Oracle® Cloud

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

F34993-01
October 2020

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

F34993-01

Primary Author: Oracle Corporation

Contents

Preface

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

Topics:

Audience

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

Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud
  `http://cloud.oracle.com`
- 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 Oracle Visual Builder Add-in for Excel*

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.

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 2.4.0

Here’s an overview of new features and enhancements added to Oracle Visual Builder Add-in for Excel in Release 2.4.0. Take note also of some limitations when using the add-in.

New and Changed Features

- Display two or more columns from the immediate parent layout in a child Table layout for multi-level business object hierarchies. Including parent columns allows users to specify the appropriate parent item when creating a child item. See Use Multiple Layouts for Multi-level Business Objects.
- Add descriptive flexfields (DFF) to your layout's forms and tables. DFFs expand to display additional fields, including context-sensitive fields, for tracking supplemental information. These fields are supported in REST services using Oracle ADF Business Components. See Add Descriptive Flexfields to a Layout.
- An enhancement to the way the add-in handles TLS security protocols during the HTPS handshake. In 2.4, the add-in uses the system default for TLS in accordance with Microsoft guidelines. See TLS Security.
- Configurable options to publish a workbook, enabling you to now choose whether you want to disable the designer tools, clear all layouts, enable worksheet protection, and so on. See Publish an Integrated Excel Workbook.
- Process custom actions, along with other types of changes, in batches. See Batch Mode for Custom Actions.
- Refresh a business object catalog after a workbook is integrated with a service. See Refresh a Business Object Catalog.
- Control the REST-Framework-Version header value for Oracle business object REST API services. See Configure the REST-Framework-Version.
- Define a list of values for a custom action's payload field. See Configure a List of Values for a Custom Action Field.
- Add, remove, and update a business object's available row finders for Oracle business object REST API services that support row finders. See Configure Row Finders for a Business Object.
- Change how a business object field's 'required' property is determined and introduce two new properties (**Required for update** and **Required for create**) to distinctly manage the values for an update and a create. See 'Required' Fields.
- Delete the parent form row in a Form-over-Table layout. See Delete a Form Row in a Form-over-Table Layout in *Managing Data Using the Oracle Visual Builder Add-in for Excel*.
- Display event details and verbose output in the Log Console. See Logging Console.

Known Limitations

- Review REST Service Support to understand the limitations around data type support.
- Review Use Lists of Values in an Excel Workbook to understand limitations around using lists of values.
- Review Custom Actions to understand custom action support.
- Workbooks created or modified with version 2.3 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, and 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.
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 data entry users in published workbooks.

Installation

Oracle Visual Builder Add-in for Excel is bundled with Visual Builder and is available to you from your visual application's Data Manager, a tool that lets you manage application data. You can also get the latest version of the add-in directly from the OTN Downloads page.

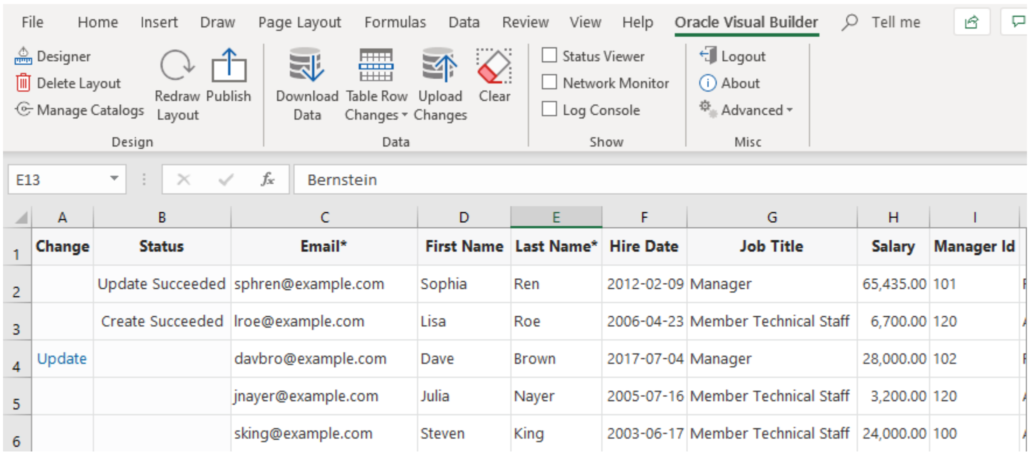
Before you begin, ensure that your environment meets the requirements for installation. The add-in is supported on Windows 10 operating systems that run the 32-bit version of Microsoft Excel 2016 or 365 (see Supported Platforms for the Visual Builder Add-in for Excel (Doc ID 2474783.1)).

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:



	A	B	C	D	E	F	G	H	I
1	Change	Status	Email*	First Name	Last Name*	Hire Date	Job Title	Salary	Manager Id
2		Update Succeeded	sphren@example.com	Sophia	Ren	2012-02-09	Manager	65,435.00	101
3		Create Succeeded	lroe@example.com	Lisa	Roe	2006-04-23	Member Technical Staff	6,700.00	120
4	Update		davbro@example.com	Dave	Brown	2017-07-04	Manager	28,000.00	102
5			jnayer@example.com	Julia	Nayer	2005-07-16	Member Technical Staff	3,200.00	120
6			sking@example.com	Steven	King	2003-06-17	Member Technical Staff	24,000.00	100

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

1. In the **Oracle Visual Builder** tab, click **Designer**.
2. Provide the URL of a service 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.

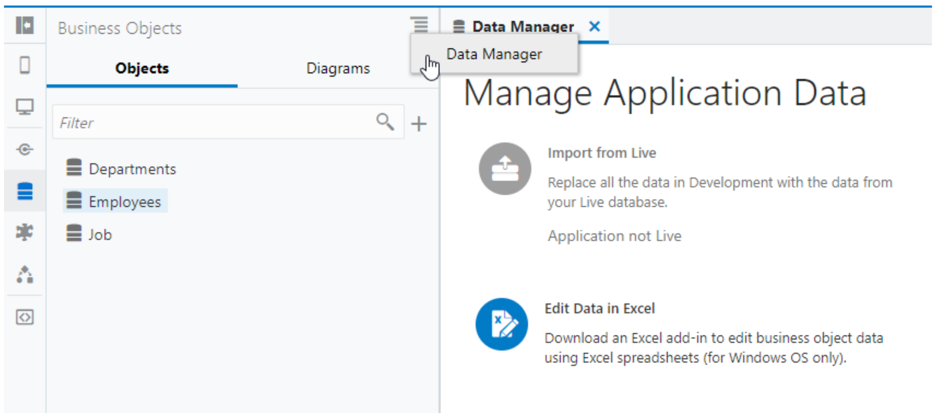
3

Install Oracle Visual Builder Add-in for Excel

To install Oracle Visual Builder Add-in for Excel, you must download the `vbafe-installer.exe` installation file, from either your visual application's Data Manager or the Downloads page.

You can install the add-in on Windows 10 operating systems that run the 32-bit version of Microsoft Excel 2016 or 365. Follow these steps to install the add-in afresh or upgrade an existing installation. If you're upgrading, make sure you follow the recommended practices (see Best Practices for an Upgrade).

1. To make sure you're installing the latest version of the add-in, we recommend that you download the installation file from the Downloads page. However, if you prefer to get the installer from your visual application:
   a. Click the **Business Objects** tab for your visual application.
   b. Click the menu option, then select **Data Manager**.
   c. Click **Edit Data in Excel**.



Replace all the data in Development with the data from your Live database.

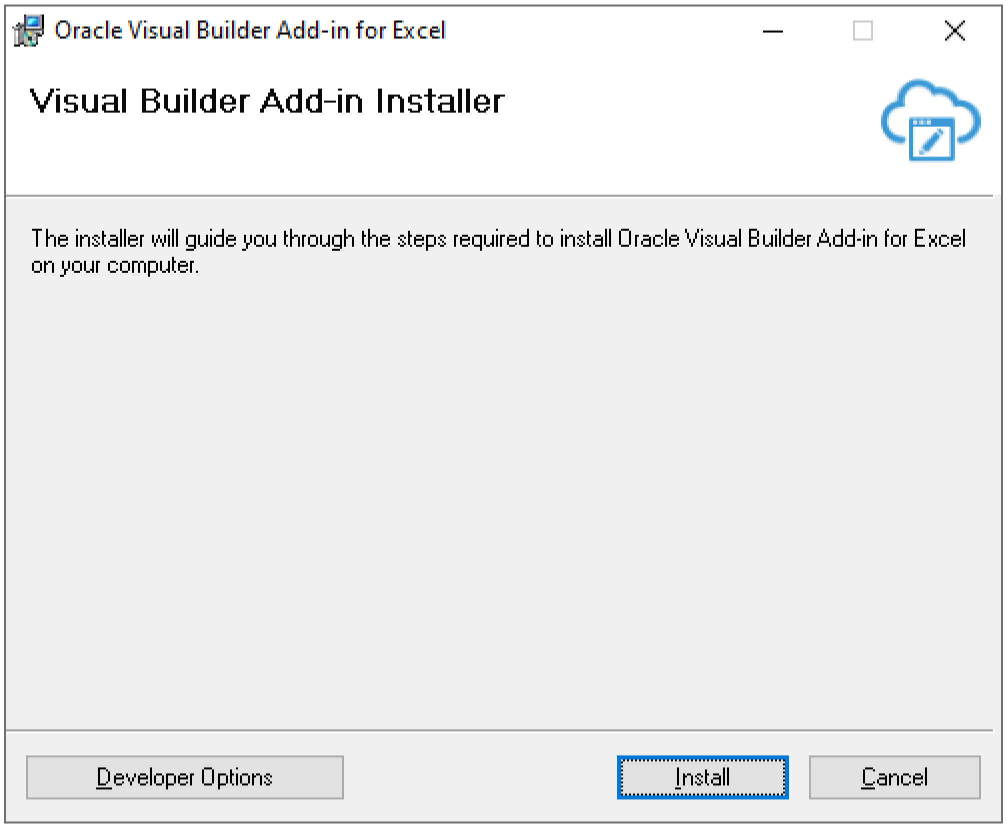
Download an Excel add-in to edit business object data using Excel spreadsheets (for Windows OS only).

2. Quit Excel before you begin installation.
3. Double-click the `vbafe-installer.exe` file that you downloaded previously to launch the installation wizard.
4. When you install Oracle Visual Builder Add-in for Excel, you get a full set commands to design layouts, publish workbooks, and more. If you don't need access to these designer tools, you can choose to install only the tools that help manage data:
   a. Click **Developer Options**.
   b. Select **Disabled**.

Tip:

If you want to change this setting after initial installation, re-run the installer, choose the option to repair your installation, and change your selection.

5. Click **Install**.


Oracle Visual Builder Add-in for Excel
Visual Builder Add-in Installer
Developer Options
Install
Cancel

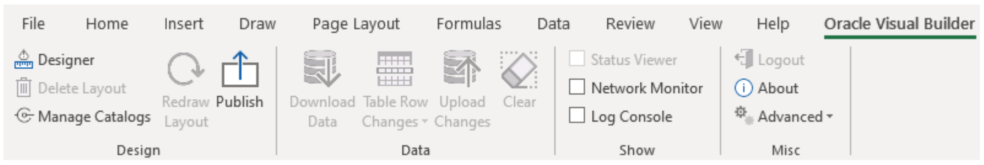
The installer will guide you through the steps required to install Oracle Visual Builder Add-in for Excel on your computer.

The add-in requires the Microsoft .NET Framework and Visual Studio Tools for Office Runtime to be installed on your computer. If this software is not present, both are installed.

Note:

While administrator privileges are not required to install the add-in, you must have administrator privileges to successfully install Microsoft .NET Framework and Visual Studio Tools for Office Runtime. The add-in is installed for the current Windows user only.

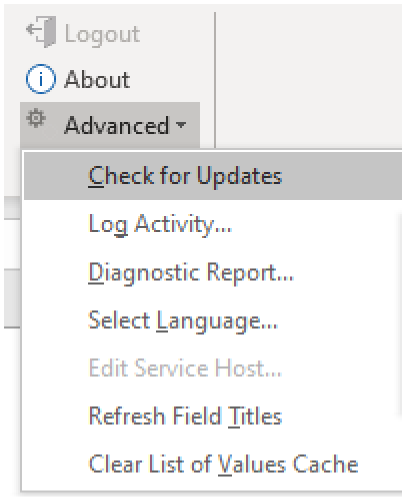
6. Once installation is complete, click **Close**.
   If you run into issues, see Troubleshooting Your Installation.
7. Start Excel and open a new workbook.
   A new **Oracle Visual Builder** ribbon tab appears, with commands to integrate the Excel spreadsheet with a REST service. If you disabled the design tools during installation, you won't see the **Design** options in the first pane of the ribbon tab.



Check for Updates

It's a good idea to periodically check whether a newer version of the add-in is available for you to install. The latest version of the add-in is posted to the Downloads page. Make sure you have an Oracle.com account to sign in and download the new version.

1. From the Advanced drop-down in your existing installation, select **Check for Updates**.



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

Best Practices for an Upgrade

To ensure a clean upgrade, follow these instructions when upgrading your installation of Oracle Visual Builder Add-in for Excel.

Tip:

If you are upgrading from version 1.x, review Migration first.

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 `vbafe-installer.exe` installer file for the new version. The installer will automatically replace 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.

Install from the Command Line

If you want to install Oracle Visual Builder Add-in for Excel from the command prompt, you can use command-line switches with the `vbafe-installer.exe` file to control or customize your installation.

Table 3-1 Oracle Visual Builder Add-in for Excel Installer Command-Line Switches

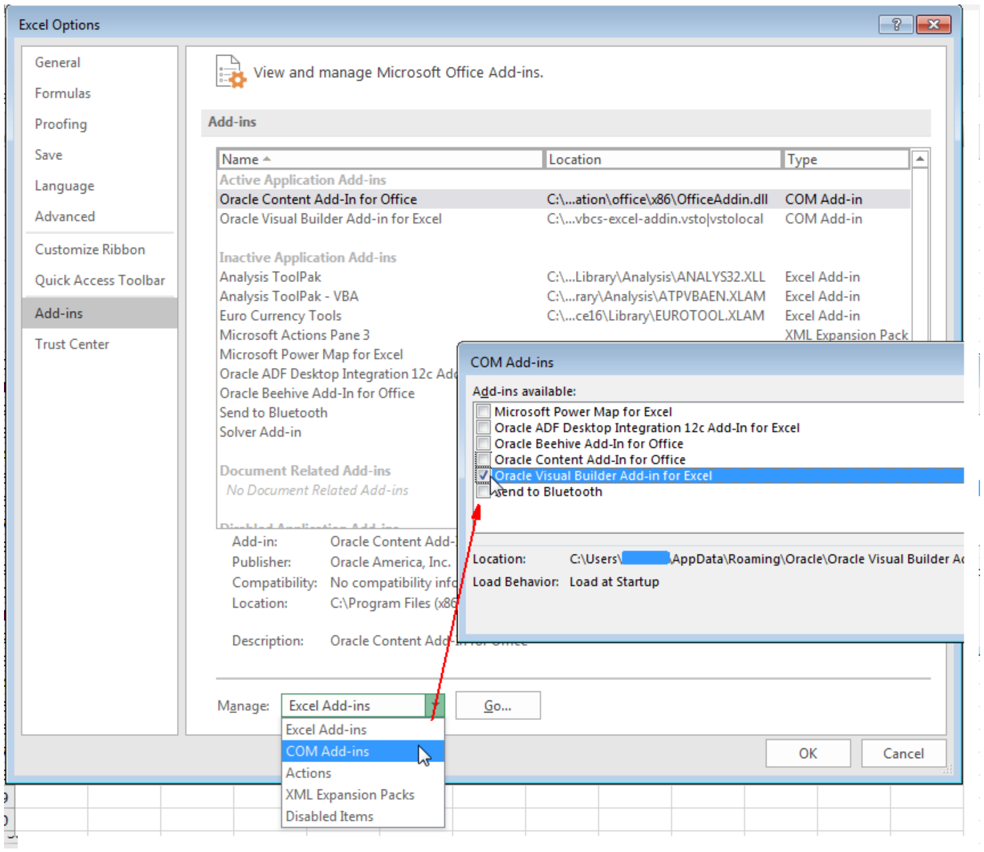
Switch	Description
`/help`	Displays a list of supported switches with description.
`/quiet`	Suppresses the interactive mode of the installer and does not install any missing prerequisite software. You can combine this switch with others to customize your silent installation.
`/designer 0|1`	Use this switch as follows: • `0` to disable the designer tools and install only the data tools. • `1` to enable the designer tools. This is the default installation option, unless a previous installation setting exists.
`/log` *`path`*	Runs the installer and directs the log output to the specified log file. The default log file location is `%TEMP%\vbcs\vbcs-installer-log.txt`.
`/roaming 0|1`	Use this switch as follows: • `0` to install the add-in to the local application data folder (`%localappdata%\Oracle\Oracle Visual Builder Add-in for Excel`). Use `/roaming 0` to install to the local application data folder during an upgrade from a prior installation that was installed to the roaming application data folder. • `1` to install the add-in to the user's roaming application data folder (`%appdata%\Oracle\Oracle Visual Builder Add-in for Excel`). This is the default installation location.

