. If the parameter is not configured, the **Batch Mode Enabled** check box is selected by default.
>
> After the catalog is created, the check box setting controls how the add-in behaves.

Custom Action Limitations

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 3 service description document that have request payload schema members that match business object fields are unlikely to function properly.
- Oracle business object REST API service OpenAPI 3 service descriptions 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 Edit Custom Actions.
- Oracle business object REST API services do not return the updated row. As a result, you need to download before making further changes.
- The results of custom actions that return complex object values for the "result" member in the JSON response payload are not shown in the Status Viewer.

8

Use Lists of Values in an Excel Workbook

You can configure a **list of values** 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 lists of values 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.

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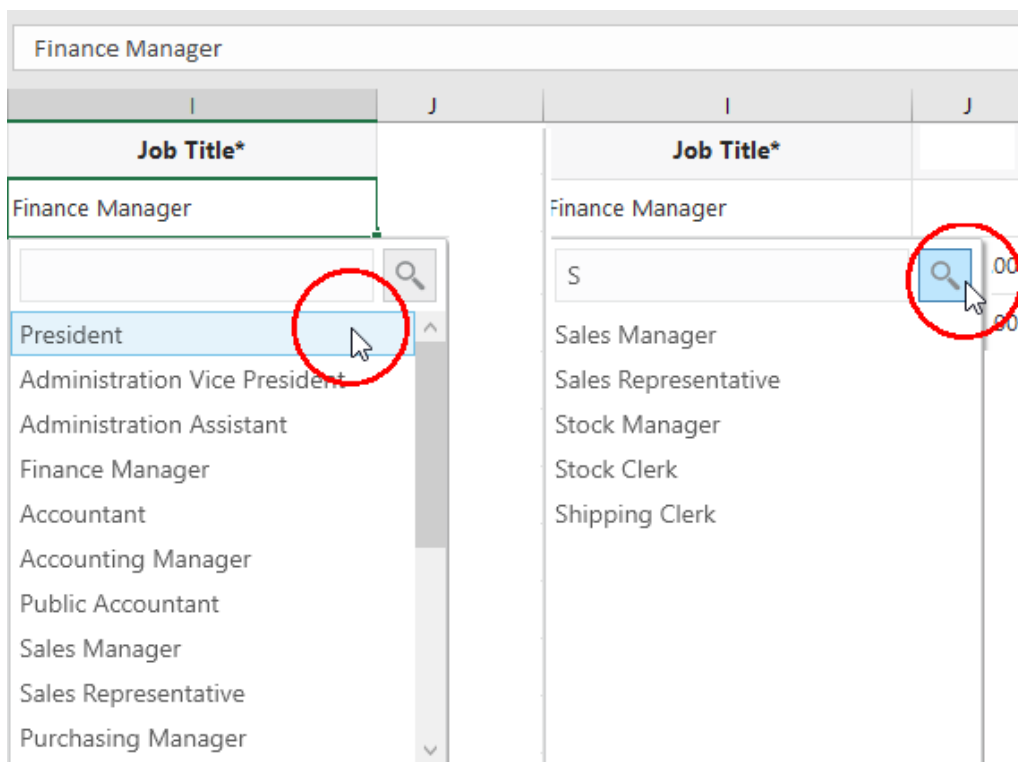
When a business user clicks the **Search** icon, the list of values displays only job titles that begin with s such as "Sales Manager", "Sales Representative", and so on. Business users can also enter the full value in the search box, assuming it's a valid value such as "Sales Manager".

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

Configure a list of values for a business object field, custom action payload field, search field, or a row finder variable.

When you create a list of values, you 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.

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.

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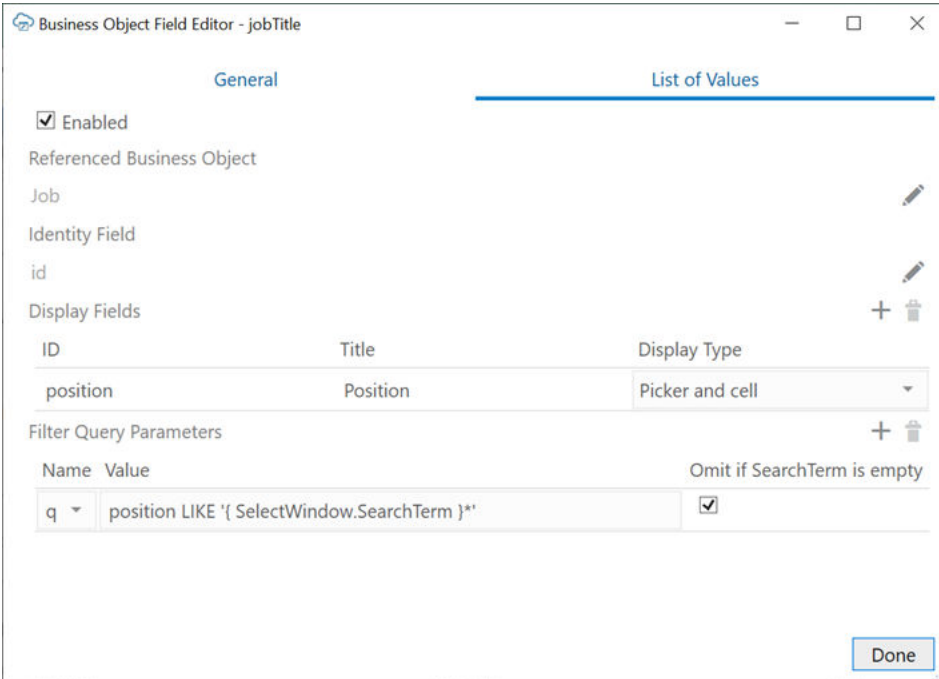
If the catalog is missing the desired business object, you can add a new business object to an existing catalog (see Add a Business Object to an Existing Catalog).

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.

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 **Referenced Business Object** field, then pick an appropriate business object from the same catalog used to create the layout.
   This business object 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 referenced business object.
   This is the field in the referenced business object 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 from the referenced business object.
   These fields come from the referenced business object 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, click **Omit if SearchTerm is empty**.

   Repeat this step to create additional parameters.

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.

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 in the Misc ribbon tab.

Configure a Filter for a List of Values

Configure a filter for your list of values to determine which items from the referenced business object 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.

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. 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 in a Layout.

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 business object REST API 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:

Filter Query Parameters		+ 🗑
Name	Value	Omit if SearchTerm is empty
q ▾	firstName LIKE '{ SelectWindow.SearchTerm }*' OR lastName LIKE '{ SelectWindow.SearchTerm }*'	☑

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:

Filter Query Parameters		+ 🗑
Name	Value	Omit if SearchTerm is empty
finder ▾	LookupTypeFinder;LookupType=I_Status	☐
q ▾	DisplayValue LIKE '{ SelectWindow.SearchTerm }*'	☑
orderBy ▾	DisplayValue	☐

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 or search field. For example, you may want your business users to see only those job titles in a list of values that are relevant for the department that they have selected.

When used in a layout as shown on the left, 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").
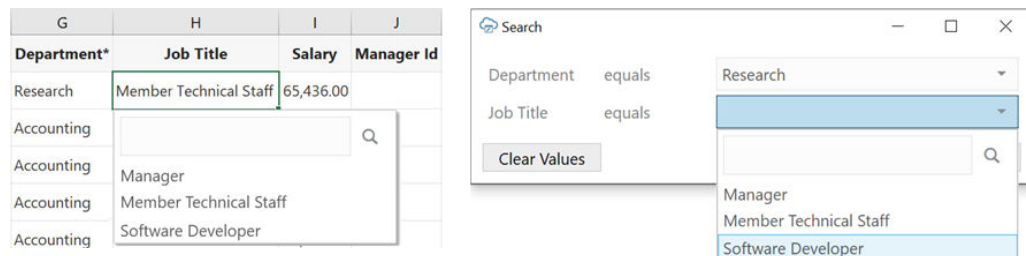
To configure a dynamic filter for the Job Title list, configure a parameter that filters based on the department Id like this:

This parameter includes an expression of 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`.

> **Note:**
>
> 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 Search to Limit Downloaded 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.

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 Objects 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 ancestor column in the current layout, the add-in uses existing 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 Ancestor Columns to a Dependent Layout.

Configure a Cascading List of Values in a Layout

Configure a **cascading list of values** using dynamic query parameters.

In a cascading list of values, the value selected in one list determines the range of values that users can select from subsequent lists. For example, suppose a table displays columns with

lists of values for Countries, States, and Cities. The value that a user chooses in the Countries list determines the values that appear in the list for States, and so on.

Here's how you configure the filter query parameters in the Business Object Field Editor to create a list of values for Countries, States, and Cities. To filter the list of values by the current country:
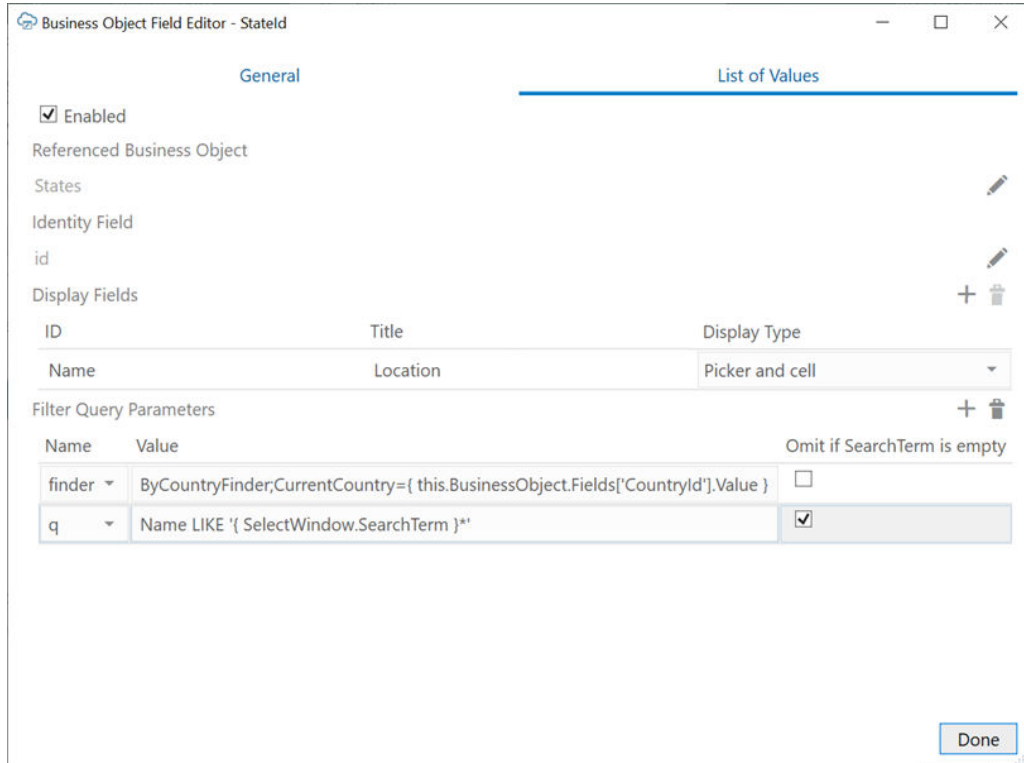
To filter the list of values by Country and State:
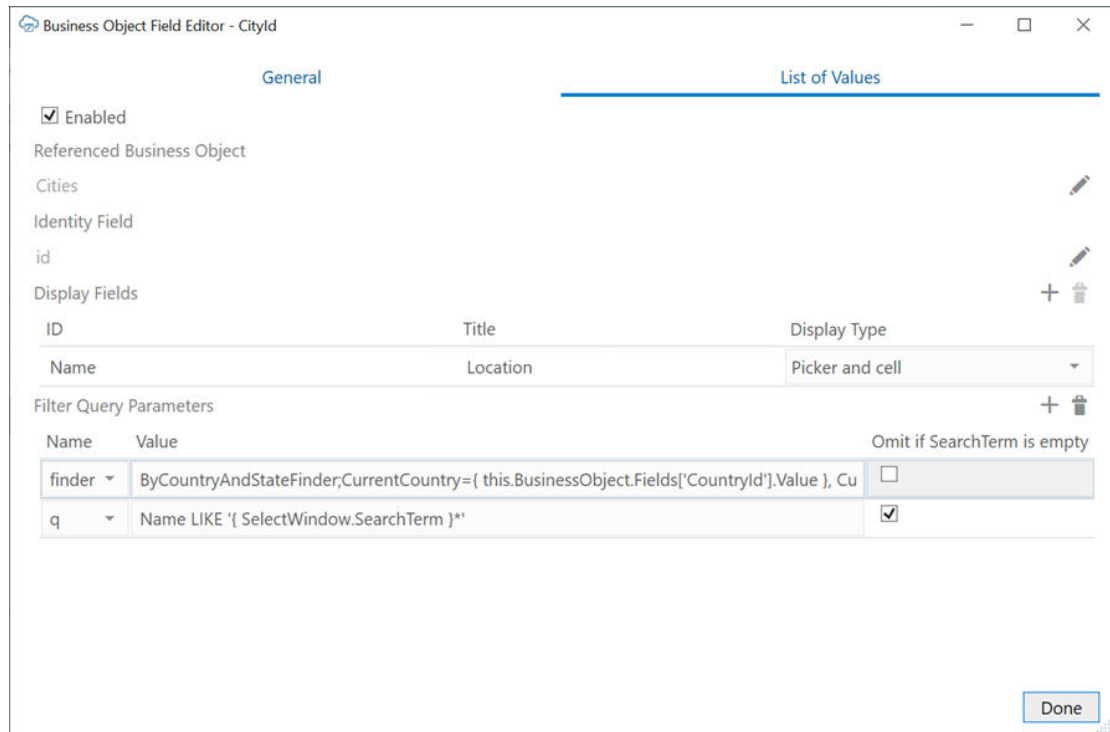
In a scenario like this with three fields ("fieldA", "fieldB", and "fieldC"), remember that fieldC depends on both fieldB and fieldA. So when you create the filter for fieldC's list of values, also include fieldA's value expression. In general, all of fieldB's dependencies must also be fieldC's dependencies.