Troubleshooting Your Installation

If you run into issues when installing the add-in, follow these steps to check your environment and troubleshoot issues:

1. Download and run the Visual Builder Add-in for Excel - Client Health Check Tool (see How to use Visual Builder Add-in for Excel - Client Health Check Tool (Doc ID 2477792.1)).
2. Check the `%TEMP%\vbafe\vbafe-installer-log.txt` installation log.

3. If the preceding steps don't work, disable and re-enable the add-in:
   a. In Excel, click **File** > **Options** > **Add-Ins**.
   b. Select **COM Add-ins** in the Manage drop-down list and click **Go**.
   c. Deselect the **Oracle Visual Builder Add-in for Excel** check box and click **OK**.
   d. Restart Excel.
   e. Follow steps **a** to **d** to enable the add-in.



Name	Location	Type
Active Application Add-ins		
Oracle Content Add-In for Office	C:\...ation\office\x86\OfficeAddin.dll	COM Add-in
Oracle Visual Builder Add-in for Excel	C:\...vbcs-excel-addin.vsto|vstolocal	COM Add-in
Inactive Application Add-ins		
Analysis ToolPak	C:\...Library\Analysis\ANALYS32.XLL	Excel Add-in
Analysis ToolPak - VBA	C:\...rary\Analysis\ATPVBAEN.XLAM	Excel Add-in
Euro Currency Tools	C:\...ce16\Library\EUROTOOL.XLAM	Excel Add-in
Microsoft Actions Pane 3		XML Expansion Pack
Microsoft Power Map for Excel		
Oracle ADF Desktop Integration 12c Add		
Oracle Beehive Add-In for Office		
Send to Bluetooth		
Solver Add-in		
Document Related Add-ins		
No Document Related Add-ins		

Add-in: Oracle Content Add-
Publisher: Oracle America, Inc.
Compatibility: No compatibility info
Location: C:\Program Files (x86
Description: Oracle Content Add-

Manage: Excel Add-ins

COM Add-ins
Add-ins available:
- [ ] Microsoft Power Map for Excel
- [ ] Oracle ADF Desktop Integration 12c Add-In for Excel
- [ ] Oracle Beehive Add-In for Office
- [ ] Oracle Content Add-In for Office
- [✓] Oracle Visual Builder Add-in for Excel
- [ ] Send to Bluetooth

Location: C:\Users\[redacted]\AppData\Roaming\Oracle\Oracle Visual Builder A
Load Behavior: Load at Startup

4

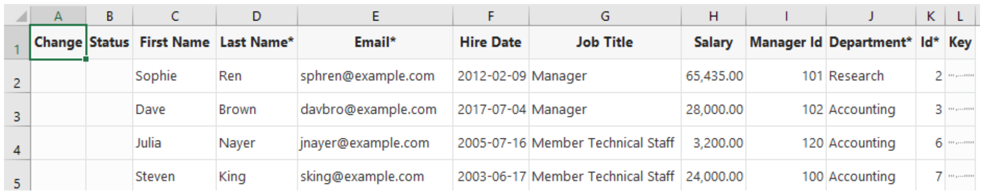
Create Layouts in an Excel Workbook

Start interacting with data in your business objects, first by creating a blank Excel workbook. You can then integrate the workbook with a REST service and create a layout to download data to a data table in the workbook.

Layouts

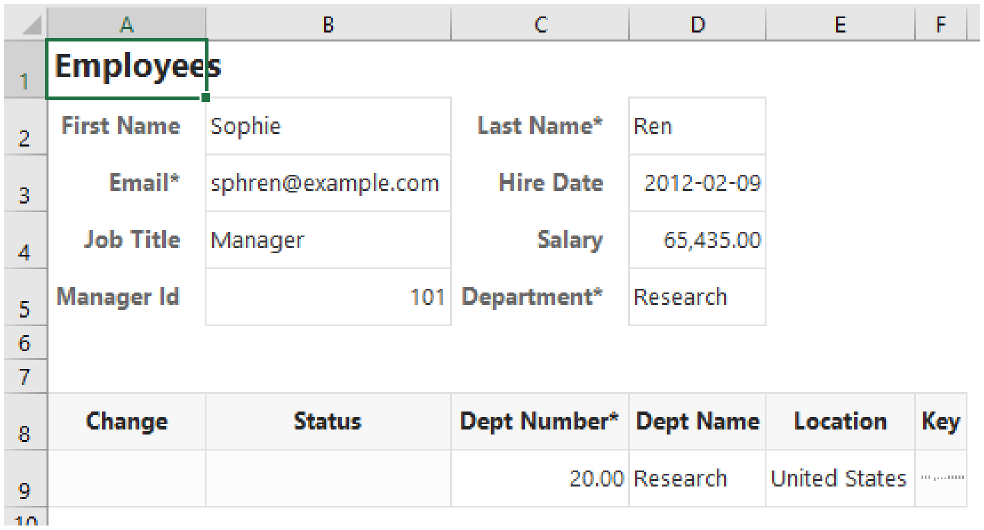
Oracle Visual Builder Add-in for Excel lets you create different layouts to work with data in an Excel worksheet. Layouts are a way to display a business object in an Excel worksheet. Each worksheet supports one of two layouts: **Table** or **Form-over-Table**.

- 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:



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

- 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 employee data in a Form-over-Table Layout, where the parent object's data (Employees) is shown in the form and the child object's data (Departments) is shown in the table:



	A	B	C	D	E	F
1	Employees					
2	First Name	Sophie	Last Name*	Ren		
3	Email*	sphren@example.com	Hire Date	2012-02-09		
4	Job Title	Manager	Salary	65,435.00		
5	Manager Id	101	Department*	Research		
6						
7						
8	Change	Status	Dept Number*	Dept Name	Location	Key
9			20.00	Research	United States	[illegible]
10						

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`, 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/`.

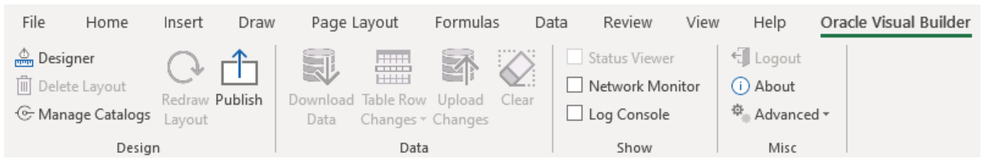
You can provide the service description document when you create a layout by clicking **Designer** in the Oracle Visual Builder tab, as described in subsequent sections. You can also provide the service description document by clicking **Manage Catalogs** in the Oracle Visual Builder tab (see Edit Service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

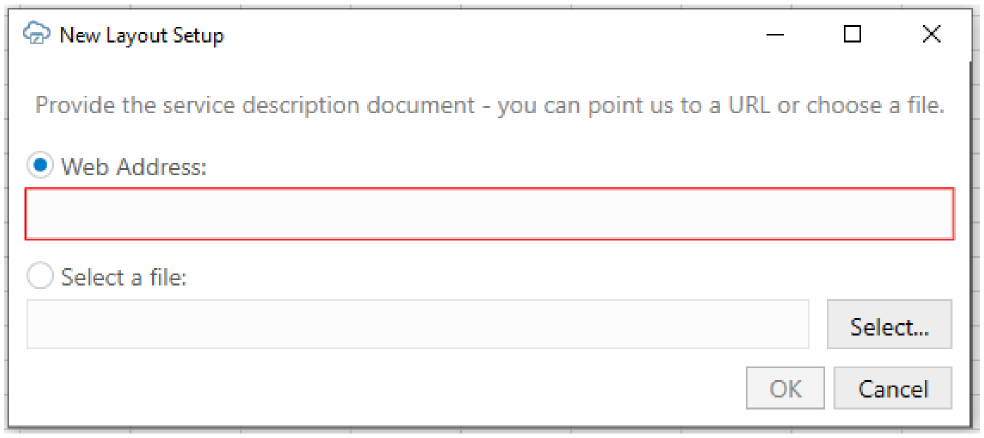
Create a Table layout in the Excel worksheet when you want to view and edit data from a REST service in a tabular format.

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**.
5. When prompted, provide the service description document. Use the **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. Use the **Select a file** option if the service description document is a local

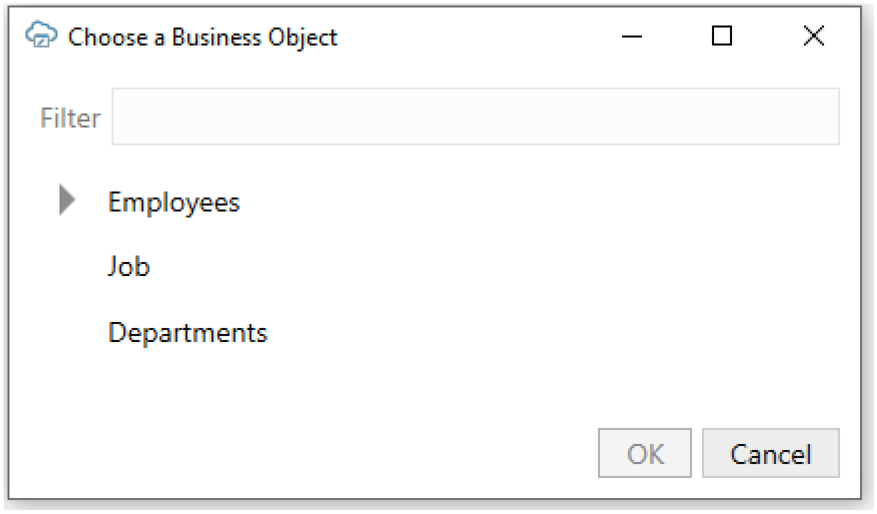
file on your computer.


New Layout Setup
Provide the service description document - you can point us to a URL or choose a file.
Web Address:
Select a file:
Select...
OK
Cancel

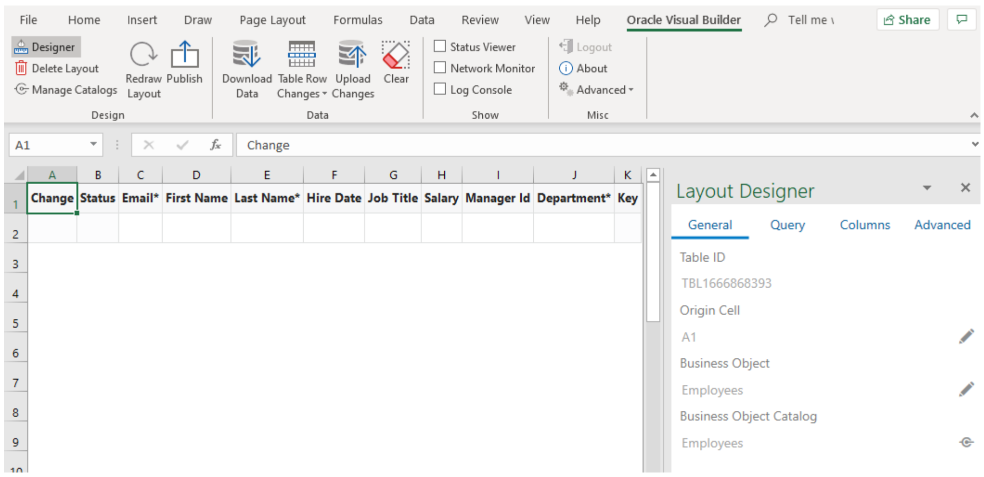
Tip:

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/latest/invoices/describe`.

If multiple business objects are found, the Choose a Business Object window prompts you to choose from the available business objects.



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.



A1 | Change

A	B	C	D	E	F	G	H	I	J	K
Change	Status	Email*	First Name	Last Name*	Hire Date	Job Title	Salary	Manager Id	Department*	Key

Table ID
TBL1666868393
Origin Cell
A1
Business Object
Employees
Business Object Catalog
Employees

6. 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.
   After making changes, click **Redraw Layout** in the ribbon tab to see your changes reflected in the worksheet.
7. 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

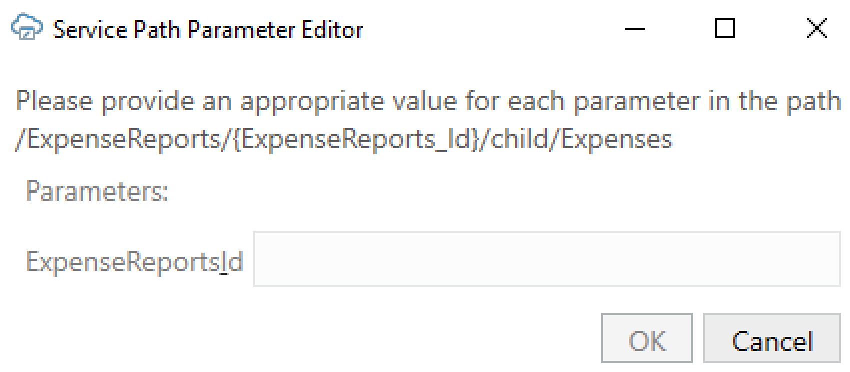
Some service paths include path parameters. The add-in provides support for configuring a Table layout using a parameterized service path. It automatically extracts the path parameters and prompts the user to provide the corresponding values at download time.

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.



Please provide an appropriate value for each parameter in the path /ExpenseReports/{ExpenseReports_Id}/child/Expenses

Parameters:

ExpenseReportsId

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, `1232456`, 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.

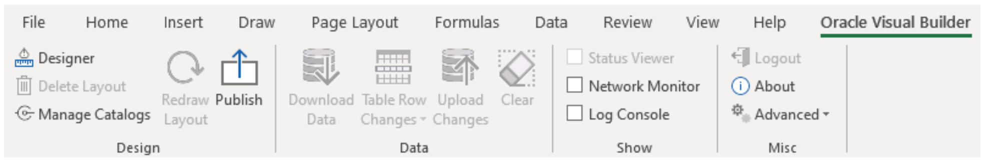
Note:

A parent-child relationship at the service level requires:

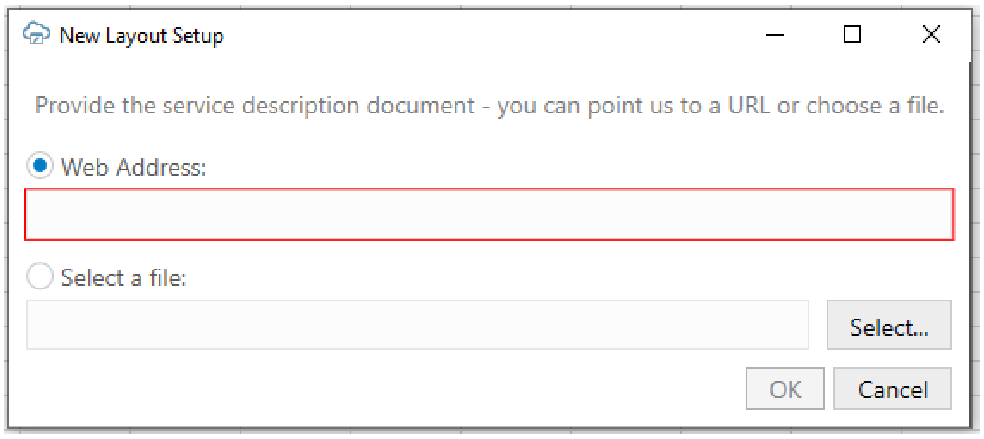
- A parent service path, for example, `fscmRestApi/resources/latest/invoices`
- A child service path with a parameter, for example, `fscmRestApi/resources/latest/invoices/{invoice-id}/child/invoicelines`

In your workbook, both business objects must be declared in the same catalog. Continuing our example, `invoicelines` must appear as a child of `invoices`. The add-in uses the primary ID of the parent row and inserts it into the child's path. The workbook service must be able to support operations on these paths.

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**.
5. When prompted, provide the service description document. Use the **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. Use the **Select a file** option if the service description document is a local file on your computer.


New Layout Setup

Provide the service description document - you can point us to a URL or choose a file.

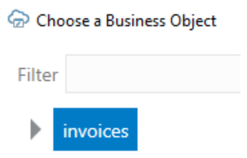
[✓ Web Address:]

Select a file:

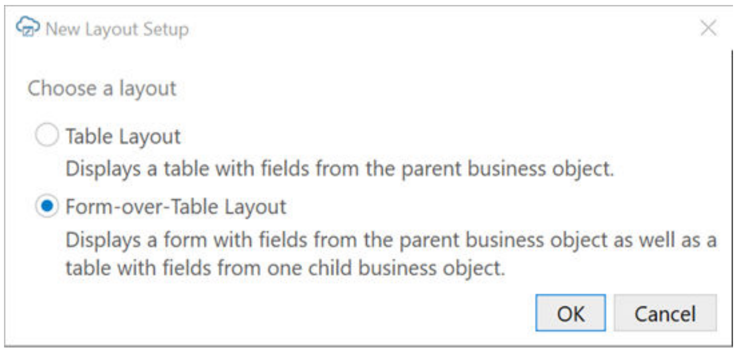
Tip:

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/latest/invoices/describe`.

When your OpenAPI service description includes multiple business objects, a window similar to the following prompts you to pick the business object you want to use in the Form-over-Table layout:



6. Choose a business object where a parent-child relationship exists (such as `invoices`), choose **Form-over-Table Layout** in the New Layout Setup window that appears, and click **OK**.



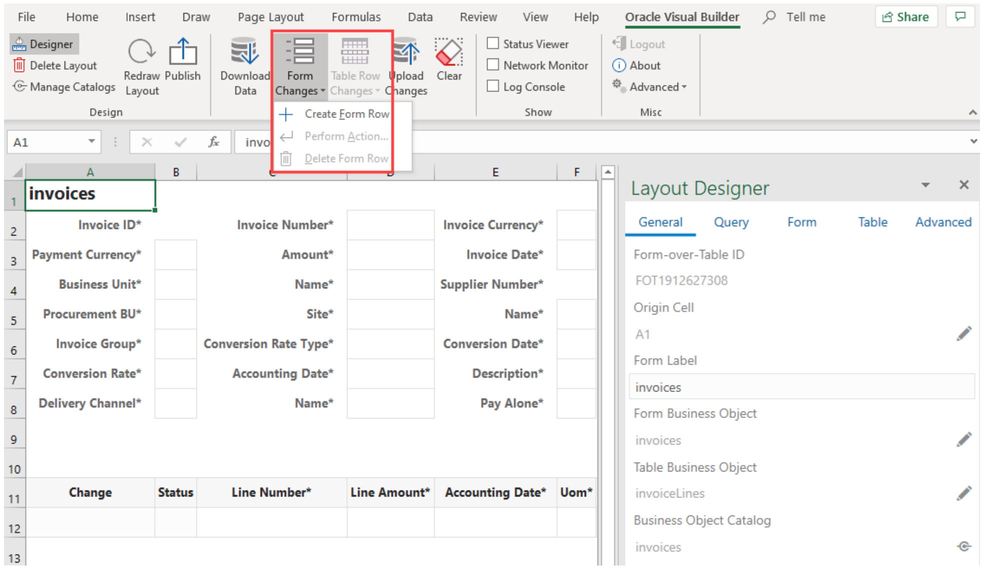
Choose a layout

( ) Table Layout
Displays a table with fields from the parent business object.

(●) Form-over-Table Layout
Displays a form with fields from the parent business object as well as a table with fields from one child business object.

If there's more than one child business object, select the child to use in the Form-over-Table layout and click **OK**. This example uses the `invoiceLines` child business object.

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:



Form-over-Table ID
FOT1912627308
Origin Cell
A1
Form Label
invoices
Form Business Object
invoices
Table Business Object
invoiceLines
Business Object Catalog
invoices

If the parent business object (in this case, `invoices`) 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*).

7. Customize the form and table by modifying the automatically populated properties in the Layout Designer. Click **Redraw Layout** to see changes you make in the Layout Designer reflected in the form and table.
   The Form tab enables you to add or remove fields to the form. The Table tab performs a similar function for the table under the form.

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

8. Optionally, configure the workbook to limit the data that the add-in downloads, as described in Configure Search Options for Download. If, for example, 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*.

Add Descriptive Flexfields to a Layout

The Oracle Visual Builder Add-in for Excel supports the use of descriptive flexfields (DFF) in your layout's forms and tables. DFFs expand to display additional fields for tracking supplemental information. These fields are supported in REST services using Oracle ADF Business Components.

A DFF consists of a top-level field, or "discriminator", and a number of sub-fields called "segments". Segments may be global (available for all values in the discriminator) or context-sensitive (dynamic based on the value in the discriminator).

For example, an "Employees" REST resource may include a child DFF that defines regional information for employees. When added to a table, the DFF displays a Region column (for the discriminator) as well as additional columns for the DFF's segments. These columns could, for example, be used to capture extra regional information such as site and time zone. The DFF may also include context-sensitive columns such as a "Postal Code/Zip Code" column for employees in the North American region.

DFFs in Forms and Tables

Add DFFs to forms and tables using the Form Field Manager or Table Column Manager. See either Create a Table Layout in an Excel Workbook or Create a Form-over-Table Layout in an Excel Workbook.

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 composed of the child DFF resource name ("EmployeesDFF" in this example) followed by the discriminator field ID ("__FLEX_Context").

You can add multiple DFFs and place them anywhere in the form or table. .

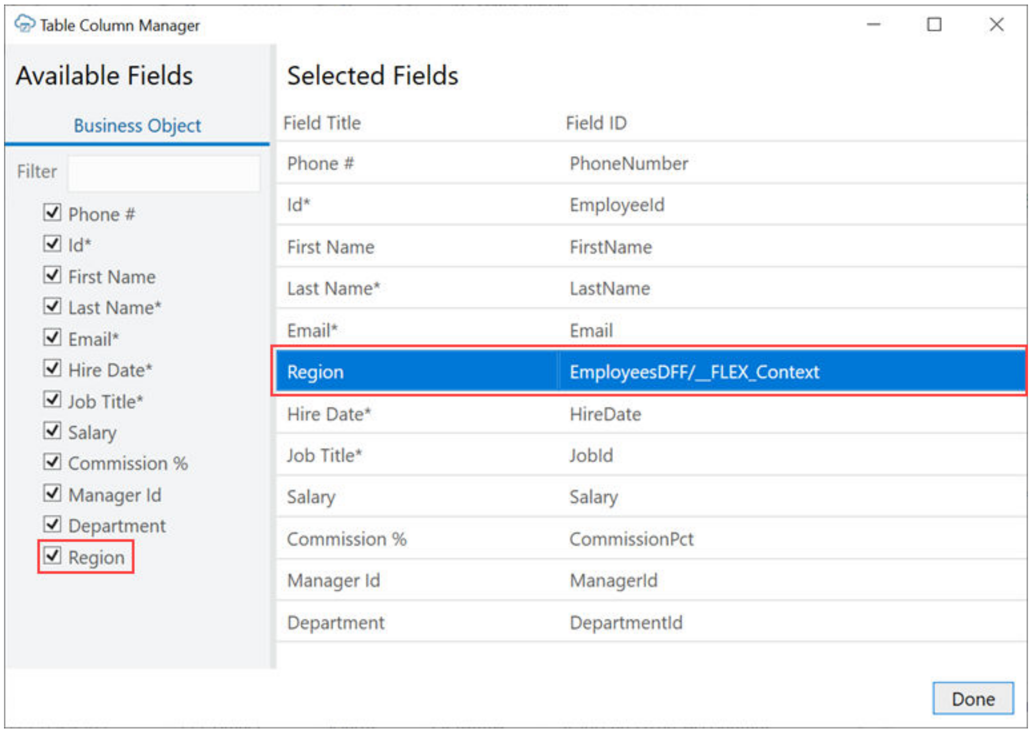


Table Column Manager

Available Fields

Business Object

Filter

- ☑ Phone #
- ☑ Id*
- ☑ First Name
- ☑ Last Name*
- ☑ Email*
- ☑ Hire Date*
- ☑ Job Title*
- ☑ Salary
- ☑ Commission %
- ☑ Manager Id
- ☑ Department
- ☑ Region

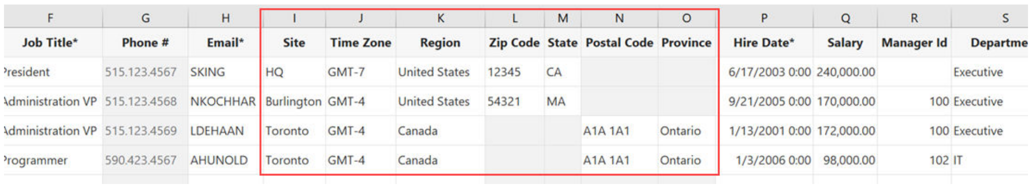
Selected Fields

Field Title	Field ID
Phone #	PhoneNumber
Id*	EmployeeId
First Name	FirstName
Last Name*	LastName
Email*	Email
Region	EmployeesDFF/__FLEX_Context
Hire Date*	HireDate
Job Title*	JobId
Salary	Salary
Commission %	CommissionPct
Manager Id	ManagerId
Department	DepartmentId

Done

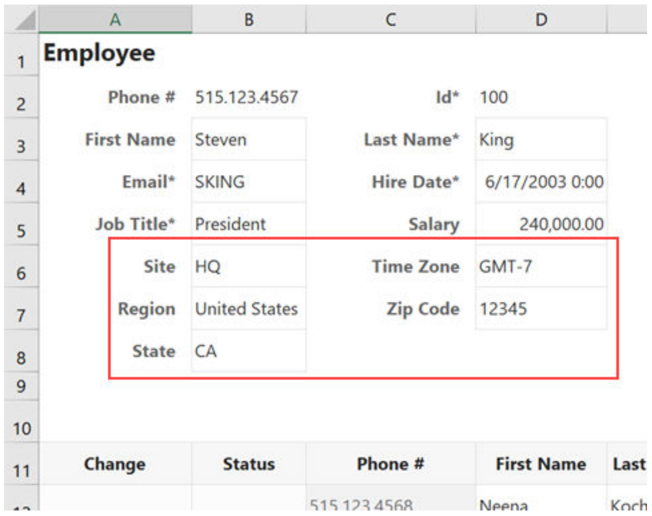
After you add a DFF to a layout, click **Redraw Layout** to expand the DFF. The associated segments appear in the following order: global segments, the discriminator, and context-sensitive segments.

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



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

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



	A	B	C	D
1	Employee			
2	Phone #	515.123.4567	Id*	100
3	First Name	Steven	Last Name*	King
4	Email*	SKING	Hire Date*	6/17/2003 0:00
5	Job Title*	President	Salary	240,000.00
6	Site	HQ	Time Zone	GMT-7
7	Region	United States	Zip Code	12345
8	State	CA		
9				
10				
11	Change	Status	Phone #	First Name	Last
			515.123.4568	Neena	Koch

Refreshing DFF Metadata

Because DFFs are configurable by customers and subject to change, metadata stored in the workbook for the DFF may become stale.

To clear metadata when publishing a workbook, select **Clear all layouts** from the Publish Workbook window. The add-in refreshes the DFF metadata during the first download operation performed after you open the workbook.

Limitations

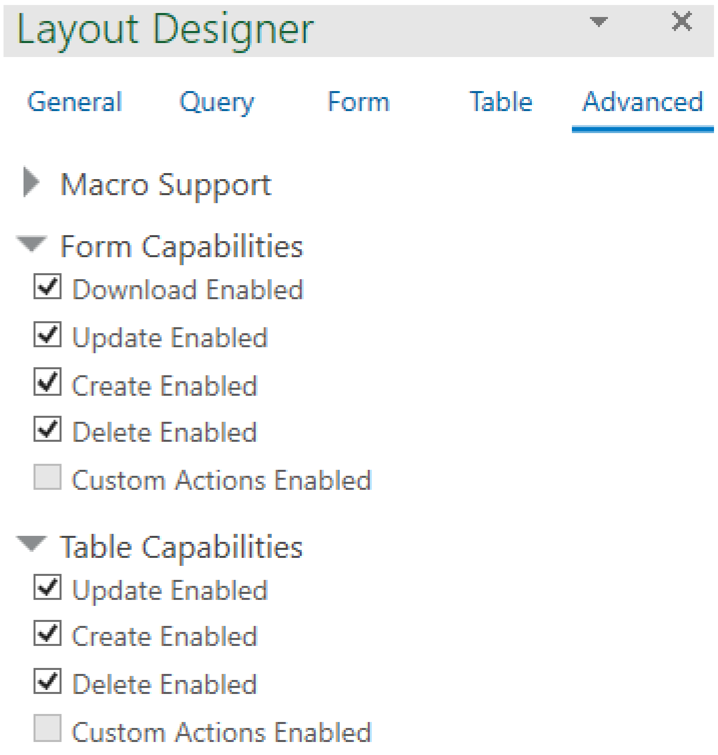
- A DFF child resource must have a one to one relationship with its parent resource.
- A DFF child resource can only contain a single discriminator field. Multiple discriminators are not supported.
- Changing the order of the global or context-sensitive segments is not supported.
- Hiding specific global or context-sensitive segments is not supported.
- Binding layouts directly to a DFF resource is not supported. Layouts must be bound to the parent resource of the DFF resource.
- Before you can add a DFF to a table, it must expose polymorphic business object metadata. This means that the OpenApi3 describe document 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 DFF, the "fields" and "expand" query parameters should not be manually configured in the "Search Parameters" in the "Query" tab of the designer.

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:



Form Capabilities
- [✓] Download Enabled
- [✓] Update Enabled
- [✓] Create Enabled
- [✓] Delete Enabled
- [ ] Custom Actions Enabled

Table Capabilities
- [✓] Update Enabled
- [✓] Create Enabled
- [✓] Delete Enabled
- [ ] Custom Actions Enabled

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).
4. Click **Redraw Layout**.

5

Edit Service Descriptions and Business Objects

Oracle Visual Builder Add-in for Excel provides editors to modify a workbook's service description, including its definition of one or more business objects. A service description that contains multiple business objects is sometimes known as a **Catalog**.

Using the Catalog editors, you can change the URL of the host that provides access to the REST service your workbook uses. You can also add or remove fields that the business object exposed by the service supports (any changes you make to a business object must be supported by the service that the workbook uses). If, for example, you add a field to a business object in the service description that the workbook uses, and the service does not support this field, errors occur when the workbook connects to the service.

Other examples of tasks that you can accomplish using these editors include:

- Edit field titles
- Configure the list of values for a field
- Change when a field is editable
- Configure pagination
- Adjust the field data types (Advanced)

While you can provide the service description document when you create a layout, using the Catalog editors lets you improve the service description in a variety of ways to enhance overall user experience.

Note:

When you edit the service description 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 service description in the workbook must be compatible with the service.