Note the following name-value parameters under Filter Query Parameters:

Table 8-1   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 }`

If the service has complete metadata for a cascading list of values, the filter query parameters of the corresponding field's list of values is 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 a cascading list of values, you can manually configure a list of values by adding filter query parameters. See Configure a Filter for a List of Values.

Notes on Filters

- 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*.

Filter Query Parameters		+ 🗑
Name	Value	Omit if SearchTerm is empty
q ▾	DepartmentId={ this.BusinessObject.Fields['DepartmentId'].Value }	☐
q ▾	DepartmentId={ this.BusinessObject.Fields['DepartmentId'].Value } AND FirstName LIKE '{ SelectWindow.SearchTerm }*'	☑

List of Values for Descriptive Flexfields

Oracle Visual Builder Add-in for Excel supports lists of values for context-sensitive descriptive flexfields (DFF) with "parameter" and "segment" client binds.

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.

Segment type client binds are supported automatically as long as the service's OpenAPI 3 document provides all the proper metadata. If the metadata is present, the add-in configures the list of values at runtime.

Limitations

- Client binds are not supported on "global" DFF fields.

- Only binds of data type string and number are supported. Binds of date type are not supported.

Configure the Bind Parameters for a Descriptive Flexfield's List of Values

Configure parameter-type client binds used by segments 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. Other details of the list of values for DFFs must come from the metadata and are not configurable.

To configure polymorphic bind parameters (also known as "flex bind variables"), open the **Business Object Field Editor** for a DFF from the Layout Designer, and then configure the bind parameters in the **Polymorphic Bind Parameters** area of the **Polymorphic Information** page.

For example, suppose you want to configure the bind parameter, **BusinessUnit**, for a list of values on a polymorphic field set (Context Value column in the Layout Designer). To configure the polymorphic bind parameters in this case:

1. From the Layout Designer, click the **Columns** tab and then select the Context Value column.

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:
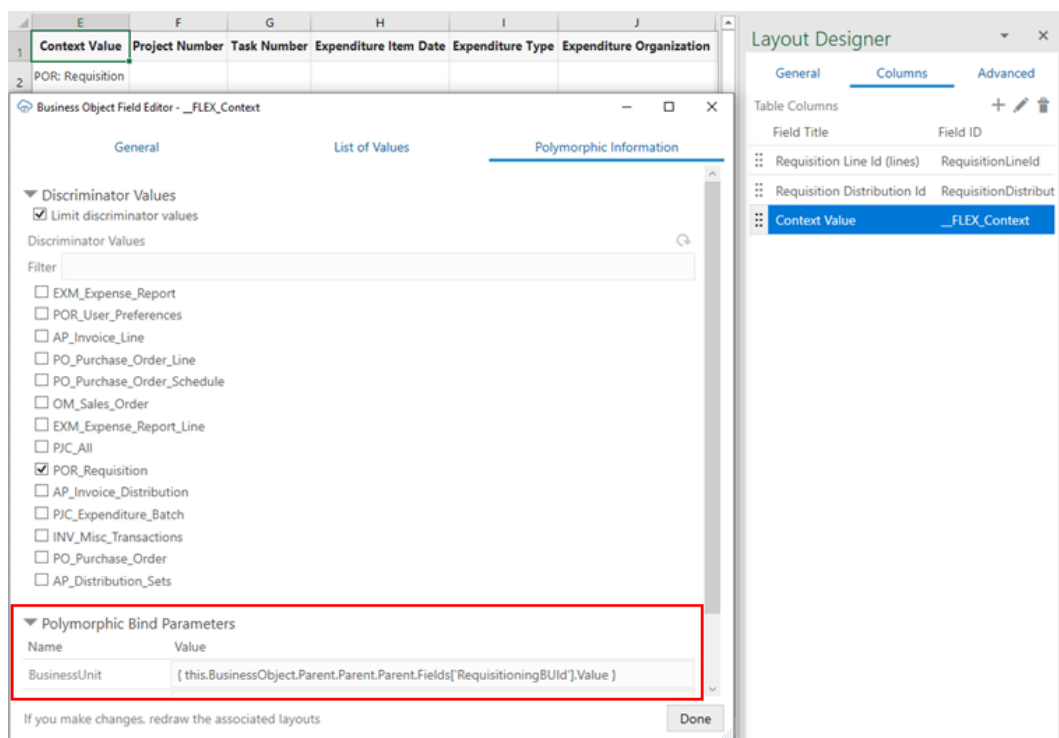
- The value of a bind parameter can be a literal value like `100001`, `Requisition001`, or a string that contains one or more expressions surrounded by curly braces { }

- 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 the 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.

See About Expressions for more information.

Limitations

- This feature only supports lists of values in context-sensitive DFF fields and not "global" DFF fields.

- Only binds of data type "string" and "number" are supported.

- The add-in uses client binds if and only if the OpenAPI 3 document provides proper metadata.

- Expression syntax is limited to what's shown in the example (key parts: *this*, *BusinessObject*, *Parent*, *Fields[<field ID>].Value*).

Clear Cache for a List of Values

A list of values' choices are always cached in the workbook.

The cache contains 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.

Notes and Limitations for Lists of Values

Review the notes and limitations listed here when planning to use lists of values (LOV):

- For Oracle business object REST API services, "row context" list of values is not supported.

- Only one identity field is supported in a list of values.

- Display values support strings; identity values support integers or strings. 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 Oracle business object REST API services, if the catalog includes metadata 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 column in the current layout is configured properly so that the add-in can map a row to its ancestor row.

- Accessing an ancestor field with a dependent LOV—that is, the ancestor field itself uses a field value expression in filter—is not supported.

9

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 Alta UI standards and 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 drop-down.

> **✐ Note:**
>
> The **Reset Styles** command is only available when the designer tools have been enabled.
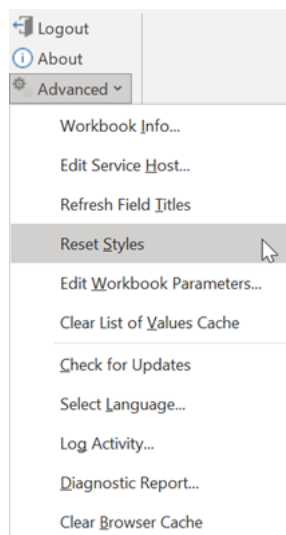
You are prompted to confirm. After confirmation, each style definition used by the add-in is updated. The list of styles includes any style that would be created by the current add-in, including the Normal style.

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.

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 ($\pi = 3.14$) while a French user will see a decimal comma ($\pi = 3,14$).
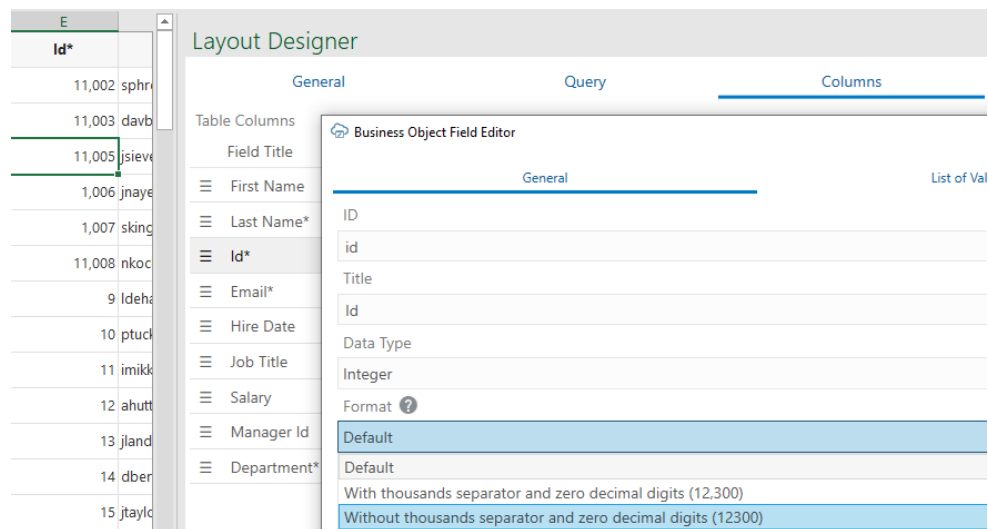
3. Click **Done**.

4. Click **Download Data** to process the change.

Add Tooltips to Your Workbook

You can add tooltips to the form field labels or table column headers of your workbook to provide help to your users. You can also add tooltips to custom action fields.

To add a tooltip, open the Business Object Field Editor for a field and type help text in the Tooltip property on the General tab.

The tooltip is displayed when the user selects the form field label or table column header.

10
Upload Changes

When you are done making changes in an Excel workbook, you are ready to upload the changes to the REST service.

Upload Changes from a Table Layout

When a user clicks 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 in *Managing Data Using Oracle Visual Builder Add-in for Excel*). 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.

   > ✎ **Note:**
   >
   > Batch processing is handled differently. See Upload Changes Using Batch Requests.