Click **Manage Catalogs** in the main Oracle Visual Builder tab to view the Catalog editors progressively. That is, you first access the editor that lets you edit the catalog, then access subsequent editors for business objects and business object fields that the catalog supports. 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:

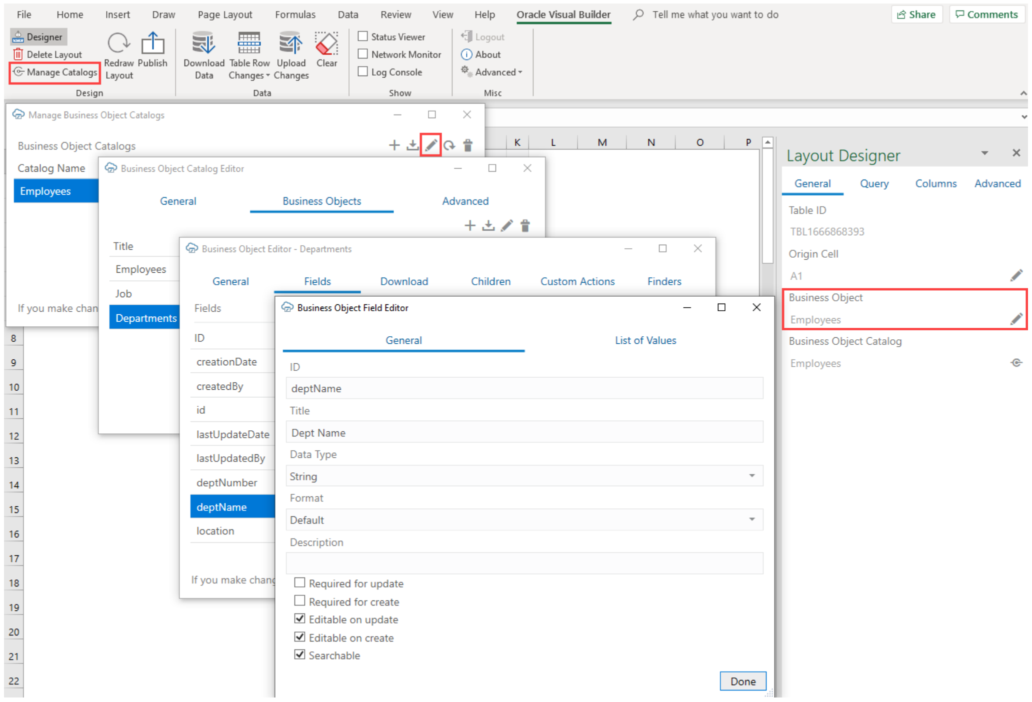


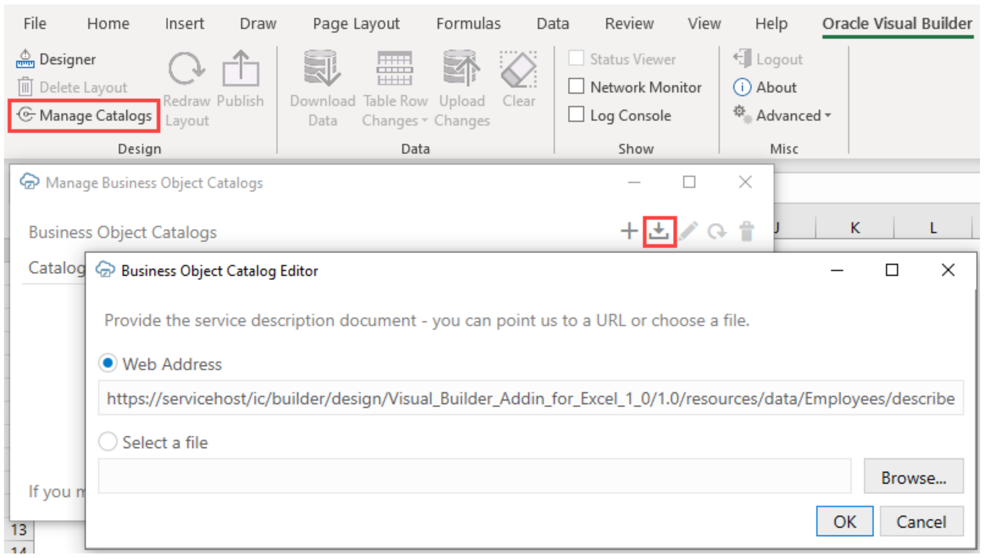
Table ID: TBL1666868393
Origin Cell: A1
Business Object: Employees
Business Object Catalog: Employees

ID: deptName
Title: Dept Name
Data Type: String
Format: Default
Description:

- [ ] Required for update
- [ ] Required for create
- [x] Editable on update
- [x] Editable on create
- [x] Searchable

To import a service description document, you have two options:

- To create a new catalog, particularly if the OpenAPI document has multiple business objects, use Manage Catalogs.



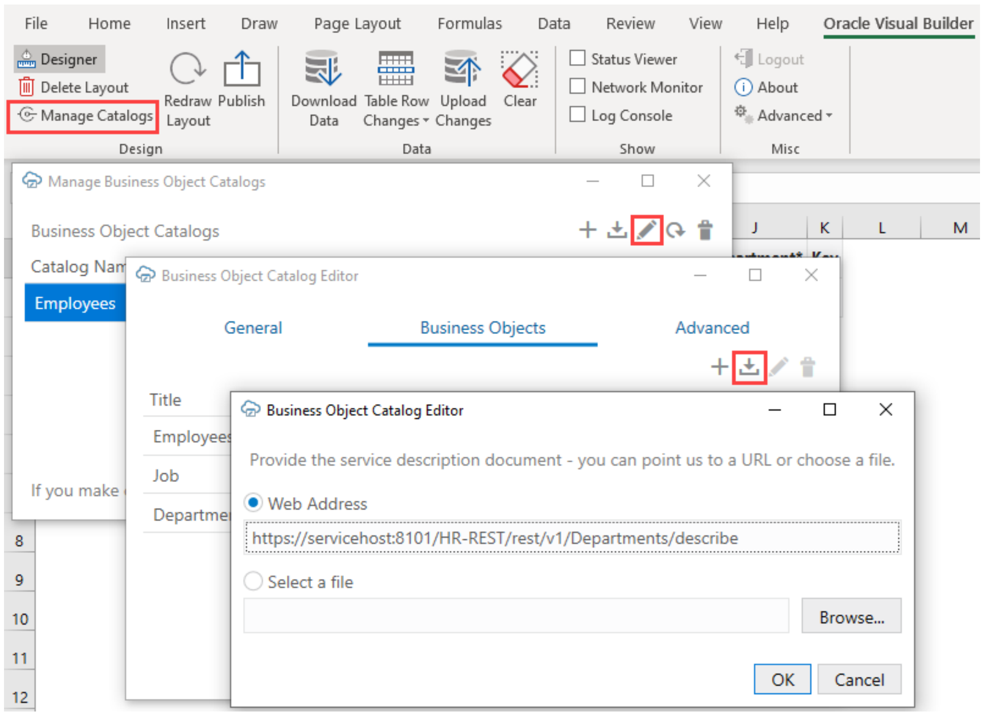
Provide the service description document - you can point us to a URL or choose a file.

(●) Web Address

https://servicehost/ic/builder/design/Visual_Builder_Addin_for_Excel_1_0/1.0/resources/data/Employees/describe

( ) Select a file

- To add a business object to an existing catalog (with the understanding that the new business object will be located at the same host/base path as the other business objects in that catalog), use the Business Object Catalog editors (see Add a Business Object to an Existing Catalog). This option is particularly useful when configuring a list of values.



Business Object Catalog Editor

Provide the service description document - you can point us to a URL or choose a file.

(●) Web Address

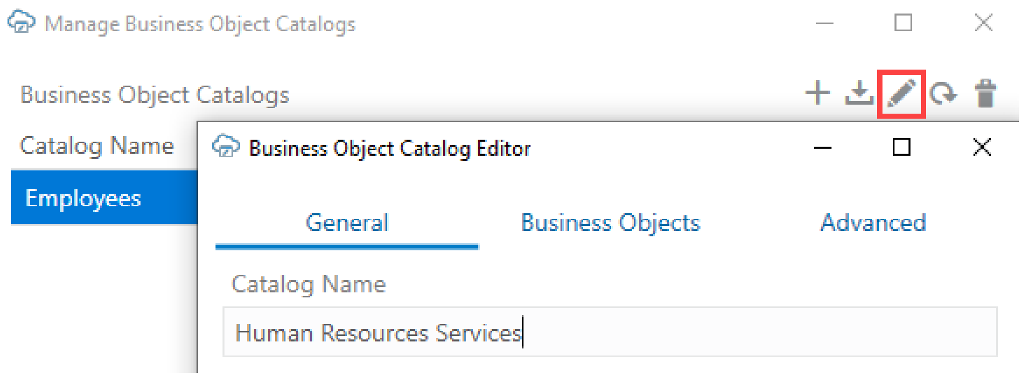
https://servicehost:8101/HR-REST/rest/v1/Departments/describe

( ) Select a file

Browse...

OK Cancel

After you import a service description document 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`.



Catalog Name

Human Resources Services

After making changes in the Business Object Catalog Editor, such as changing the service host or details of a business object, click **Redraw Layout** in the Oracle Visual Builder tab to see your changes reflected in the worksheet.

Create a Service Description in Oracle Visual Builder

If your target service does not provide an OpenAPI-compliant service description, you can create one in Visual Builder.

1. Sign in to your Visual Builder account in the Oracle Cloud.
2. Create a new visual application or open an existing visual application.
3. Create a service connection from an endpoint. See Create a Service Connection from an Endpoint.
4. Configure the service with the details that you will need in the Excel workbook integration.
5. In a Visual Builder web application, test the service connection by creating a sample page and table that retrieves data from the service.
6. In the source view of the visual application, locate and download the `service.json` file to your computer.

   The `service.json` file contains the OpenAPI service description for the REST service that you connected to from your visual application.

Add a Business Object to an Existing Catalog

If your service definition is missing a business object, you can add a new business object description to an existing service description:

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 **Business Objects**, then click the Import a Business Object from a Service Description icon.
4. Provide the URL or a file that contains the OpenAPI description of the new business object and click **OK**.
5. Click **Done** first in the **Business Object Catalog Editor**, then in the **Manage Business Object Catalogs** window.

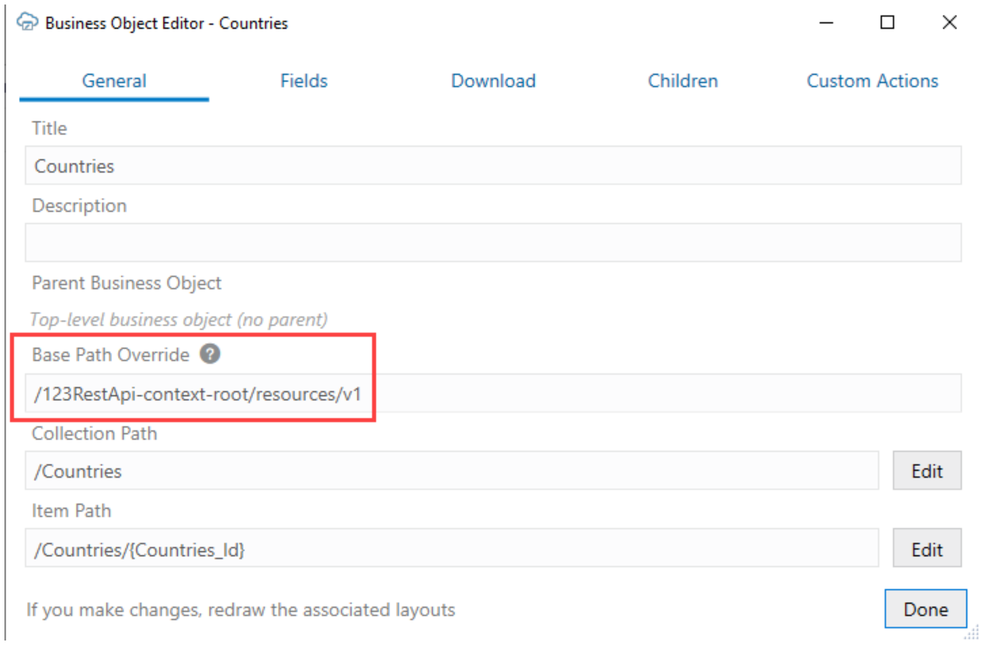
Override a Business Object's Base Path

Typically, a Business Object Catalog holds business objects that share a base path (say, `/abcRestApi-context-root/resources/v1`). But if your list of values' `operationRef` points to another business object (say, `/123RestApi-context-root/resources/v1`), you can configure the business object to use the different base path. The add-in will then use this base path when making REST requests for the business object.

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. Use the newly created business object as the **Referenced Business Object** for your list of values. See Configure a List of Values for a Business Object Field.
3. 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**.



Title: Countries
Description:
Parent Business Object: *Top-level business object (no parent)*
Base Path Override: /123RestApi-context-root/resources/v1
Collection Path: /Countries
Item Path: /Countries/{Countries_Id}
If you make changes, redraw the associated layouts

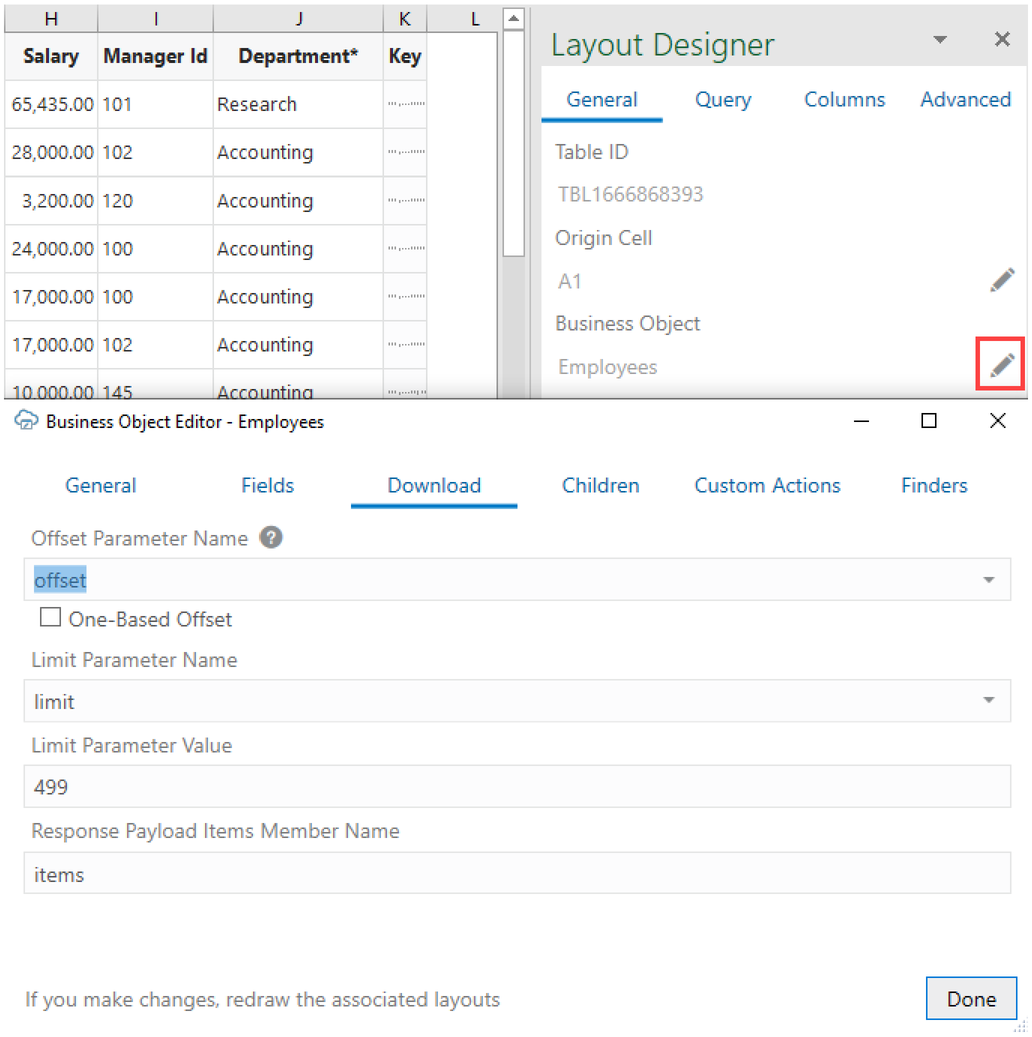
The add-in assumes that no extra authentication is required by business objects with a base path override.

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.

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



H	I	J	K	L
Salary	Manager Id	Department*	Key	
65,435.00	101	Research		
28,000.00	102	Accounting		
3,200.00	120	Accounting		
24,000.00	100	Accounting		
17,000.00	100	Accounting		
17,000.00	102	Accounting		
10,000.00	145	Accounting		

Layout Designer

General Query Columns Advanced

Table ID
TBL1666868393
Origin Cell
A1
Business Object
Employees

Business Object Editor - Employees

General Fields Download Children Custom Actions Finders

Offset Parameter Name: offset
One-Based Offset: ☐
Limit Parameter Name: limit
Limit Parameter Value: 499
Response Payload Items Member Name: items

If you make changes, redraw the associated layouts

Done

Configure the following as required:

- **Offset Parameter Name**: 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 500, assuming the limit value is 499.
- **One-Based Offset**: Controls whether the service starts counting from one. If this check box is cleared, the service assumes the service starts counting from zero.
- **Limit Parameter Name**: 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 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.

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.

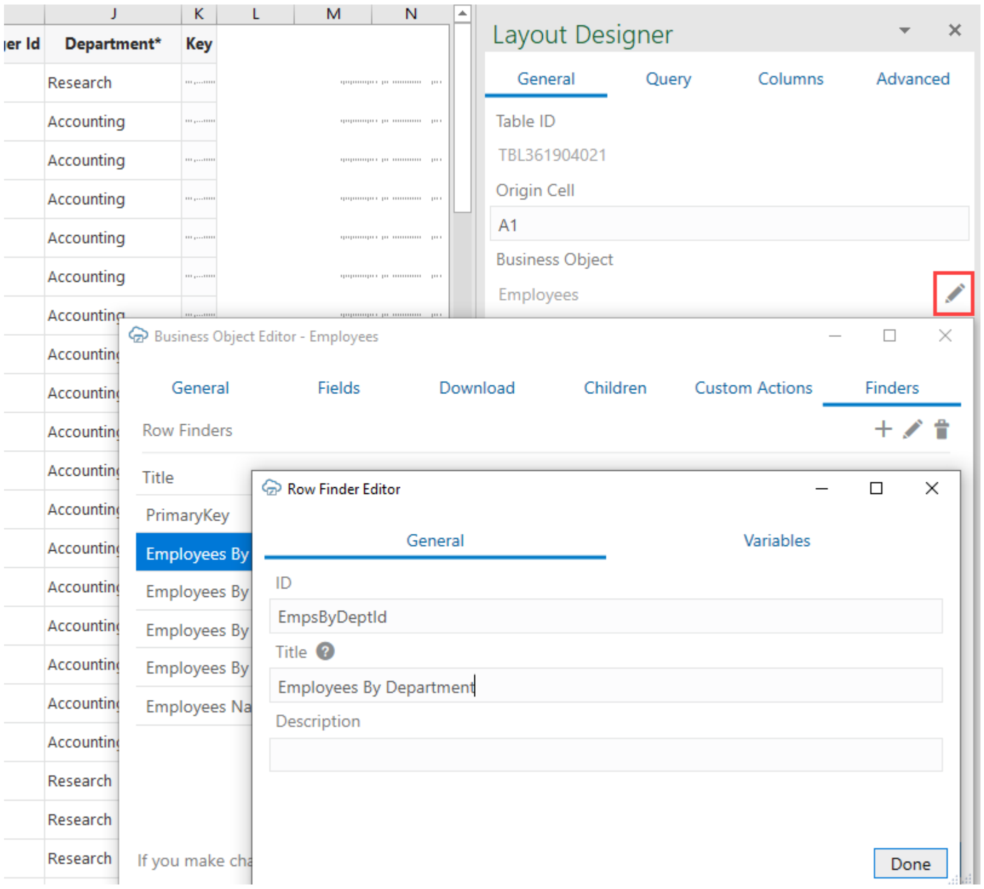
Configure Row Finders for a Business Object

For Oracle business object REST API services that support row finders, you can add, remove, and update row finders that are available for a business object. You can use row finders to configure search options when data is downloaded to an Excel workbook.

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. Add or remove row finders as required. You can also edit a row finder's details, like changing the title to be more readable:



Layout Designer

General | Query | Columns | Advanced

Table ID: TBL361904021

Origin Cell: A1

Business Object: Employees

Business Object Editor - Employees

General | Fields | Download | Children | Custom Actions | Finders

Row Finders

Row Finder Editor

General | Variables

ID: EmpsByDeptId

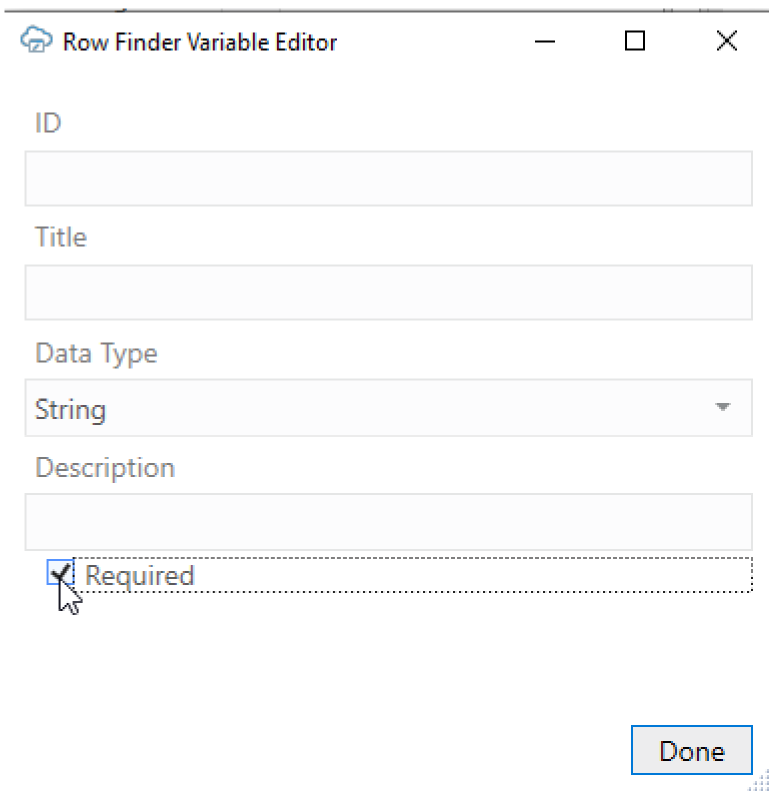
Title: Employees By Department

Description:

Done

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 to the finder, click the **Variables** tab to add, 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. 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

A business object catalog is generated when a workbook is first configured to integrate with a service, based on the service description that you provide when creating a layout. If the 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, row finders, or business objects 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 an important property, such as a required value for a field, has a change that you'd like to readily use in your workbook.

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. Provide the URL or a file that contains the new service description.
5. Click **OK**, then **Yes** when prompted to confirm the overwrite.
6. Click **OK**, then **Done**.
7. Refresh all layouts in the workbook before you proceed to other tasks.

If the new catalog has new business objects, those objects are added to the catalog. Existing business objects are refreshed with the corresponding ones in the new 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.

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**.
3. Click **Redraw Changes**.

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

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 and is set to the default when a layout is first created in a workbook.

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.

Here are some things to keep in mind when configuring search options:

- The search properties that you can configure depend on the type of service that you integrate your Excel workbook with. For example, all properties are available to use when your workbook integrates with an Oracle business object REST API service; when you use Oracle REST Data Services, you can configure Search and Search Parameters properties only.
- Configure values for the Search property when you want to provide users with the option to enter search terms to filter the data that they download from the Oracle business object REST API service or Oracle REST Data Services.
  Consider, for example, a data table that downloads data about employees. In this scenario, you can add the Department name business object field as a search field to the Search property, so that users can enter search criteria to download the employee records of those employees who belong to the department that matches the search criteria. You can enter multiple fields to the Search property so that users can enter multiple terms in their search.
- Configure a value for the Row Finder property if the Oracle business object REST API service you connect to supports finders. If you select a value for the Row Finder property, the add-in uses it during download. A row finder may also have variables. If the finder you choose for the Row Finder property specifies variables, the add-in prompts users to provide variable values during download.
- Add values for Search Parameters if the service that you download data from supports the addition of various parameters in the query portion string.

Depending on the service you use, you can configure one or more of these properties. If you configure values for all properties, the add-in prompts the user to enter values supported by the Row Finder property before it prompts the user to enter values for the Search property. Once the user enters the requested values, the add-in downloads data based on the user's input.

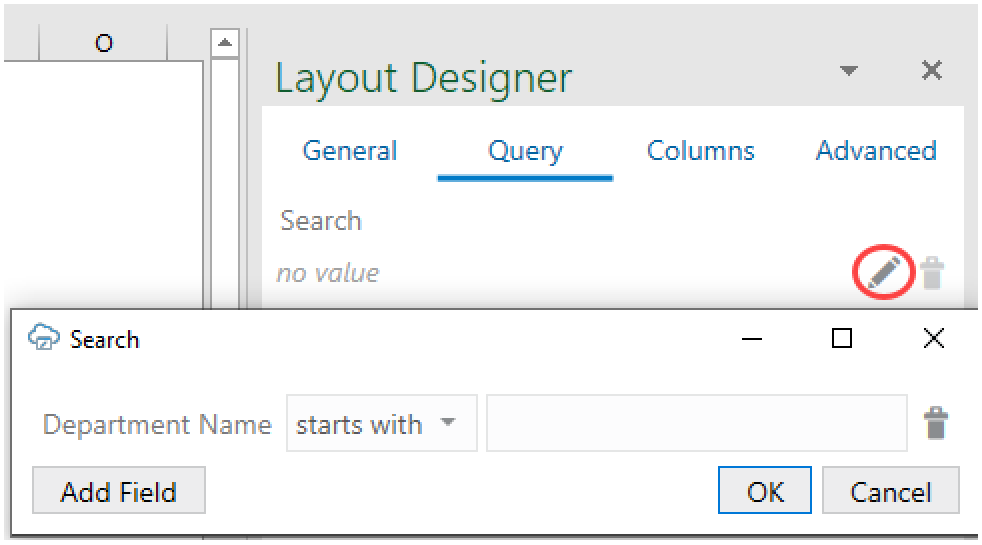
Tip:

If you are troubleshooting 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.

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 the workbook to enable a user to specify search values to limit the data that the add-in downloads to the workbook.

1. In the Excel ribbon, click **Designer**.
2. In the Layout Designer's Query tab, click the Edit icon next to the Search property to open the Available Business Object Fields window.
3. Select the business object field that you want to enable users to enter search terms for. For example, select **Department Name** and click **OK** if you want to enable users to search on employees by department, as shown here:

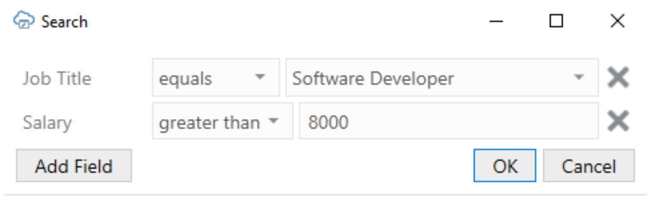
   Search

   no value

   Search

   Department Name | starts with

   To add more fields for complex searches, click **Add Field**. For example, you might want to search for employees with a job title of Software Developer, but only those who earn more than 8000. In this case, first select **Job Title**, then click **Add Field** and select **Salary**. Finally, define the search parameters as follows:

   Job Title | equals | Software Developer

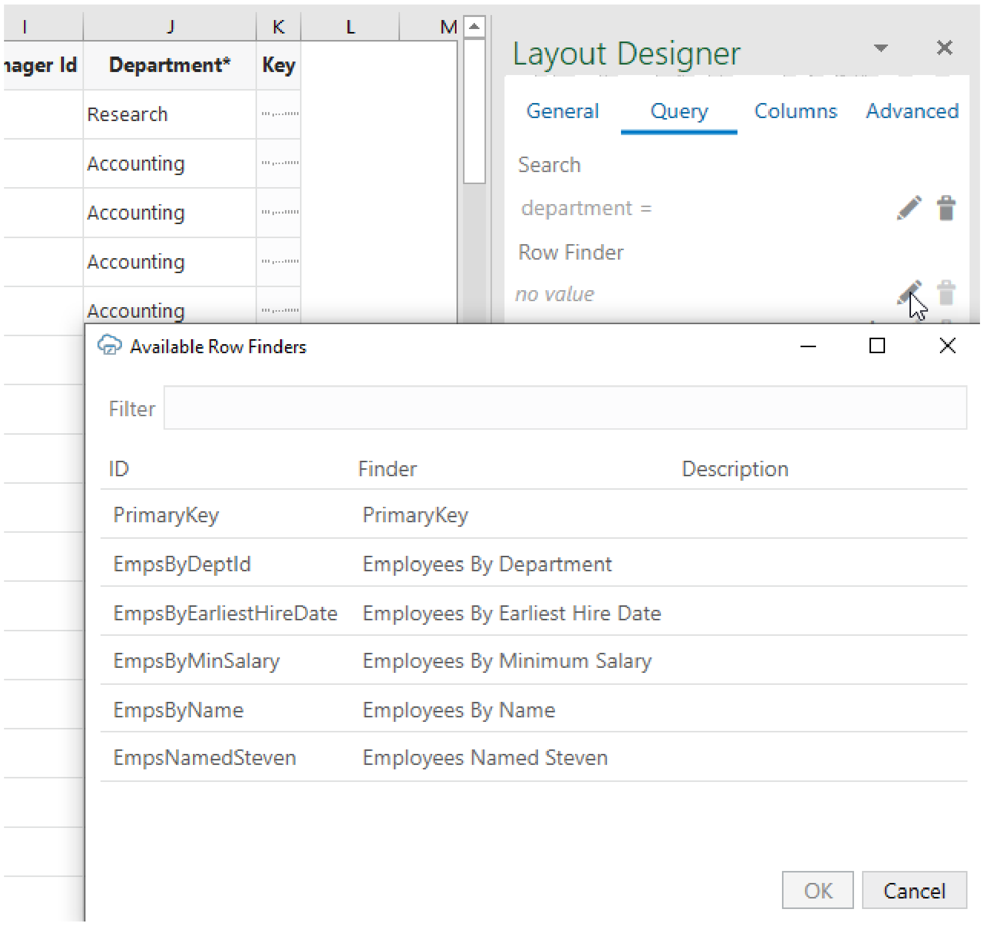
   Salary | greater than | 8000

   All searchable fields are available for you to choose from. You can adjust which fields show by selecting or deselecting **Searchable** in the Business Object Field Editor. Make sure that the service supports searching on that field.
4. 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:



I	J	K
ager Id	Department*	Key
	Research	
	Accounting	
	Accounting	
	Accounting	
	Accounting	

ID	Finder	Description
PrimaryKey	PrimaryKey	
EmpsByDeptId	Employees By Department	
EmpsByEarliestHireDate	Employees By Earliest Hire Date	
EmpsByMinSalary	Employees By Minimum Salary	
EmpsByName	Employees By Name	
EmpsNamedSteven	Employees Named Steven	

For details on configuring available row finders, see Configure Row Finders for a Business Object.

3. Click **OK** to close the open windows.

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

You can specify additional search parameters to add to the REST endpoint URL on download.

Many REST services support the addition of various parameters in the query string portion of the URL, as in this example:

```
GET…/orderReleaseLines?q=ID=1234
```

where:

- q is an optional parameter supported by the orderReleaseLines service
- `ID` is the name of a field that supports query

- `1234` is the search value

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 (=).

The add-in does not know which parameters are useful for search. Likewise, the add-in does not know the proper syntax for the parameter values. So, you will need to 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.

There is no validation in this editor or at download time. If you enter invalid information, you may get a bad request error.