5. Success and failure is noted in the Status column. Rows with warnings are indicated with a warning icon.

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 and converted into existing rows that can be edited.

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 enabled, empty values are skipped. 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.

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 (for example, data type, required, and so on) are validated (see Data validation in *Managing Data Using the Oracle Visual Builder Add-in for Excel*); read-only fields are skipped. If validation failures exist, the upload is stopped and no subsequent REST requests are made.

   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.

If the form and table both have pending changes, there are a minimum of two requests: one for the form and one (or more) for the child table.

The form and child table changes are never sent in a single request. In particular, sending a single request where the payload contains values from the parent form fields along with an array of values from the child table rows is not supported.

For the form changes, notification of success and failure is noted in the Status Viewer.
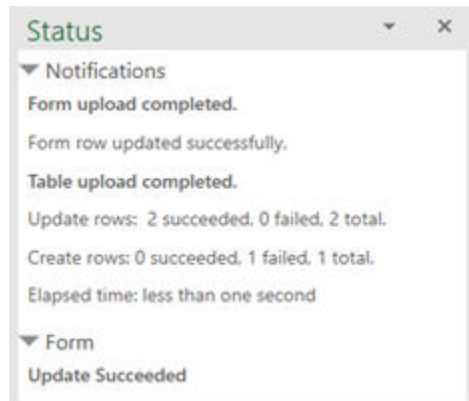
For table changes, success and failure is noted in the table's Status column and well as in the Status Viewer. See Upload Changes from a Table Layout for more information about table status after Upload.

Upload Changes for Custom Actions

For Oracle business object REST API services that expose custom actions on 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.

During the upload operation (which may also include update, create, and delete tasks), the add-in performs the following steps for each marked row:

1. 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).

   • If any values from these columns 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. 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)

3. Makes the request:

- Sends the POST to the custom action path (POST is the only HTTP method supported for invoking custom actions)
- See the note below about batch requests.

4. 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 Oracle business object REST API 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 with a simple value such as a string or number, then the add-in displays that result in the Status Viewer. The response payload for a successful invocation should have a result member in the following format:

```
{
 "result": "<return-value-from-custom-action-method>"
}
```

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 26281."`. This result is displayed in the Status Viewer, as shown here:
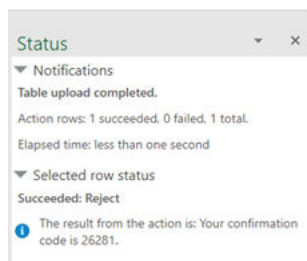
See Executing a Custom Action in the *Developing Fusion Web Applications with Oracle Application Development Framework*.

For information on custom actions, see Custom Actions.

Upload Changes Using Batch Requests

Oracle Visual Builder Add-in for Excel supports the use of batch requests for uploading changes to Oracle business object REST API services.

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 batch requests.

For other REST service types, the add-in sends just one row per request.

The add-in uses an algorithm to determine the optimal size of the payload (number of rows) for a batch request as the upload operation proceeds. This is done to optimize performance and avoid timeouts. It works like this:

1. The add-in sends an initial set of batch requests using 10 rows per batch.

2. The add-in measures the response times for those initial requests and calculates the approximate number of rows that can be processed in 10 seconds.

3. Using the new value, the add-in sends one or more batches to the service.

4. The add-in recalculates the number of rows after each set of batch requests, using the target of 10 seconds per request, until all rows have been processed.

> **Note:**
>
> The target of 10 seconds is not guaranteed, since actual response times vary depending on the REST service business logic, server load, network latency, and so on.

If a batch request contains one or more errors, no changes are made by the service. In that case, a second batch request containing only the rows that succeeded during the first batch request is sent. If the second batch request fails, the add-in falls back to sending one row per request. For more information, see Making Batch Requests.

If a set of changes includes custom actions and those actions have batch enabled, the changes are included in a batch request. However, if one or more rows is marked for a custom action that has batch disabled, the add-in reverts to sending the changes row by row. (See also Batch Mode for Custom Actions.)

If you configure the add-in for parallel requests, it sends up to 4 batch requests in parallel depending on your settings. See Configure Parallel Requests During Upload.

Upload Changes Using Upsert Mode

Oracle business object REST API services support an optional "upsert" mode that you can use when uploading changes to your REST service.
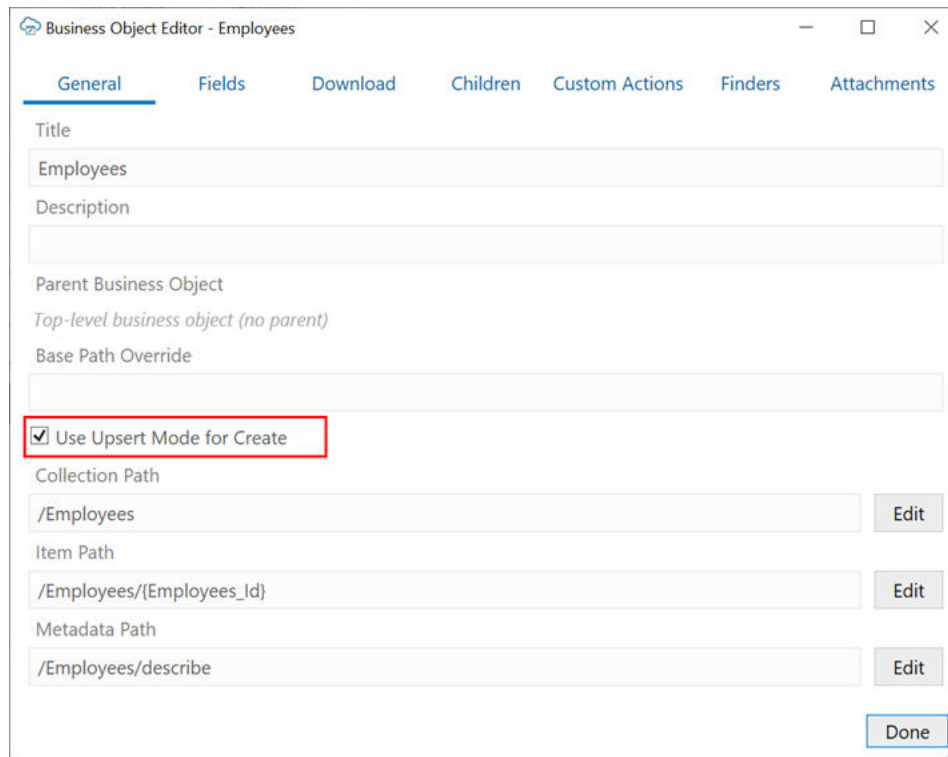
Ensure that your REST services business object supports upsert mode before you enable it.

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.

> **Note:**
>
> 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 General tab of the Business Object Editor. After you enable it, refresh any layouts in the workbook that are based on that business object.

When upsert mode is enabled, the add-in sends the HTTP header, `Upsert-Mode: true`, with POST requests for single-row creates. For batch mode, the add-in sets the operation value for each batch part to "upsert" instead of "create".

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 value is empty or null. Null values can potentially cause validation errors.

To avoid these errors, you can configure the add-in to omit null values 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 v3 (OA3) document for this field. If "nullable" is true, the check box is deselected and null values are uploaded; if "nullable" is false (or the property is missing), then the check box is selected and null values are omitted from the upload.

> **✎ Note:**
>
> If this check box is selected (null values are skipped), you won't be able to change a "non-null" value to a null 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 already sends only changed values from a form by default.

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 batch 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.

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**.
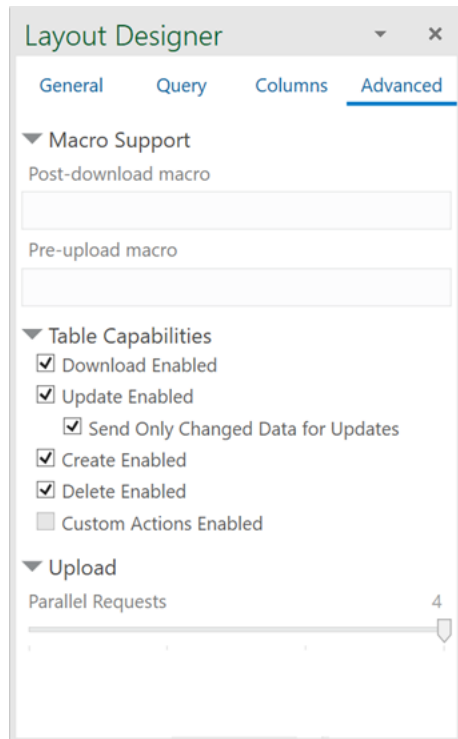
When the check box is enabled, the add-in caches a copy of all of the data in the table during download. Then, when the business user triggers an upload, the add-in compares values in editable cells in rows marked for update to the cached values, and only includes values that are different in the request 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.
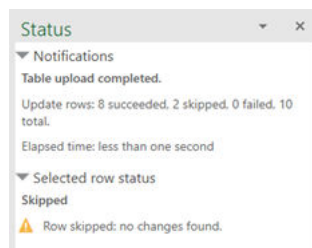
On a successful Create or Upload of a row, the cache is updated with all the current values to be compared during the next upload.

> **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 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 is 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 an Oracle business object 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. Instead, when the server detects such a change, its response allows the add-in to display an error message similar to the following for any such rows in the table:

   ```
   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 creator of your workbook 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.

Configure Parallel Requests During Upload

You can configure Oracle Visual Builder Add-in for Excel to send multiple requests in parallel during upload to improve performance and shorten the overall time to upload a large set of changes.

The add-in allows you to set from 1 to 4 requests in parallel. If you leave the setting on the default (4), the add-in sends four requests at the same time during upload instead of one request after another. If you choose a value of "1", the add-in sends only one request at a time during upload.

This feature is available for both layout types and can be set independently for each one.

Parallel requests behave differently than batch requests. With a batch 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.

> **Note:**
>
> While the add-in collects batches of rows in "table order" (from top to bottom), the threading used for parallel requests may result in batches of rows arriving at the REST service endpoint "out of order". For example, batch 7 may arrive at the endpoint before batch 6.

1. Open the Layout Designer for a worksheet and then click the Advanced tab.

2. Under Upload, drag the **Parallel Requests** slider to your desired setting.

   We recommend setting the value to 4 parallel requests (the default) during upload. However, some services are not compatible with parallel requests due to the their internal business logic. For such services, set this value to "1". Keep in mind that this setting generally results in slower upload performance.

   > **Note:**
   >
   > Take care to test thoroughly in a realistic environment when adjusting this value. Performance may actually degrade if you have too many parallel requests, as may happen if you have multiple users performing uploads at about the same time.

11
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.

> **Note:**
>
> Before you create a set of dependent layouts, ensure your REST service meets the requirements as set out in Requirements for Dependent Layouts.

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.
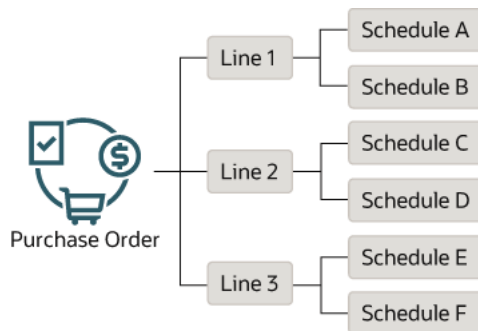
A hierarchy of business objects can go to even more levels ("great-grandchildren" and "great-great-grandchildren") or have more than one business object at each level ("siblings"). For example, the following hierarchy has "sibling" grandchildren (**attachments** and **schedules**) and a "great-grandchild" (**distributions**) under **schedules**.
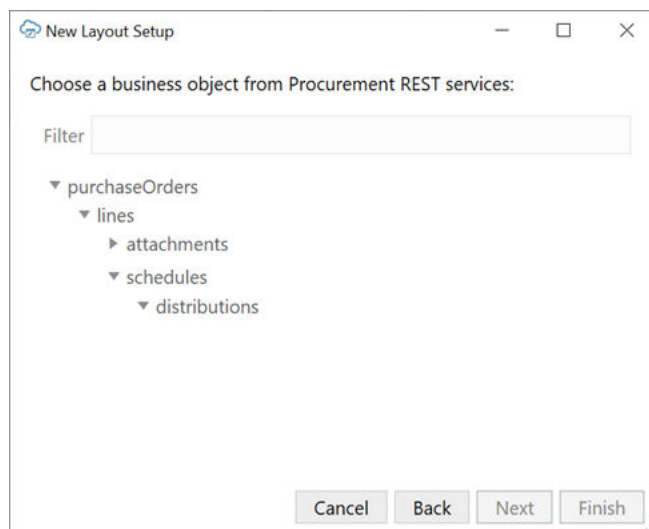
Configuring Dependent Layouts

To create a set of dependent layouts, create either a Form-over-Table or a Table layout as your top-level or "primary" layout on the first worksheet. Then create Table layouts for each subordinate level on separate worksheets. When you are done, link each dependent layout to its immediate parent layout.

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.
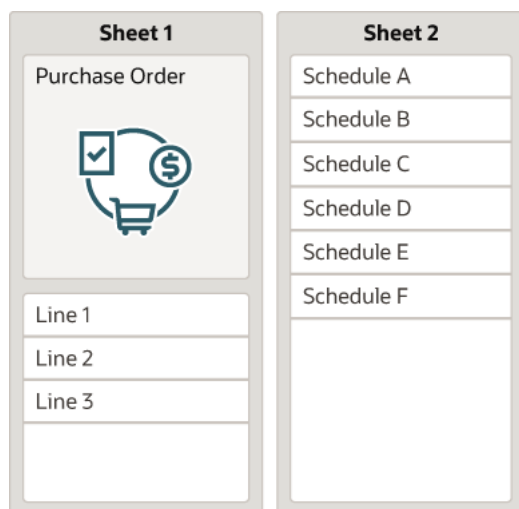
You can also use a Table layout as your primary layout instead of a Form-over-Table layout. In this case, create Table layouts for each level in the hierarchy

(**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.
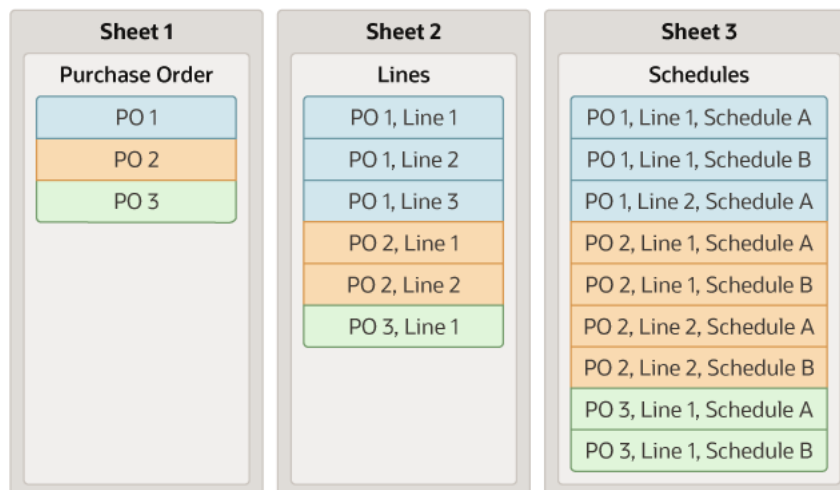
> **Note:**
>
> Keep in mind that a single parent may have multiple children with each of these having multiple grandchildren. Consequently, a large number of results in the primary layout may result in very large volumes of data in the subordinate layouts. To avoid performance issues when downloading and uploading data, Oracle recommends that you configure an appropriate query to limit the number of results in the primary layout. See Use Search Parameters to Limit Downloaded Data.

"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.

When you've created all the necessary layouts, remember to link each dependent layout back to the parent.

When you create a dependent layout, you can choose to include columns from the parent or higher ("ancestor") layout. For example, if you allow your business users to create new rows in a layout (**Create Enabled** is selected in the Table's capabilities), you must add at least one column from the parent layout that uniquely identifies the parent of the new child item. In order to create a new distribution record, for example, the business user must be able to specify the correct schedule for the new distribution.

You might also want to add an ancestor column to help your business users track which child items are associated with which high-level items. See Add Ancestor Columns to a Dependent Layout.

Once the dependencies are established, download, upload, and clear operations act on all the linked layouts, starting from the parent 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.

To create a set of dependent layouts, create either a Form-over-Table or a Table layout as your primary layout on the first worksheet and then create Table layouts for the other business objects in your hierarchy on separate worksheets. When you are done, link each dependent layout to its immediate parent layout. Finally, configure an appropriate query to limit the number of results in the primary layout.

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 items are associated with which high-level items. See Add Ancestor Columns to a Dependent Layout.

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 items 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.

1. Create a Form-Over-Table layout for the first two levels in the business object hierarchy.

   a. In the Oracle Visual Builder tab, click **Designer** to launch the New Layout Setup wizard.

   b. When prompted, provide the service description document.

   c. Choose the parent business object. For example, select **purchaseOrders** to create a layout for purchase orders over lines (the first two levels in our hierarchy) and click **Next**.

   d. Select **Form-over-Table Layout** as the new layout and click **Next**.

   e. Choose the child business object. For example, select **lines** to complete the purchase orders over lines layout and click **Next**.

   f. Review the layout details and then click **Finish**.

   A Form-over-Table layout is created in the worksheet, where **purchaseOrders** is the form business object and **lines** is the table business object. This worksheet is now your primary layout.

Layout Designer

General   Query   Form   Table   Advanced

Form-over-Table ID

FOT197405977

Origin Cell

A1

Form Label

purchaseOrders

Form Business Object

purchaseOrders

Table Business Object

lines

Business Object Catalog

Fusion Procurement REST API

2. Create a Table layout for each of the other business objects in your hierarchy. Ensure that you select the same business object catalog that is used by your primary Form-Over-Table layout.

a. Click the New Sheet icon to add a new worksheet.

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

c. From the first screen of the wizard, choose the business object catalog and click **Next**.

d. Choose the business object that's next in the hierarchy—for example, **schedules**—and click **Next**.

e. Select **Table Layout** as the new layout and click **Next**.

f. Review the layout details and then click **Finish**.

g. Repeat steps **a** to **e** to create Table layouts for all the other business object in your hierarchy. Continuing our example, create a Table layout for the sibling, **attachments**, and one for **distributions**.

A Table layout is created for each business object in your hierarchy. Notice the **Parent Layout** field in the Layout Designer's General tab, shown here for the schedules layout. This field shows only in layouts where the business object is a child of another business object in the same business catalog.
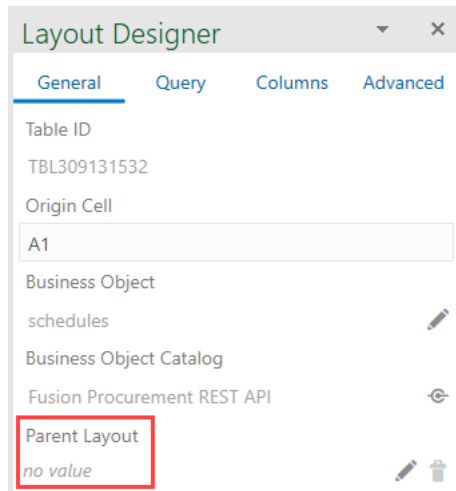
3. Set the parent layout for each Table layout in the hierarchy starting with the lowest level.

   a. On the worksheet for the last item in the hierarchy (**distributions** in our example), click the Choose Parent Layout icon (✎) in the Layout Designer's General tab.

   b. 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:
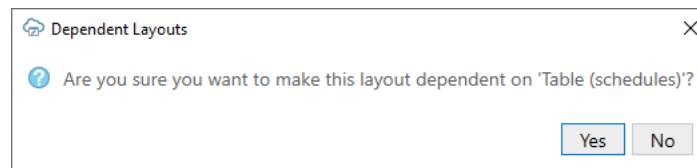