When the user clicks the Download Data button, the search parameters are added to the appropriate URL along with the other search options (if applicable).

The add-in applies URL encoding to the parameter value at download time. So, you should not enter URL-encoded values. The search parameter name is not encoded.

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

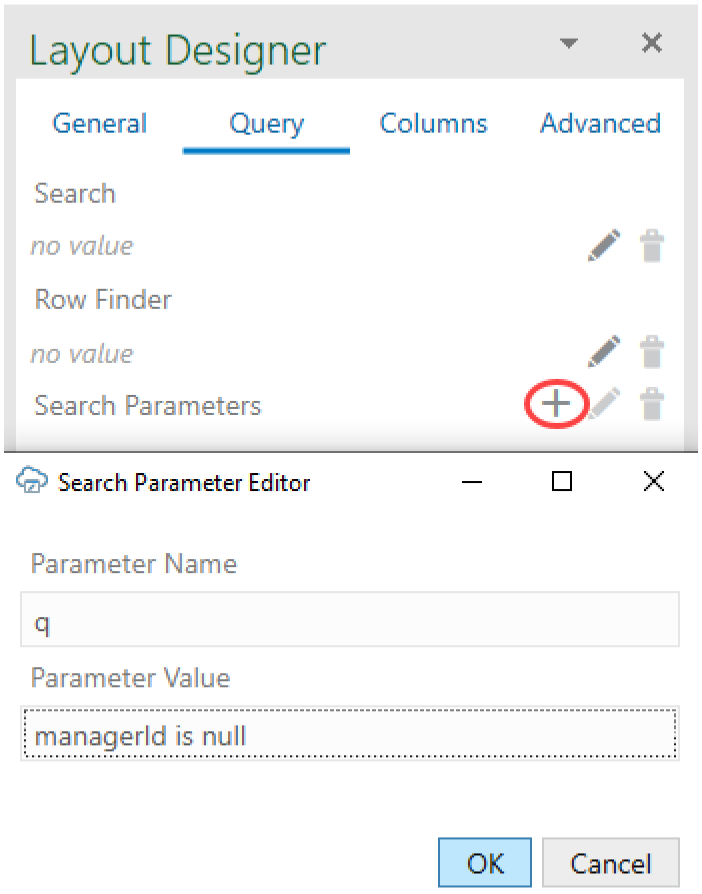
```
q=((firstName LIKE '*es*') or ((hireDate< "2001-01-13") and (department =
10)))
```

The Search Parameters property does not support "dynamic" search values. Using the above example, the value, 1234, is specified in the Layout Designer as a constant.

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.

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.



3. Click **OK** to close the open windows.

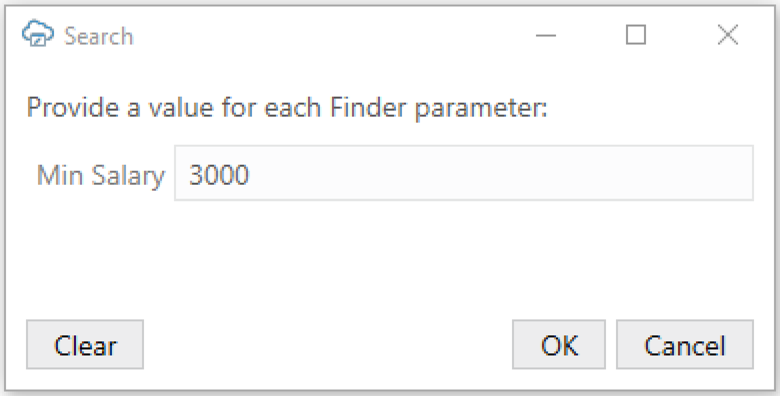
Download Behavior in Layouts that Use Search and Row Finders

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:

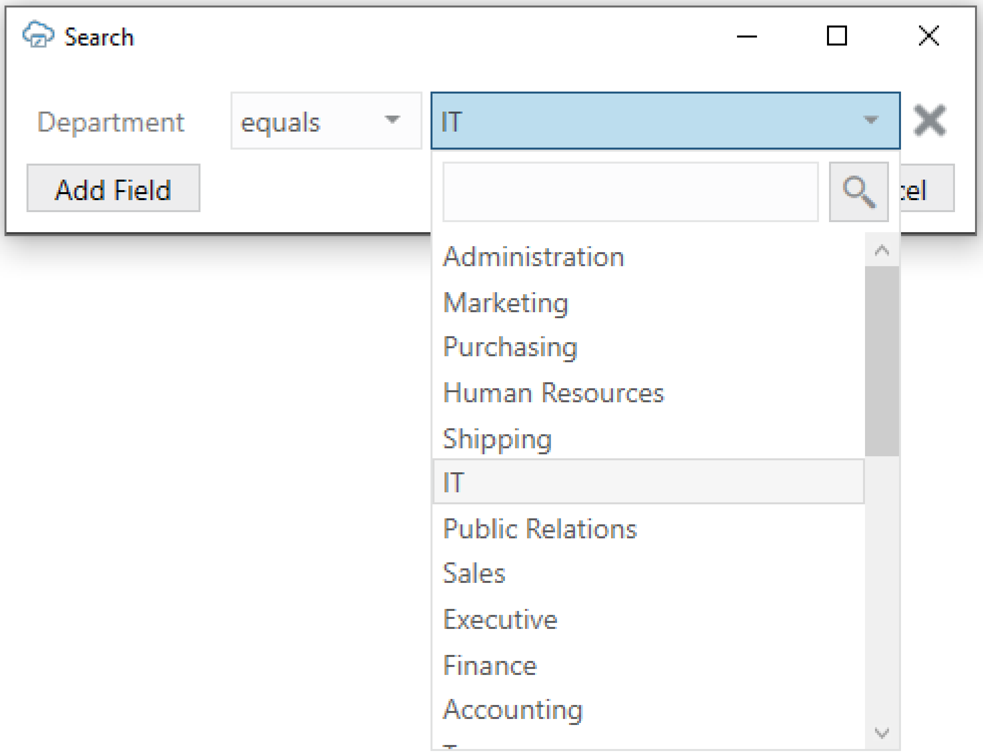
- If there are no parameters, there is no prompt.
- If there is exactly one parameter, a prompt appears that allows the user to specify a value for the parameter.
- If the finder has more than one parameter, a prompt appears that allows the user to specify values for each parameter.

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

Workbooks that integrate with Oracle business object REST API services can allow the user to perform custom actions on rows of data. For example, an expense report business object might support a custom action called "reject". Users could download many expense reports and reject many of them in a single upload.

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

- Custom actions that correspond to view object methods (as opposed to view object row methods) are not currently supported.
- 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.
- The response payload from a successful custom action invocation is ignored by the add-in.
- 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.

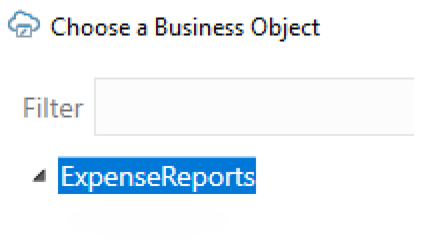
Use Custom Actions

Let's take a look at how you can invoke a custom action in your Excel workbook, for example, the "`reject`" custom action that's exposed by an expense report business object to reject expenses.

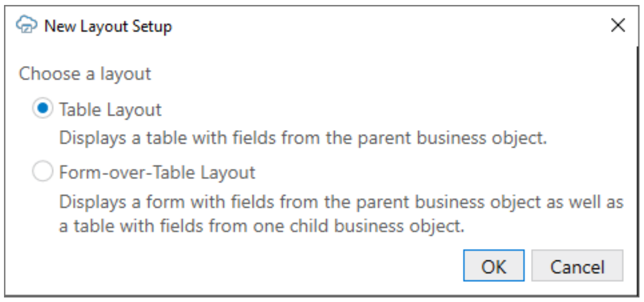
Note:

Your REST API must expose custom actions on the item path for your resource, something like a POST to `/contextRoot/v1/myBusObj/{itemId}/action/doMyAction`. Actions defined on the collection path are not supported.

1. Create a layout for the custom action, for example, a Table layout for the "`reject`" custom action.
   a. In the Oracle Visual Builder tab, click **Designer**.
   b. When prompted, provide the service description document.
   c. Choose a business object, for example, `ExpenseReports`.



d. Select a layout and click **OK**.


New Layout Setup

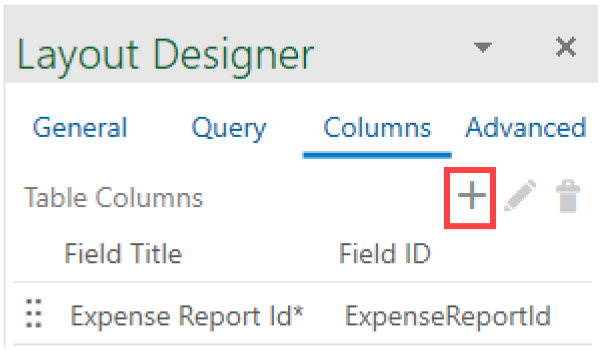
Choose a layout

- (●) Table Layout
  Displays a table with fields from the parent business object.
- ( ) Form-over-Table Layout
  Displays a form with fields from the parent business object as well as a table with fields from one child business object.

In our example, a Table layout that includes column headers and a placeholder data row is created in the Excel workbook.

2. Create new columns in the layout to pass payload field values that the custom action needs. In this example, we'll add columns to show the Reason and Notes payload fields.

   a. In the Layout Designer, click **Columns**, then click Manage Columns (+).



Field Title	Field ID
Expense Report Id*	ExpenseReportId

   b. Click **Custom Actions** in the Table Column Manager.

   c. 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.

   d. In the Available Fields pane, select the payload fields (**Reason** and **Notes**).

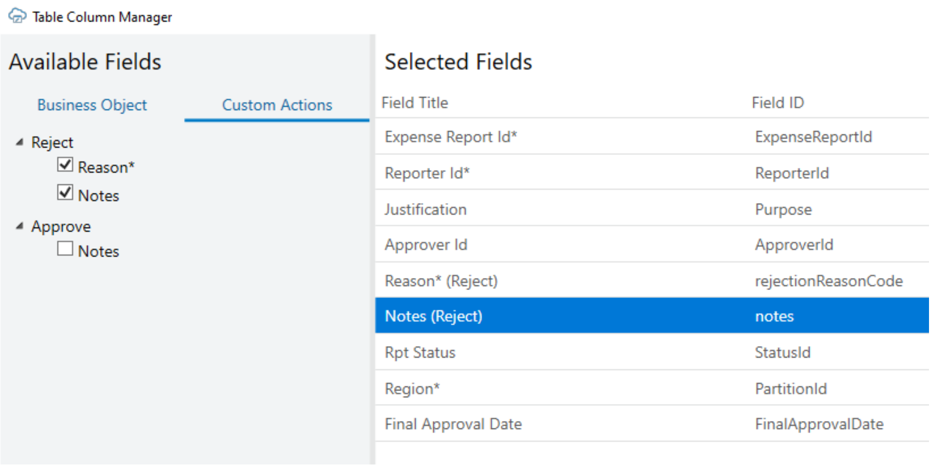


Table Column Manager

Available Fields

Business Object | Custom Actions

- Reject
  - [✓] Reason*
  - [✓] Notes
- Approve
  - [ ] Notes

Selected Fields

Field Title	Field ID
Expense Report Id*	ExpenseReportId
Reporter Id*	ReporterId
Justification	Purpose
Approver Id	ApproverId
Reason* (Reject)	rejectionReasonCode
Notes (Reject)	notes
Rpt Status	StatusId
Region*	PartitionId
Final Approval Date	FinalApprovalDate

e. Click **Done**.

f. Click **Redraw Layout**.

The newly added columns show in the worksheet table.

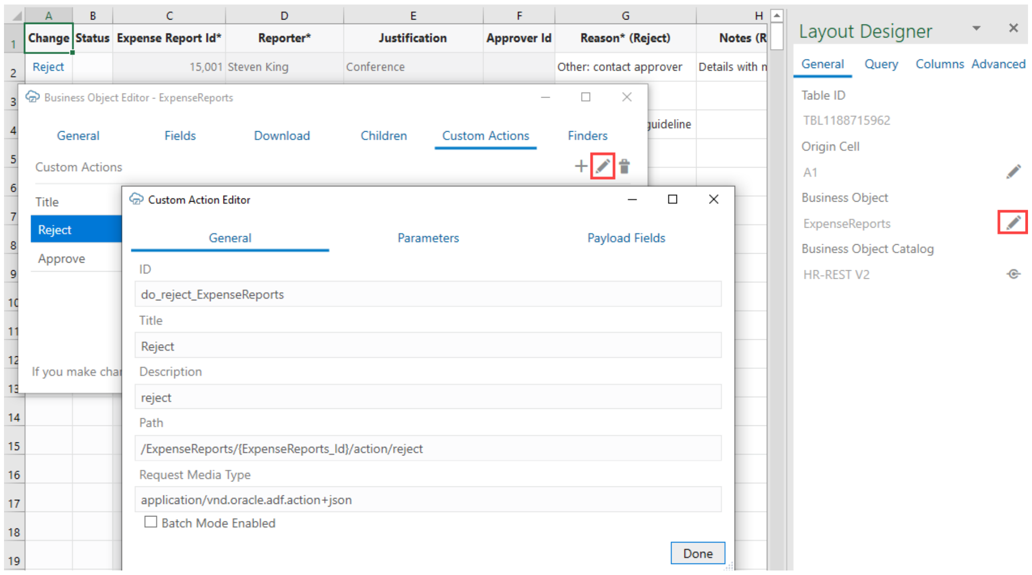
3. Mark rows for the custom action. You can mark rows by entering values in the payload field columns or by using the ribbon command. See Perform Custom Actions in Table and Form-over-Table Layouts in *Managing Data Using Oracle Visual Builder Add-in for Excel*.

4. Click **Upload Changes** to send your changes to the REST endpoint and invoke the custom action on the marked rows. See Upload Changes for Custom Actions.

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.

Properties such as Title (which appears in the UI that the workbook user sees) and Description can be edited as needed in the Custom Action Editor. Other properties of the custom action should generally be left as is.



Change	Status	Expense Report Id*	Reporter*	Justification	Approver Id	Reason* (Reject)	Notes (R
Reject		15,001	Steven King	Conference		Other: contact approver	Details with n

Business Object Editor - ExpenseReports

General | Fields | Download | Children | Custom Actions | Finders

Custom Actions

Title
Reject
Approve

If you make cha

Custom Action Editor

General | Parameters | Payload Fields

ID: do_reject_ExpenseReports

Title: Reject

Description: reject

Path: /ExpenseReports/{ExpenseReports_Id}/action/reject

Request Media Type: application/vnd.oracle.adf.action+json

[ ] Batch Mode Enabled

Done

Layout Designer

General | Query | Columns | Advanced

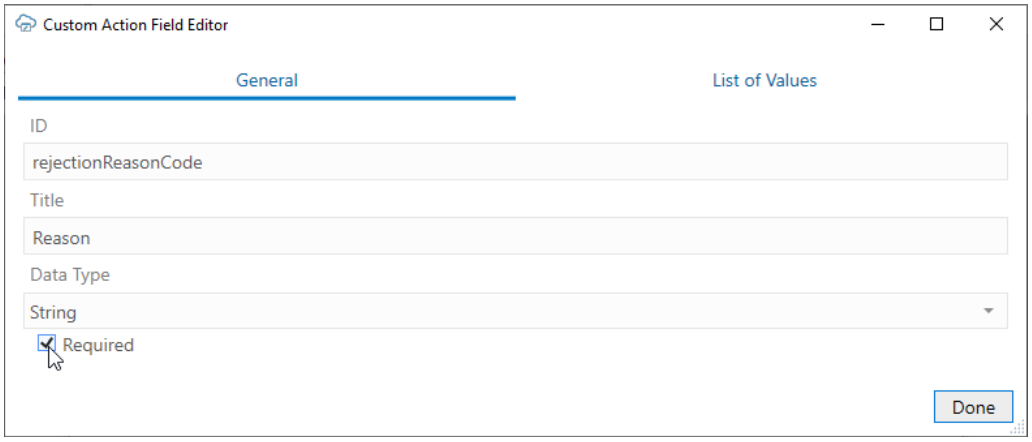
Table ID: TBL1188715962

Origin Cell: A1

Business Object: ExpenseReports

Business Object Catalog: HR-REST V2

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:



ID: rejectionReasonCode
Title: Reason
Data Type: String
[✓ Required]

You can also define a list of values for the payload field (see Configure a List of Values for a Custom Action Field).

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` 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:

- The path entry contains a path parameter for the row/item ID, for example, /`ExpenseReports/{ExpenseReports_Id}/action/reject"`
- 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, the add-in can process custom actions and other types of changes in batches.

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, clear the check box to disable batch processing. If the check box is cleared, 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.

8

Use Lists of Values in an Excel Workbook

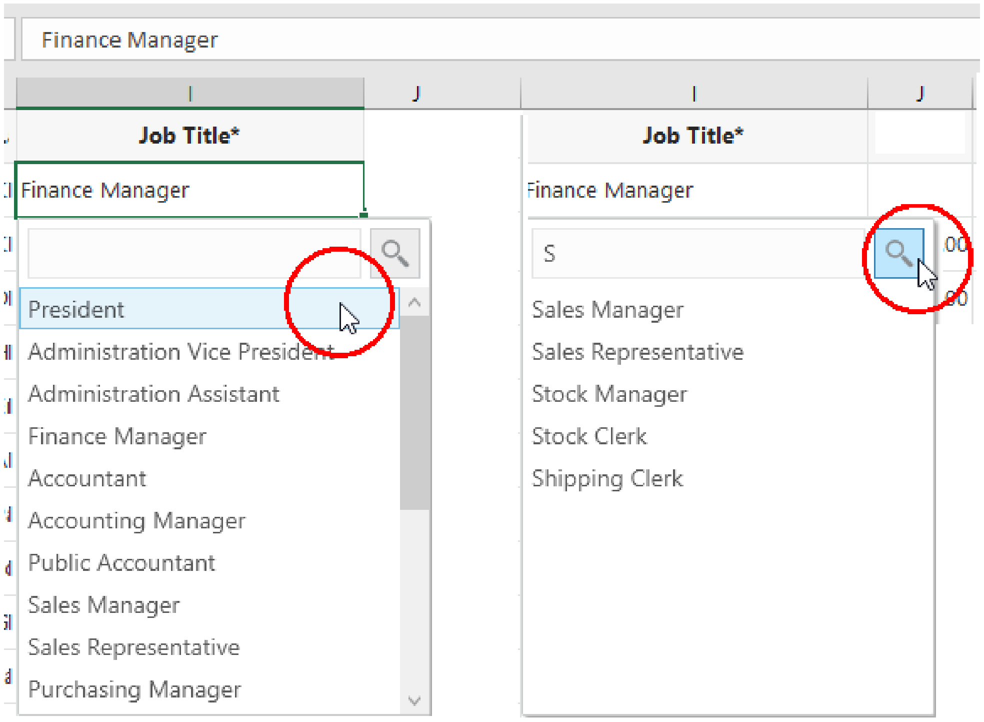
Oracle Visual Builder Add-in for Excel enables you to associate a **list of values** with a business object field.

A list of values is the set of valid values for a business object field and represents a relationship between the field in one business object and another referenced business object. 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, an Employee business object's Job attribute might contain these display and identity values:

Display value	Identity value
President	PRES
Finance Manager	FIN_MGR
Sales Manager	SAL_MGR

When a business object field has a properly configured list of values, you can expect the following behavior in Table and Form-over-Table layouts in your Excel workbook:

- On download, the identity values are replaced by the display values
- On upload, the display values are replaced by the identity values
- When a user selects a cell that is not read-only, a search-and-select window shows a list of display values for the user to select from. Alternatively, the user can enter some text and click the Search icon to filter and show values based on the user input, for example, to show values that start with the user-specified text. See Configure Search for a List of Values.
  This image illustrates both these scenarios. For the latter scenario, the user entered `S` in the search box to filter values from the list that begin with `S` (`Sales Manager`, `Sales Representative`, and so on). Users can also enter the full value in the search box, assuming it's a valid value such as `Sales Manager`.



Finance Manager

Job Title*
Finance Manager

- President
- Administration Vice President
- Administration Assistant
- Finance Manager
- Accountant
- Accounting Manager
- Public Accountant
- Sales Manager
- Sales Representative
- Purchasing Manager

Job Title*
Finance Manager

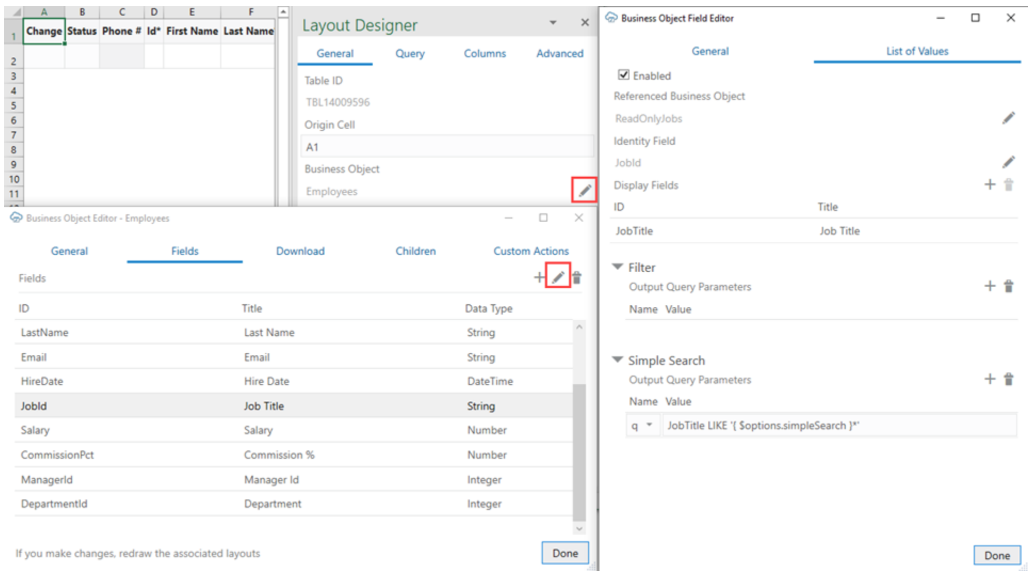
S

- Sales Manager
- Sales Representative
- Stock Manager
- Stock Clerk
- Shipping Clerk

Configure a List of Values for a Business Object Field

You can use the Business Object Field Editor to add or modify the list of values configuration for a field.

To access this editor, click the Edit icon next to the Business Object field in the Layout Designer. Once you open the Business Object Editor, click the **Fields** tab and select the business object's field. Then, click the Edit icon in the Business Object Editor to open the Business Object Field Editor, as shown in the following image where the Business Object Field Editor is open for the Employee business object's Job Title field.



Change	Status	Phone #	Id*	First Name	Last Name

Table ID: TBL14009596
Origin Cell: A1
Business Object: Employees

Fields

ID	Title	Data Type
LastName	Last Name	String
Email	Email	String
HireDate	Hire Date	DateTime
JobId	Job Title	String
Salary	Salary	Number
CommissionPct	Commission %	Number
ManagerId	Manager Id	Integer
DepartmentId	Department	Integer

If you make changes, redraw the associated layouts

☑ Enabled
Referenced Business Object: ReadOnlyJobs
Identity Field: JobId
Display Fields

ID	Title
JobTitle	Job Title

Filter
Output Query Parameters

Name	Value

Simple Search
Output Query Parameters

Name	Value
q	JobTitle LIKE '{ $options.simpleSearch }*'

The List of Values tab in the Business Object Field Editor shows the following properties:

- **Enabled**: Select to enable list of value functionality for this field; deselect to disable
- **Referenced Business Object**: This business object provides the display values for the corresponding identity values
- **Identity Field**: The field in the referenced business object used to look up the display values for the identity values in the current field
- **Display Fields**: These fields come from the referenced business object and are shown instead of the identity values where this field is used in a layout

To configure a list of values for a business object field that does not have one yet:

1. Select the **Enabled** check box in the Business Object Field Editor's **List of Values** tab.
2. Select the appropriate business object using the Edit icon next to the **Referenced Business Object** field.

   Tip:

   Pick a business object from the same catalog used to create the layout. 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).

3. Choose the appropriate identity field from the referenced business object.
4. Choose the desired display field from the referenced business object.
   You can choose multiple display fields for one list of values. In that case, values of all display fields are concatenated and the concatenated value will be shown in a single Excel cell.
5. Configure Filter and Simple Search as desired (see subsequent sections for details).
6. Click **Done**.

The add-in caches the data of list of values in the workbook. After you modify the configuration of any list of values, click **Redraw Layout**. Then, click **Clear List of Values Cache** from the **Advanced** menu in the Misc ribbon tab.

Note:

For a list of values with multiple display fields, if you enter a value directly in the cell that is not yet cached, the value will be considered invalid initially. Use the search-and-select window instead to select the value. The default search is specific to the first display field. You can modify the search expression using the Business Object Field Editor.

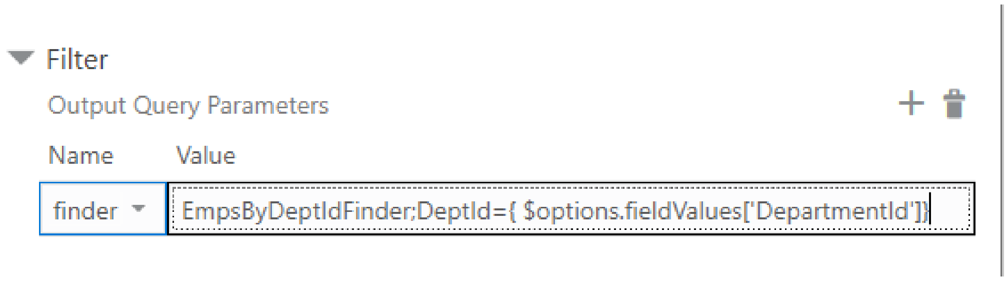
Configure Filters for a List of Values

You can allow users to filter a field's list of values by defining filters using name-value query parameters.

The query parameter's value is a string that could include the expression `{ $options.fieldValues['`*`FieldId`*`'] }`, where *FieldId* is the ID property of another field in the current business object. The add-in 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.

For example, when you work with a Table layout for departments and want users to be able to filter the current department's employees, you can specify a configuration to filter the Employees list with a Row Finder based on the current row's DepartmentId, if the REST service supports finders.

1. In the Business Object Field Editor's List of Values tab, click the Add Query Parameter icon in the Filter section.
2. In the **Name** list, select **finder** as the query parameter.
3. In the **Value** text box, enter an expression similar to:
   ```
   EmpsByDeptIdFinder;DeptId={ $options.fieldValues['DepartmentI
   d'] }
   ```
   where `EmpsByDeptIdFinder` is the finder's name, `DeptId` is the finder's bind variable, and `{ $options.fieldValues['DepartmentId'] }` is the expression for the current row's DepartmentId field value.



Name	Value
finder	EmpsByDeptIdFinder;DeptId={ $options.fieldValues['DepartmentId']}

If the list of values' default order is unclear, you can add the **orderBy** parameter if the service supports it.

Configure Search for a List of Values

You can allow your users to enter a string to search for a value in a list of values by defining search parameters.

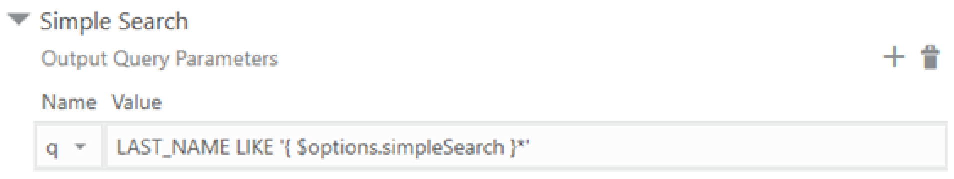
The query parameter value is a string expression that may include the expression `{ $options.simpleSearch }`. The add-in replaces this expression with the input that the user enters in the search box within the search-and-search window. The add-in does not validate the values, so if the user enters invalid information, the add-in will make invalid requests that return errors.

1. In the Business Object Field Editor's List of Values tab, click the Add Query Parameter icon in the Simple Search section.

2. Add values for the Name and Value properties based on what the service supports. Many REST services support a specific query parameter such as `q` for search, as in the following example:
   `GET…/employees?q=LAST_NAME LIKE 'Jones*'`

   Select the appropriate query parameter in the Name list; in our example, this is **q**. In the Value text box, provide a search expression that is appropriate for the service and include the `{ $options.simpleSearch }` text placeholder where appropriate. In our example, the Value is:

   ```
   LAST_NAME LIKE '{ $options.simpleSearch }*'
   ```



Name	Value
q	LAST_NAME LIKE '{ $options.simpleSearch }*'

Now, if the user enters `Jo` in the search box and clicks the Search icon, the resulting request to the referenced business object will be:

```
GET.../employees?q=LAST_NAME LIKE 'Jo*'.
```

Simple Search parameters are not applied when the add-in retrieves the initial set of values to show in the search-and-select window next to the cell in the worksheet. They are applied only when the user clicks the Search icon (after the initial set of values are already retrieved).

When both Filter and Simple Search parameters are configured, Filter parameters are first applied, then Simple Search parameters. If multiple values are specified for one parameter, the last value takes effect.

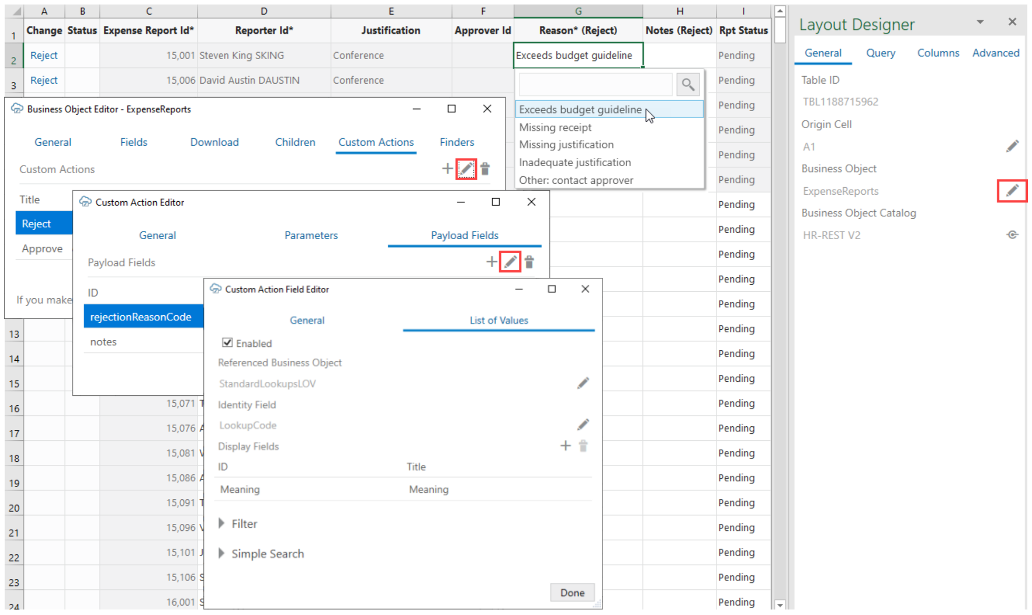
Note:

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*. Remember that the add-in uses REST framework version 6 when making requests to a business object REST API service.

Configure a List of Values for a Custom Action Field

Define a list of values for a custom action's payload field, just as you'd configure a list of values for a business object field.