   c. Repeat this step for each Table layout in your business object hierarchy. Continuing our example, the parent layout for schedules is purchaseOrders-lines (which is also the primary layout). You don't need to set a parent for the primary layout.

   > **✎ 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.

Once a layout has its **Parent Layout** defined, you'll notice a **Child Layout** field in its parent's Layout Designer. Here's the Layout Designer for the schedules Table layout, whose parent is purchaseOrders-lines and children are distributions and attachments.
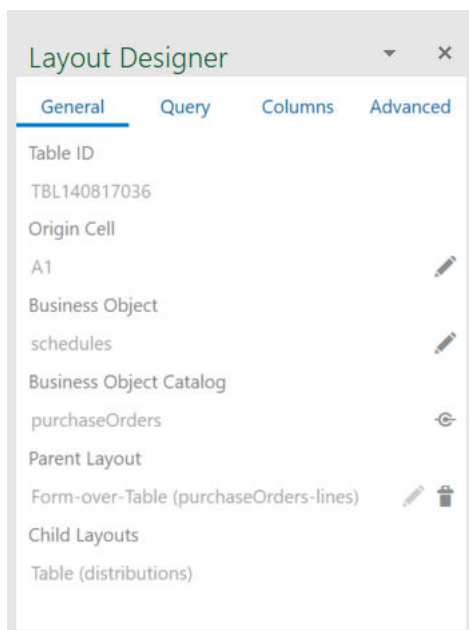
4. Add columns to a layout from a parent or higher layout if required. See Add Ancestor Columns to a Dependent Layout for the steps.

   If you plan to enable the creation of new items on layouts below the primary layout (for example, schedules and distributions), you'll need to have at least one column from the immediate parent layout to display in your layout.

   A column for each parent field added is created in the child layout, for example, the **Schedule (schedules)** column as shown here:
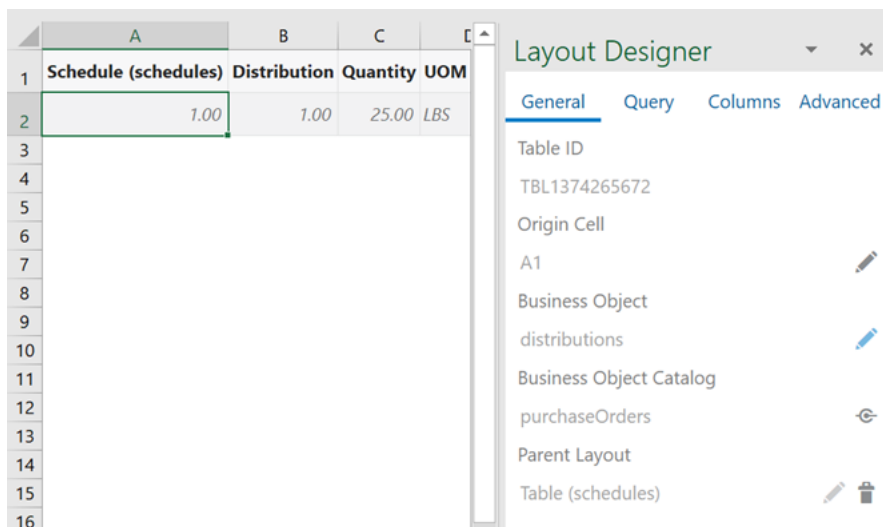
Configuration for your dependent layout is complete. You can choose to configure the workbook further to limit the data that is downloaded to the primary layout. See Configure Search Options for Download.

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. See Manage Data in a Dependent Layout in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

Add Ancestor Columns to a Dependent Layout

When you create a dependent layout, you can choose to include columns from the parent or higher layout. Columns from these higher-level layouts are referred to as "ancestor" columns.

You might add an ancestor column to a dependent layout to help your business users track which child items are associated with which high-level items. For example, if the `LineNumber` field is displayed as a column in the Lines layout, you can add it to the Schedules layout to see which schedules are associated with this line.

You might also choose to add at least one column from the layout's immediate parent if you want to allow your business users to create items in the dependent layout. The parent column you choose must uniquely identify the parent record and must be exposed in the parent layout.

When there is a parent column in the layout, your business users can specify a unique parent for any new items they create. For example, to create a new schedule, a user must be able to specify an existing line with which to associate the new item. Therefore, you would want to expose the `LineNumber` field as a column in the Lines layout and then add this column to the Schedules layout.

If no parent columns are configured, the table cannot support item creation and rows inserted into the table are ignored during upload.

Here are a few things to keep in mind when deciding whether you should add ancestor columns:

- You can add ancestor columns to a second-level sibling layout if desired, but this is not required. When you set the parent for your sibling table layout to the form part of a Form-over-Table, you ensure that all the child items displayed in the sibling layout on download will be associated with the parent item displayed in the form.

- If a desired parent field is already displayed as a column in the child layout, remove that column from the layout and instead add it as an ancestor column as described in this task.

> **✏ Note:**
>
> If a grandparent or higher field is already exposed as a column and is required for create or update, do not remove it.

- If you are adding an ancestor column that is referenced by a list of values in the dependent layout, make sure the ancestor column is positioned before the field with the list of values. See Configure a Filter with a Dynamic Parameter for more information.

To add an ancestor column:

1. Select the dependent layout you want to add an ancestor column to, then click the **Columns** tab in the Layout Designer.

2. From the top left corner of the Columns page, click the Manage Column icon (➕) to open the Table Column Manager.

3. From the Table Column Manager, 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** and those for the schedules business object are shown under **lines > schedules**.
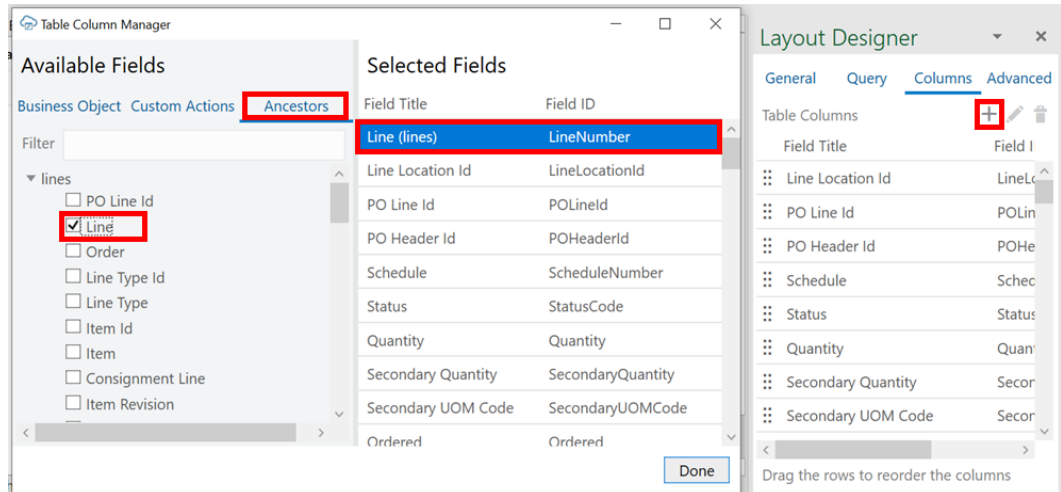
Note:

> The field must be exposed as a column or form field in one of the higher-level layouts. If you don't see the field you want, you'll need to add it to the ancestor layout.

4. Expand the list if necessary, then select the ancestor fields you want to add to your layout.

Note:

> To add an ancestor field before an existing column in the table, select the existing column in **Selected Fields** list, then select the ancestor field check box. For example, to display an ancestor field first in the table layout right after the Status column, select the first column.

5. To change the order of fields in the table, drag and drop fields in the **Selected Fields** list. If the column you added is referenced by a list of values, make sure it is before the field with the list of values.

6. When you've added all the ancestor columns you need and ordered them to your liking, click **Done**.
   The table header for the ancestor column uses the format "*<field title>* (*<business object title>*)" such as "Line Number (lines)".

You can also reorder the columns in a table layout by dragging and dropping columns in the Table Columns list in the Layout Designer.
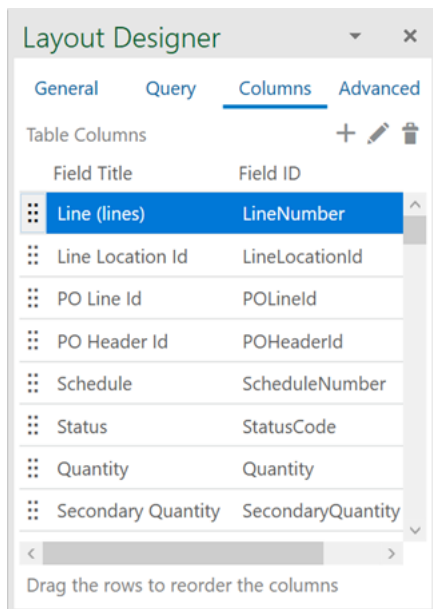
To remove an ancestor column, select it from the list, then click the Delete the Selected Columns icon (🗑). You can use the Ctrl and Shift keys to select multiple columns for deletion.
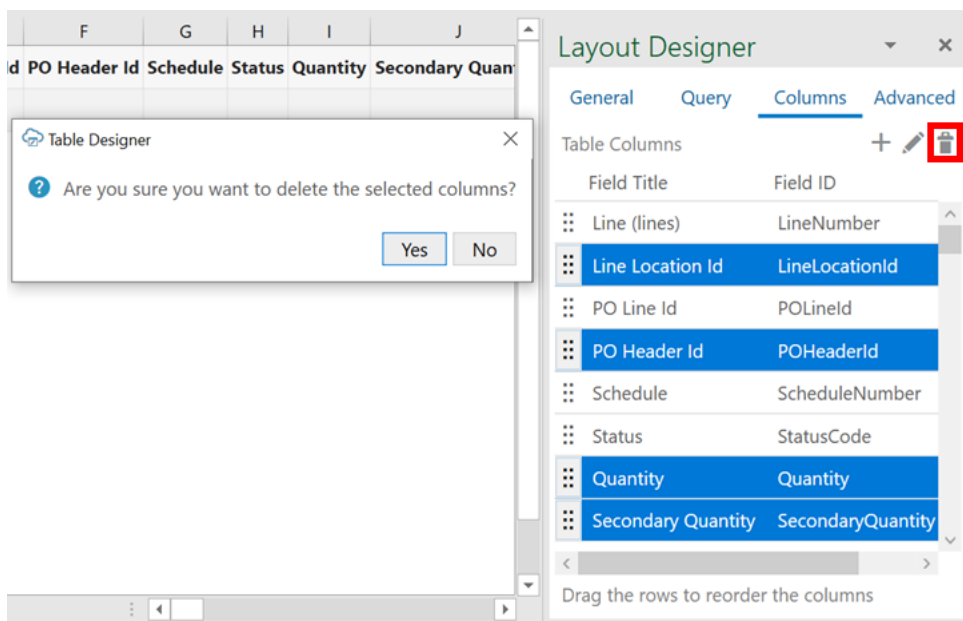
Create Search and Search Parameter Queries for Dependent Layouts

You can create search and search parameter queries on a set of dependent layouts to limit which data get returned from the REST service when a business user invokes a download.

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 search 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 search parameter queries 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").

> **Note:**
>
> Some query 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 Search to Limit Downloaded Data and Use Search 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.

> **Note:**
>
> The "Download All/Stop Now" prompt also only applies to the primary layout. This prompt appears during download when the number of downloaded rows in the primary layout reaches the configured limit. Other pagination settings apply to all layouts in the hierarchy. For more information about the Limit Parameter settings, see Configure Pagination for a Business Object

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. If there is a search query and/or search query parameters defined for the primary layout, the add-in returns only those values matching the criteria. After the primary layout is populated, the add-in makes the next worksheet in the hierarchy active and retrieves all the appropriate items. Again, if there are search parameters defined here, the add-in retrieves all matching values. See Create Search and Search Parameter Queries for Dependent Layouts.

The Download operation populates the "ancestor" column cells with the appropriate values. These cells are read-only.

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.

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:
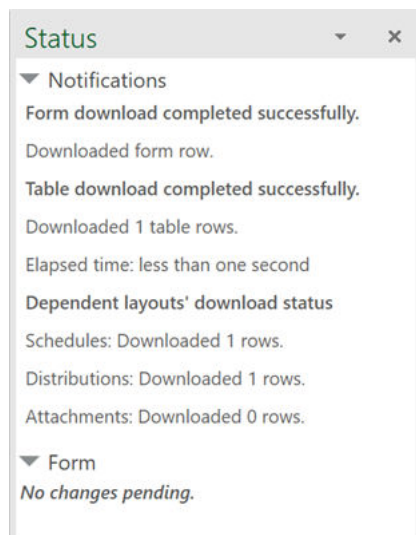
Following a download, you can edit data much as you would in a Table or a Form-over-Table layout.

Uploading Changes

On upload, the add-in makes the primary layout active and sends all updates. If the primary layout uses a Form-over-Table, 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 parent column cells. If one row in the parent table matches, it is used as the parent row. If more than one row matches, the first matching row is used. If 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.

Clear

When the clear operation is invoked, 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.
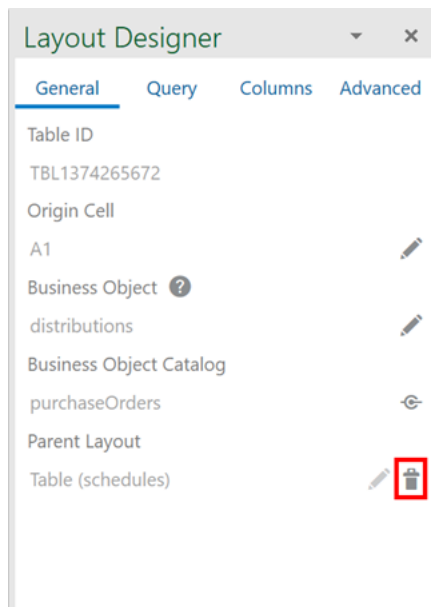
3. When prompted, click **Yes** to remove the dependency.

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:

```
▼ PurchaseOrders
    ▼ lines
        ▶ schedules
```

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 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

• Support for Oracle REST Data Services (ORDS) requires version 22.1.0 or later. 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.

12

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.
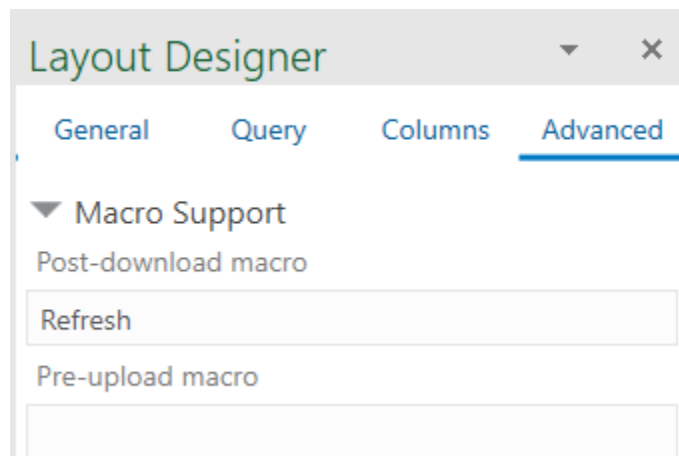
> 💡 **Tip:**
>
> Do not include the parentheses when specifying the name of the macro.

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.

> ✏️ **Note:**
>
> 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.

13
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.

> **Note:**
>
> Publishing is optional. All data editing features of an integrated workbook are available in both published and unpublished copies of the workbook.

The recommended steps to take before you publish an Excel workbook are:

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**.
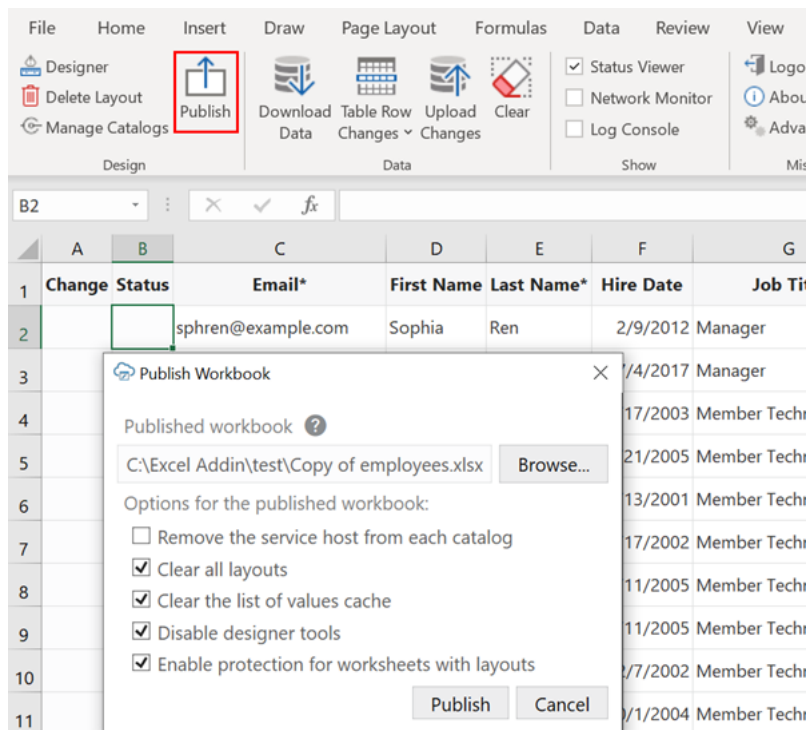
The Publish Workbook window opens. If the name of the original workbook ended with `-source` (for example, `employees-source.xlsx`), the add-in will offer the same name without `-source`.

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).

6. In the Publish Workbook window, choose other options for your published workbook:

   • If you want users to enter the service host when they open the published workbook, select **Remove the service host from each catalog**. This option lets users connect to another service host to access the REST service.

   > **Note:**
   >
   > The OAuth 2.0 properties are *not* altered in this case. See OAuth 2.0 Authorization Code Flow.

   • Select **Clear all layouts** 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.

   • Select **Clear List of Values Cache** 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.

   • If you don't want users of the published workbook to modify its layout, select **Disable Designer tools**. 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.)

- If you want to prevent users from modifying read-only fields and performing Excel actions in the published workbook, select **Enable protection for worksheet with layouts**. This option enables Excel's worksheet protection for each worksheet with a layout.

7. 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.

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:
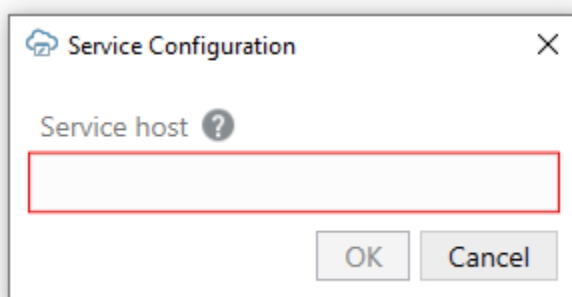
Actions that require access to the service include the Download Data and Upload Changes commands.

> 💡 **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:
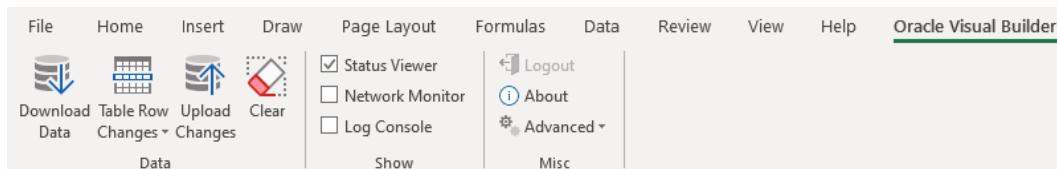
- If worksheet protection was enabled and a user tries to modify a read-only field, a message similar to the following image is shown:
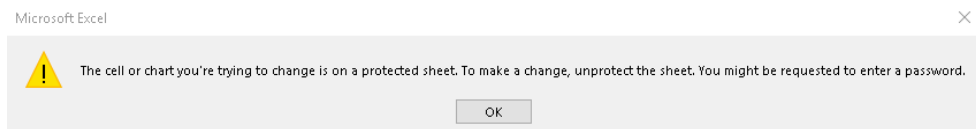
Worksheet protection also prevents the user from performing Excel actions that might disrupt the workbook's integration with the service. See Understanding Read-Only Behavior in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

14

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 service types:

- Oracle business object REST API services
- Oracle REST Data Services (ORDS)
- Other services

Oracle Business Object REST API Services

Oracle business object REST API services provide rich service descriptions that Oracle Visual Builder Add-in for Excel can analyze to provide a business object catalog with many details already filled out.

Oracle business object REST API services include many services offered by Oracle Cloud Applications as well as custom business objects in Visual Builder. When the add-in integrates an Excel workbook with Oracle business object REST API services, it supports special features such as:

- Graphical search editor
- Row finders
- Batch upload
- Polymorphic business objects
- Custom actions
- Upsert mode
- Business object hierarchies

See Accessing Business Objects Using REST APIs and Creating ADF RESTful Web Services with Application Modules.

Oracle REST Data Services

Oracle Visual Builder Add-in for Excel supports Oracle REST Data Services (ORDS) when you provide an OpenAPI description of an ORDS service. ORDS with AutoREST can provide an OpenAPI service description.

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 *Oracle REST Data Services Installation, Configuration, and Development Guide*.

For manually-created REST services using ORDS, you'll need to define modules, templates, and handlers in order to get an OpenAPI service description. See Manually Creating RESTful Services Using SQL and PL/SQL in the *Oracle REST Data Services Developer's Guide*.

After importing an ORDS service description, 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.

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 description 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 graphical search editor is not available for these services but search parameters can be used. See Use Search Parameters to Limit Downloaded Data.

Supported Data Types

The add-in supports a variety of data types exposed by business objects in web applications developed using Visual Builder and data types exposed by REST services.

The add-in supports the following OpenAPI data types (derived from the JSON Schema Specification):

- Boolean

- integer

- object

- number

- string

In addition, the add-in recognizes the optional modifier property "format", when it is applied to values of type string. The two formats recognized are "date-time" and "date". There is no support for time, binary, or byte formats or for array-valued fields

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, see the specifications for OpenAPI and JSON Schema:

- OpenApi: https://swagger.io/specification/#dataTypes

- JSON Schema: https://tools.ietf.org/html/draft-wright-json-schema-00#section-4.2

Required Fields

When a business object field is created from a service description, 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 PUT, PATCH, POST, and DELETE operations as follows:

- Existing row updates are enabled if the item path has either a PUT or PATCH operation. For these operations, the add-in includes all data values from editable cells for the changed row(s) on upload, regardless of whether the data value was edited since download.
- 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

> **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.

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 Sub-fields

Oracle Visual Builder Add-in for Excel supports fields of type `Object`. These fields may expose sub-fields, also known as nested fields.

If, for example, you have 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 sub-fields. 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 sub-fields 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 sub-fields of Address, edit Address and find the Sub-fields list on the Field Editor window. The direct sub-fields for Address are Street, City, State, Zip, and GPS Coordinates. Since GPS Coordinates is of type `Object`, its field editor will show its sub-fields (Latitude, Longitude).

2. Next, when creating a Table layout from a business object's fields, the add-in promotes the sub-fields and creates columns for each (leaf) sub-field. 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.

Many different request and response schema types are possible and we 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 API 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.

- Remote references are not supported in OpenApi service descriptions.

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 description 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.

15
Internationalization

Oracle Visual Builder Add-in for Excel is available in various languages. It automatically detects the user's preferred language from Microsoft Excel and uses that language where possible.

The date, date-time, and number formats used by the add-in are culture-sensitive. See also Appearance of an Integrated Excel Workbook.

When using the add-in, you work with:

- Labels visible on the Oracle Visual Builder ribbon and in various windows displayed by the add-in. These labels, known as the **add-in labels**, are owned by the add-in and are localized.

- Labels visible as column headers and field labels are known as **business object field titles**. These titles are owned by the REST service.
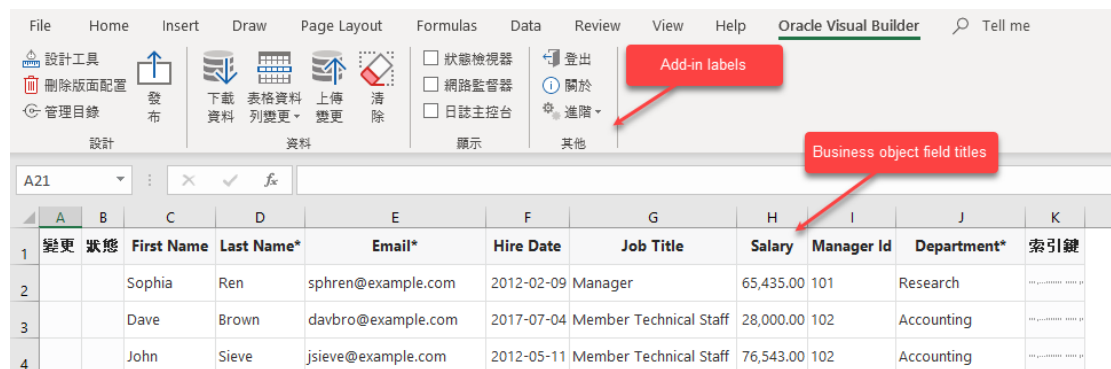
The add-in sends the `accept-language` header to the service on every request. The language setting specified for Excel is used for the add-in labels and for requests to the service, including the describe requests that fetches the initial business object field titles.

> **Note:**
>
> Because business object field titles are owned by the service, contact the service owner for any missing translations or languages.

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.

1. In Excel, click the **Oracle Visual Builder** tab.

2. Choose **Select Language** from the Advanced drop-down.

3. In the Add-in Language drop-down list that appears, select the language you want to use. The drop-down 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 selected language uses a right-to-left writing system, the add-in's user interface elements 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.

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.
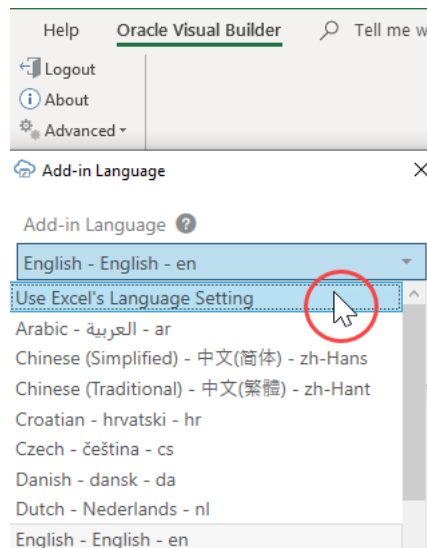
Refresh All Field Titles

You can refresh all field titles in your integrated workbook if, for example, you want to display field titles in another language.

Imagine, an invoices workbook was configured and tested with English as the current language, so all field titles are in English and this information is saved with the workbook. Now imagine this workbook was sent to someone in France for data entry. That person has Windows and Excel configured for French, and she would like to see field titles in French as well. Resetting the field titles enables this data entry operator to see field titles in French.

When you refresh field titles, Oracle Visual Builder Add-in for Excel fetches the service description for each business object, if possible, and refreshes all of the business object's field titles. If a catalog has any business objects that are not used in a layout, the unused business objects are not refreshed. When it finishes, any affected layouts are redrawn automatically.

The add-in uses saved metadata path information to access the service description document. For older workbooks or catalogs with a service type of "other", the metadata path settings are empty. In this case, the add-in sets the metadata path to

the default values for the service framework type before starting the refresh. These settings can be viewed from the **General** page of the **Business Object Editor**.

Before you begin, upload any pending changes to your workbook. If you don't, your pending changes will be discarded during the refresh.

> ⚠ **Caution:**
>
> A reset wipes out any local changes made to field titles using the Business Object Field Editor.

1. In Excel, click the **Oracle Visual Builder** tab.

2. Choose **Refresh Field Titles** from the Advanced drop-down.

3. In the Field Titles window, click **Yes**.

   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 refresh ends without completing.

When the refresh is complete, the add-in displays the results of the refresh in the Status window.

If the add-in is unable to fetch the service description for a given business object, the business object is skipped and the details are logged. To troubleshoot the error, open the log console and repeat the process to discover the cause. See Logging Console.

If the error is caused by incorrect or missing path details, consult with the REST API owner and update the path information, including the base path and metadata path settings, as required. See Manage Metadata Path Information.

> ✎ **Note:**
>
> When the add-in fetches the service description from the metadata path, it includes the `accept-language` header as described in Natural Language Support. A given service may or may not have translations for the requested language. For any missing translations, contact the owner of the service.
>
> Oracle REST Data Services (ORDS) do not provide localized service descriptions.

16
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.
- Turn off older obsolete security protocols, such as SSL.
- 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.

- **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 exposed by Oracle Cloud applications that use the Oracle Fusion Applications Token Relay servlet. Refer to this topic for technical details on the Token Relay authentication mechanism.

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.

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 the catalog's host and basepath properties to compose three other key URLs for the login sequence. For example, if the host is `https://<host>` and the basepath is `/fscmRestApi/resources/v14`, then the URLs needed for token relay authentication would be:

- The UI home page: `https://<host>/fscmUI/faces/FuseWelcome`

- The Anti-CSRF endpoint: `https://<host>/fscmRestApi/anticsrf`

- The token relay servlet endpoint: `https://<host>/fscmRestApi/tokenrelay`

The Login Sequence Starts in the Embedded Browser

1. 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 Embedded Browsers.

2. 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.

3. 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 user's gesture or login page logic causes a page navigation event in the browser, the add-in's event handler is notified and the add-in performs an authentication test (see Authentication Test) to see whether the user has logged in.

   - 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.

Authentication Test

The add-in watches for page transitions in the embedded browser and performs a test to see if the user has successfully logged in. If, using the cookies from the embedded browser, the add-in can obtain a token from the Token Relay servlet, then the user has successfully authenticated.

Details about the test:

1. When each page navigation event is raised from the browser, the add-in attempts to get an anti-CSRF token by making a GET request to the anti-CSRF endpoint. This step is skipped once an anti-CSRF token is obtained.

2. The add-in then makes a GET request to the Token Relay servlet endpoint. 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 `token_type` member must have the value "JWT" and the `access_token` member must be present and non-empty.

   b. The value for `token_type` 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.

Variations

1. If the authentication method in the service catalog of the integrated workbook is configured to be "Default", an initial ping request to the anti-CSRF endpoint is made before the login sequence begins.

   a. If the request returns a 404 or other error, then the add-in abandons the Token Relay authentication mechanism. This means it falls back to using Basic authentication.

    b. 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.

2. If the authentication method property in the service catalog of the integrated workbook is configured to be "Oracle Fusion Application Token Relay", the initial ping to the anti-CSRF endpoint is skipped. The ping occurs later during page navigation.

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. The response payload must contain a `token_type` member which must have the value "JWT" and also a `access_token` member, which must be non-empty. See RFC 7519 for more information.

- Oracle Fusion Applications Token Relay is only supported for Oracle Cloud Application deployments that include standardized `/anticsrf` and `/tokenrelay` endpoints with standardized payloads. This behavior is not configurable at this time.

- Token Relay is not supported for other service types such as ORDS or Visual Builder Business Object (VBBO).

OAuth 2.0 Authorization Code Flow

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.

The OAuth 2.0 Authorization Code flow is supported by Visual Builder Business Object services, for example.
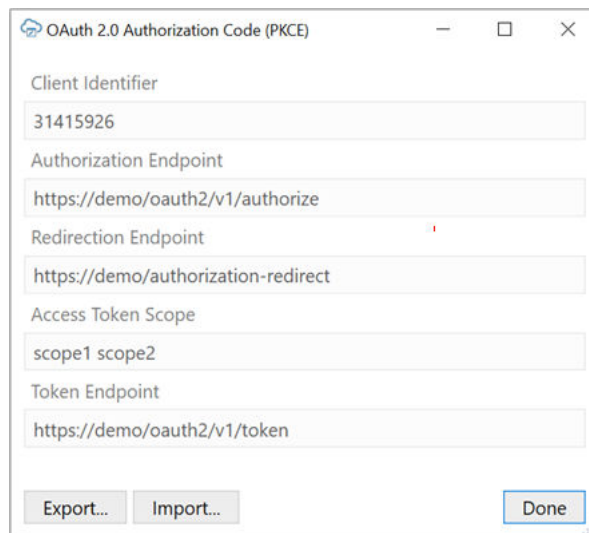
Refer to Authorization Code Grant for information on authorization code flow. See also Proof Key for Code Exchange by OAuth Public Clients for information on PKCE.

OAuth 2.0 Authorization Code Flow Configuration

In order to configure your workbook to use OAuth 2.0 Authorization Code flow, you'll 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.

You can configure this authentication method for a new workbook when you create a catalog. During the new layout setup process, you'll be prompted to provide authentication details when you select "OAuth 2.0 Authorization Code (PKCE)" from the **Authentication** list. See Create a Form-over-Table Layout in an Excel Workbook or Create a Table Layout in an Excel Workbook.

You can also configure authentication details for an existing catalog from the **Advanced** page of the **Business Object Catalog Editor**. When you select this authentication method from the **Authentication** list, click **Edit Authentication Flow Properties** to access the OAuth 2.0 Authorization Code (PKCE) screen. See Set an Authentication Method for a REST Service.

- **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:**
>
> For more information on these properties, see OAuth 2.0 Authorization Framework.

This screen also includes **Import** and **Export** buttons that you can use to upload and download a JSON file with the authentication properties. Here is an example of the JSON:

```
{
  "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"
  }
}
```

> **Note:**
>
> These values are just examples. The correct values will be different for your
> service. Contact the service admin to obtain the correct values.

Details of the Authorization Flow

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.

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.

Limitations and Known Issues

- 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.

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.

17
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.

- 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.

> **✎ 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.

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.
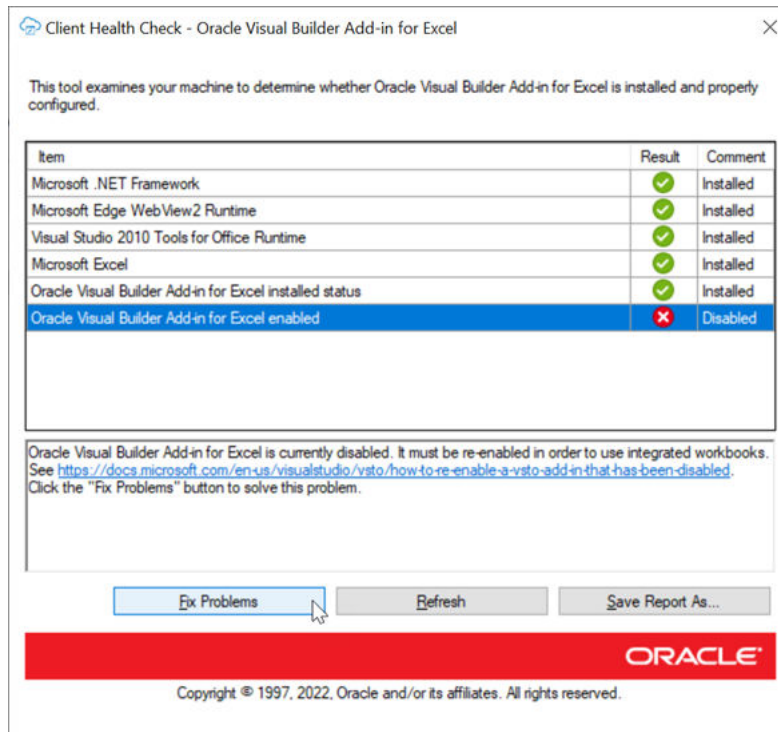
Select each failed item (⊗) for more details, including the steps necessary to resolve the issue. If the add-in can resolve the issue, the **Fix Problems** button is enabled.

> **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:

    a. click **Save Report As...** and choose a name and location for the report.

    b. Review the report and remove any sensitive information.

    c. Send the report to Oracle Support.

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. 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. 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 drop-down list 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.

Logging Console

The Logging console displays log messages based on the actions performed. If you encounter any issues, view the logging messages to troubleshoot and diagnose issues.

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 drop-down.

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.
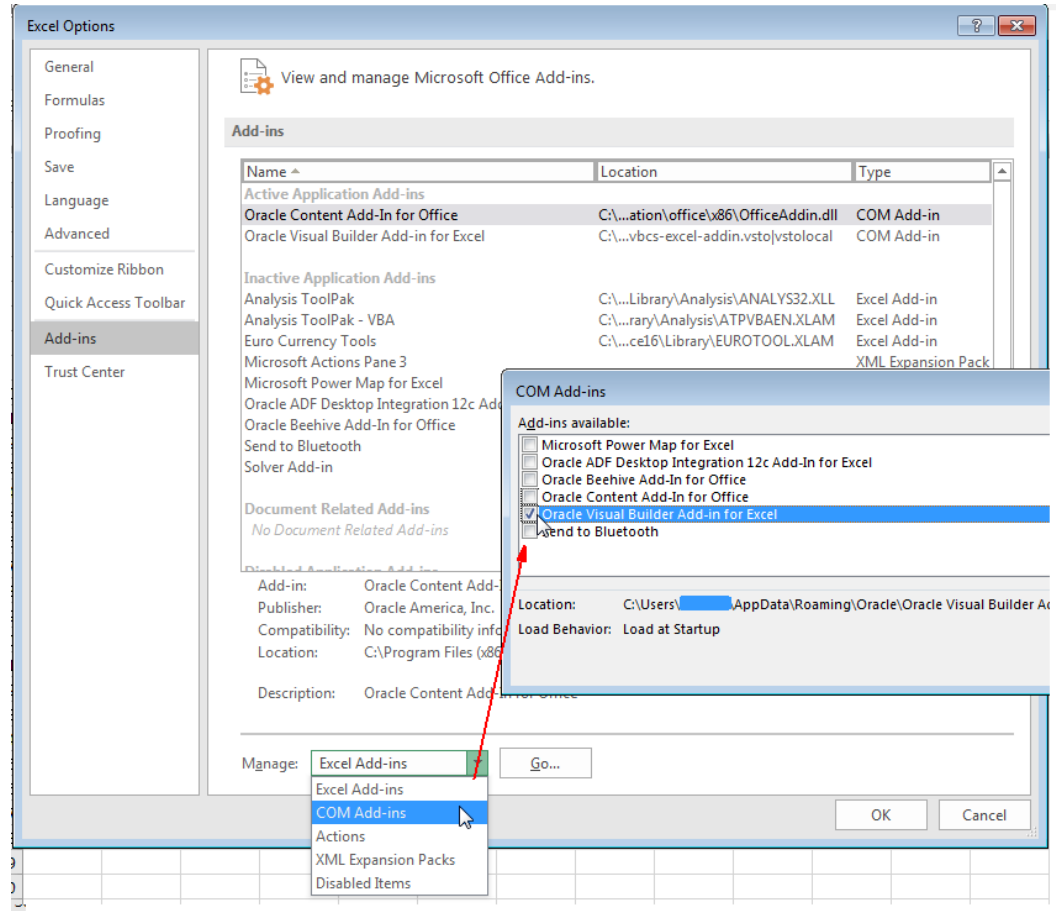
Resolve Workbook Issues

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.
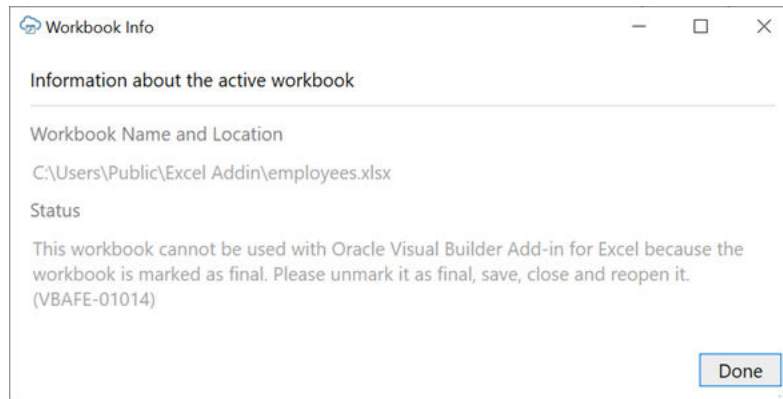
Check the status for the description of the issue and for any troubleshooting steps. Issue-free workbooks have a status of "Integrated".

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.

18
Migration

You can migrate an Excel workbook created or modified with version 2.*x* or 3.*x* of Oracle Visual Builder Add-in for Excel to use the current version of the add-in.

Workbooks created or modified with version 2.*x* or 3.*x* of the add-in migrate seamlessly to the latest version of the add-in. 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. However, once you configure your workbook to use the latest features, it may no longer be compatible with the older version of the add-in. So before distributing your updated workbook, make sure your target audience has access to the corresponding add-in version.

19
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.

For example, you might have 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. For example,
`ProjectNumber={ this.BusinessObject.Parent.Parent.Parent.Fields['ProjectNumber'].Value }` returns the value of the `ProjectNumber` field that belongs to the current business object's great grandparent business object.

The value of an operand or an intermediate result in an expression can be a string or integer. However, the result of a single expression is converted to a string when resolving the entire property value.

For any given configuration property that supports expressions, you must escape any curly braces (`\{` or `\}`) that you wish to use literally.

String literals inside expressions must be enclosed in single quotes ('). Single quotes inside string literals must be escaped (`\'`).

> **Note:**
>
> A given configuration property only supports a few specific types of expressions. Some configuration properties do not support expression language at all.
>
> See Configure a Filter for a List of Values and Configure the Bind Parameters for a Descriptive Flexfield's List of Values for information about support for expressions for these specific configuration properties.

Operators

Operator precedence is high to low.

Operator	Note
[ ] .	Collection access, object member access
( )	Grouping to change precedence
– !	Unary minus, negation
* /	Math (multiplicative)
+ -	Math (additive), also + for string concatenation
< > <= >=	Relational
== !=	Equality
&&	Logical AND
||	Logical OR
? :	Ternary conditional

Reserved Words

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.
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.
Value	Value of a parameter or a field
SelectWindow	Search-and-select window in a list of values

Examples

Here are some sample 'q'-type filter query parameters for use in list of values configurations:

Parameter Value	Use	Sample Value	Final Parameter Value
`DepartmentId={ this .BusinessObject.Fie lds['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.Par ent.Parent.Fields[' DepartmentId'].Valu e }`	This string sets the value of `DepartmentId` in the query to the department Id value in the current row item's "grandparent" layout.	Department id is `101`	`DepartmentId=101`
`FirstName LIKE '{ SelectWindow.Sea rchTerm }*'`	This string matches employees whose first name begins with the user-provided search term entered in the Search-and-Select window.	Search term is `Steve`	`FirstName LIKE 'Steve*'`
`DepartmentId={ this .BusinessObject.Fie lds['DepartmentId'] .Value } { SelectWindow.Sear chTerm == '' ? '' : 'AND FirstName LIKE \'' + SelectWindow.Search Term + '*\'' }`	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*'`

20

Embedded Browsers

Oracle Visual Builder Add-in for Excel uses either the WebView2 or the .NET WebBrowser control as an embedded web browser to display web pages from inside Microsoft Excel.

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.

WebView2 is based on Edge/Chromium. The .NET WebBrowser control is based on Microsoft Internet Explorer technology.

The add-in uses WebView2 if the appropriate runtime is detected on the local computer. Otherwise, it uses the .NET WebBrowser control. There is no option to choose the embedded web browser. Your default web browser setting in Windows Settings has no effect on the add-in.

> **Note:**
>
> If you wish to avoid using Internet Explorer with the add-in, make sure that the WebView2 runtime is installed as described in The WebView2 Control.

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

- Choose one of the "evergreen" installers. Do not choose the "Fixed Version" option.
- Using the WebView Refresh page function can interrupt the login sequence, 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*.

The .NET WebBrowser Control

If Oracle Visual Builder Add-in for Excel cannot find a compatible WebView2 runtime, it uses the .NET WebBrowser control instead. The .NET WebBrowser control is based on Microsoft Internet Explorer technology.

Support Notes

- Microsoft ended support for Internet Explorer (IE) 11 desktop application for certain operating systems on June 15, 2022. Refer to *Lifecycle FAQ - Internet Explorer and Microsoft Edge* at https://docs.microsoft.com/en-us/lifecycle/faq/internet-explorer-microsoft-edge.

- Oracle support for the .NET WebBrowser Control is deprecated and will be dropped in a future release. You should install the WebView2 runtime as soon as possible.

Installation

There is no special installation required. This control is part of the Microsoft .NET Framework.

Since the .NET WebBrowser control is based on Microsoft Internet Explorer, make sure that Internet Explorer has been upgraded to version 11 and that all Microsoft updates and security patches have been applied.

Technical Notes

- 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 Microsoft Internet Explorer (IE).

- When troubleshooting a problem, be sure to pay special attention to your IE configuration. There are key settings under Internet Options that can influence the behavior of the add-in. These settings include the LAN settings, Security settings, Languages settings, and Certificates settings.

- The add-in installer sets the `FEATURE_BROWSER_EMULATION` feature control key for Excel in the registry to IE 11 mode. See Browser Emulation in the Microsoft Internet Explorer documentation.

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.

Oracle Visual Builder Add-in for Excel uses the WebView2 control as an embedded browser if it detects a compatible version of the WebView2 runtime. Otherwise, it uses the .NET WebBrowser control, which is based on Microsoft Internet Explorer (IE).

Clear the WebView2 browser cache by following the instructions in this topic. Clear your IE browser cache through Internet Explorer instead. Check your Internet Explorer help for details.

To clear the cache for the WebView2 embedded browser, choose **Clear Embedded Browser Cache** from the **Advanced** menu.
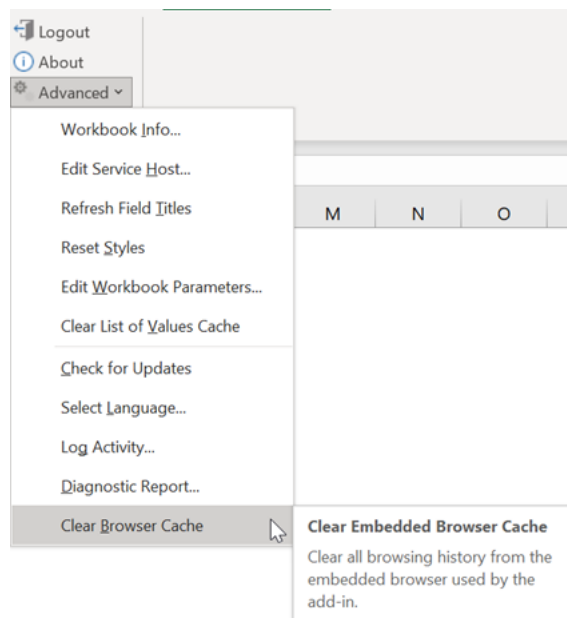
> **Note:**
>
> You can also clear this cache by simply deleting the `EBWebView` folder at `%localappdata%\Oracle\Visual Builder\EBWebView`. The browser recreates the folder when the add-in next accesses the browser.

The **Clear Embedded Browser Cache** command only clears the WebView2 cache. If the add-in is using the .NET WebBrowser control, you'll see this message:
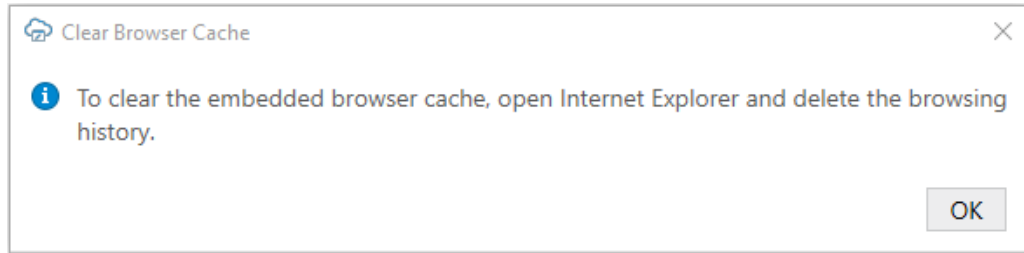
Close this message and clear the browser cache through Internet Explorer instead.

> **✎ Note:**
>
> You can also clear the Internet Explorer browser cache from the Internet Properties dialog. Open this dialog by clicking **Internet Options** from the Windows **Control Panel**.

21

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 1.3.1

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: 1.0.1343.22

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 13.0.1

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 1.9.0

Copyright (c) 2013-2021 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.