Use the Business Object Editor to access a business object's custom actions and associated payload fields. You can then edit a payload field to define a list of values:



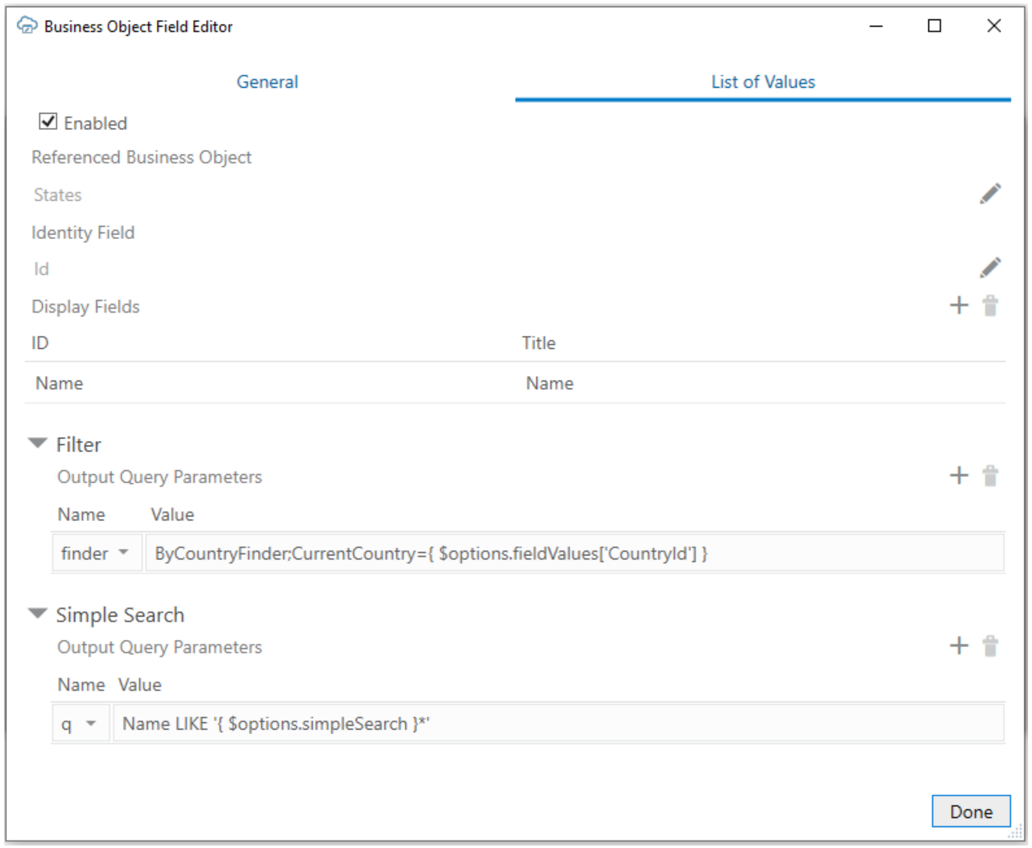
Once you define a list of values, the choice list will appear wherever that payload field appears, as shown in the preceding image where the referenced business object's list of values appear for the **Reason (Reject)** column.

Configure a Cascading List of Values in a Layout

You can configure a **cascading list of values** if the REST service for your Excel workbook supports it.

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 use the Filter section in the Business Object Field Editor to configure a list of values for Countries, States, and Cities. To filter the list of values by the current country:



Enabled: ✓

Referenced Business Object: States

Identity Field: Id

Display Fields

ID	Title
Name	Name

Filter

Output Query Parameters

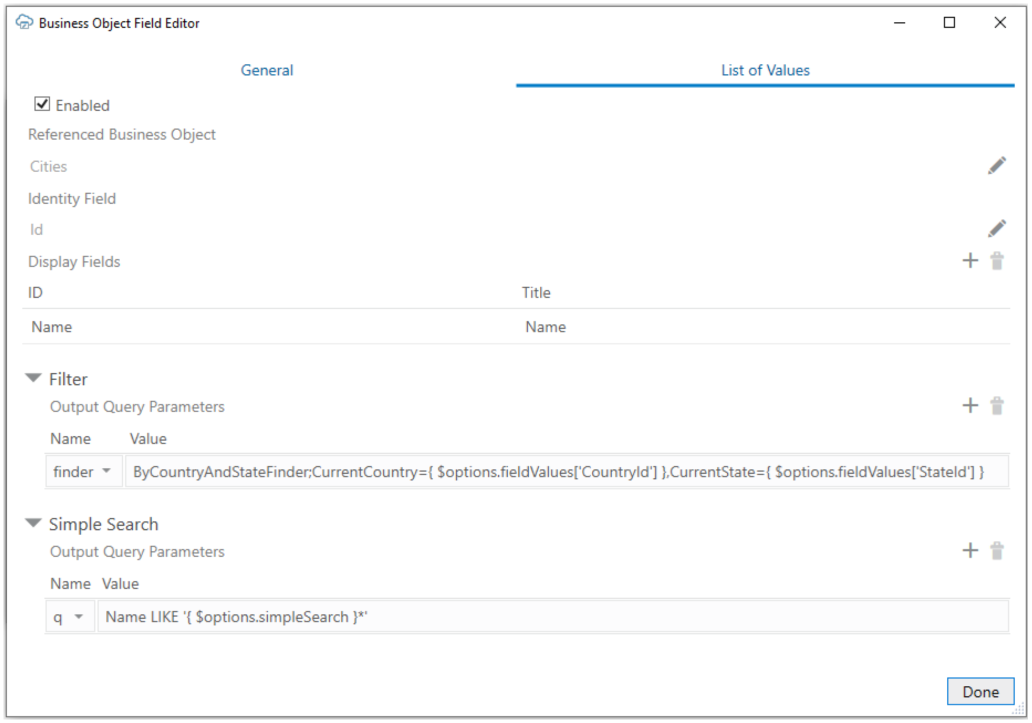
Name	Value
finder	ByCountryFinder;CurrentCountry={ $options.fieldValues['CountryId'] }

Simple Search

Output Query Parameters

Name	Value
q	Name LIKE '{ $options.simpleSearch }*'

To filter the list of values by Country and State:



Enabled: ✓

Referenced Business Object: Cities

Identity Field: Id

Display Fields

ID	Title
Name	Name

Filter

Output Query Parameters

Name	Value
finder	ByCountryAndStateFinder;CurrentCountry={ $options.fieldValues['CountryId'] },CurrentState={ $options.fieldValues['StateId'] }

Simple Search

Output Query Parameters

Name	Value
q	Name LIKE '{ $options.simpleSearch }*'

Note the following name-value parameters in the Filter sections:

Table 8-1 Example Name-Value Parameters in Filter

Field	Name	Value
StateId	**finder**	`ByCountryFinder;CurrentCountry={ $options.fieldValues['CountryId'] }`
CityId	**finder**	`ByCountryAndStateFinder;CurrentCountry={ $options.fieldValues['CountryId'] },CurrenState={ $options.fieldValues['StateId'] }`

If the service has complete metatdata for a cascading list of values, the Filter section 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 use the Filter to manually configure a list of values. See Configure a List of Values for a Business Object Field.

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 cached data is not what the user expects.

Notes and Limitations for List of Values

- 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.
- 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.

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.

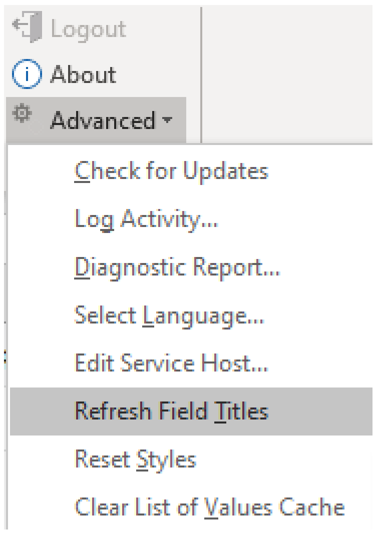
The add-in automatically formats cells in a workbook by creating a set of named Excel styles when the workbook is first initialized. The styles created by the add-in are consistent with Oracle Alta UI standards and Oracle accessibility guidelines. The format portion of the styles is sensitive to the 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.



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.

> **Note:**
>
> You can't reset styles in published workbooks.

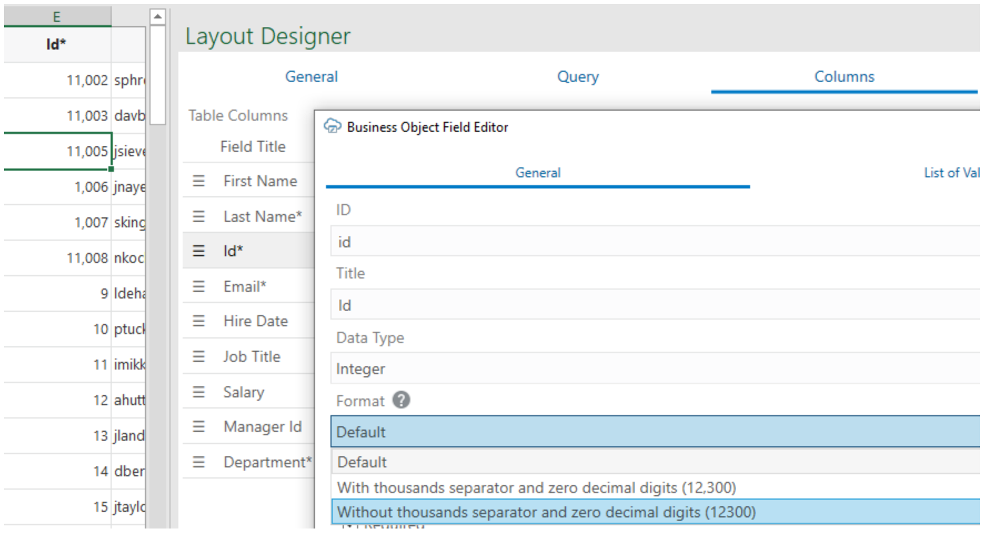
Choose Field Formats

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 (π = 3.14) while a French user will see a decimal comma (π = 3,14).

   Business Object Field Editor

   ID: id

   Title: Id

   Data Type: Integer

   Format: Default

   - Default
   - With thousands separator and zero decimal digits (12,300)
   - Without thousands separator and zero decimal digits (12300)

3. Click **Done**.
4. Click **Redraw Layout** or **Download Data** to process the change.

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 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.
5. Success and failure is noted in the Status column. Errors are cached and displayed in the status viewer when the user clicks on a failed row.
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.

The request payload for Create and Update operations includes a value (possibly empty) for every column in the table, except those for read-only fields and custom action payload fields (see Upload Changes for Custom Actions). The add-in does not track which columns have been altered, it always sends values for the entire row.

The add-in may send up to 4 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.

   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 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.

   The child table is not involved in this step.
4. The results of the form upload are reflected in the status viewer immediately.
5. If the form upload fails, the add-in stops and does not attempt an upload on the child table.
6. If the form upload succeeds, the add-in proceeds with the child table as follows:
   a. Checks the child table for pending changes, creates, deletes, custom actions, etc.
   b. If changes are found, the child table upload proceeds in the same manner as a Table layout upload with one important difference: the child business object's paths are used for each request.
   The add-in ensures that the parameters in the child business object paths are replaced with the correct values during the table upload operation.

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.

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 processes rows marked for custom actions:

1. Validates values from custom action payload fields (if any) for 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). Values from other columns are ignored.
2. Makes a POST request to the custom action path. The payload consists (only) of all the values for all the columns that correspond to the particular custom action's parameters; no other columns' values are included.
   For each marked row, the add-in performs the following steps:
   - 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. 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.
   - 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)
   - Makes the request
     – Sends the POST (POST is the only HTTP method supported for invoking custom actions)

     – See the note below about batch requests.
   - 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.

For information on custom actions, see Custom Actions.

Upload Changes Using Batch Requests

For Oracle business object REST API services, the add-in uses a batch API to send 25 rows per request for pending changes. During an Upload operation, rows marked for Update, Create, Delete, and custom actions are included in the batch requests.

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 request per row. For more information, see Making Batch Requests.

For Oracle REST Data Services and other service types, there is one request per row.

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.)

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. User A downloads information from a business object into a table in their integrated workbook.
2. User B downloads the same information into a table in their integrated workbook, edits it, and uploads changes.
3. User 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 User A's changes from overwriting those changes made by User 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.

For information about the entity tag (ETag) mechanism, see Data Consistency Tasks in *Accessing Business Objects Using REST APIs*.

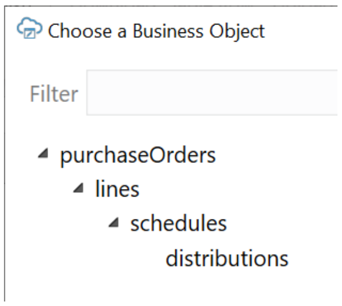
If your workbook uses other types of REST services, the last writer wins. So, for the scenario just outlined, User A’s changes in Step 3 will overwrite the changes of User B in Step 2.

11

Use Multiple Layouts for Multi-level Business Objects

When business objects in your REST service have a parent-child relationship of 3 or more levels, you can create a set of dependent layouts and then perform operations such as download data and upload your changes to all your layouts at once.

Consider this example hierarchy of business objects, where **purchaseOrders** is the parent, **lines** is the child, **schedules** is the grandchild, and **distributions** is the great-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 and each of these schedules may include one or more distributions for managing accounting or project details.

To create a set of dependent layouts, create a Form-over-Table layout for the first two levels (purchaseOrders and lines) and separate Table layouts for each subordinate level (schedules and distributions). Then link each dependent layout to its direct parent 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 layout, the grandchild layout, and so on.

If you allow your users to create new rows in a layout (create is enabled in the Table's capabilities), you must add one or more columns from the parent layout to uniquely identify the parent of the new child item. To create a new distribution record, for example, the user must be able to specify the correct line and schedule for the new distribution.

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.

Note:

Settings such as those for queries, macros, and pagination only apply to the primary layout and are not enforced on third-level and deeper dependent layouts.

Downloading Data

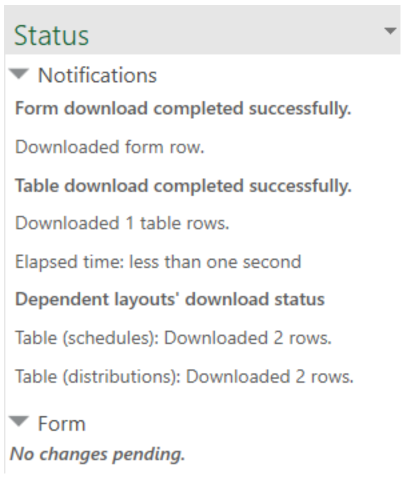
On download, the add-in first checks all layouts in the hierarchy for any pending changes. If there are changes pending, the user is prompted to confirm the download operation. If the user chooses to proceed, all pending changes are lost.

During download, the add-in first retrieves the values for the primary layout from the REST service. After the primary layout is populated, the add-in makes the next worksheet in the hierarchy active and retrieves all the appropriate items.

All items for all rows from the parent layout are downloaded at each level. (Any search specifications, if configured, apply only to the primary Form-over-Table layout.) 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 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:



Form download completed successfully.

Downloaded form row.

Table download completed successfully.

Downloaded 1 table rows.

Elapsed time: less than one second

Dependent layouts' download status

Table (schedules): Downloaded 2 rows.

Table (distributions): Downloaded 2 rows.

No changes pending.

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 updates first from the form and then from the table of the layout. The add-in then moves to the worksheet with the first dependent table layout active and uploads changes before proceeding to the next layout.

Pending changes may include creation of new items, updating of existing items, and invocation of actions on items. For rows pending Update, values in the parent column cells are ignored.

For new items at the third-level or deeper, values in the parent columns must match a row in the parent layout. For example, to create a new distribution, you must specify an existing line and schedule in the parent columns with which to associate the new item. Empty cells in the dependent layout or 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.

Create a Set of Dependent Layouts

Create a set of dependent layouts for a hierarchy of business objects to three or more levels and link the layouts together.

The primary layout is always a Form-over-Table, one that's based on the parent business object (in the form) and a child business object (in the table). In our example, the form displays the details of a purchase order and the table lists the associated lines for this purchase order.

Additional Table layouts, created on separate worksheets, are based on descendant business objects, relative to the parent business object. For example, the first Table layout displays the grandchild business object (schedules, in our example), the next one displays the great-grandchild business object (distributions), and so on.

Note:

Sibling relationships (two or more layouts with the same parent layout) are not supported.

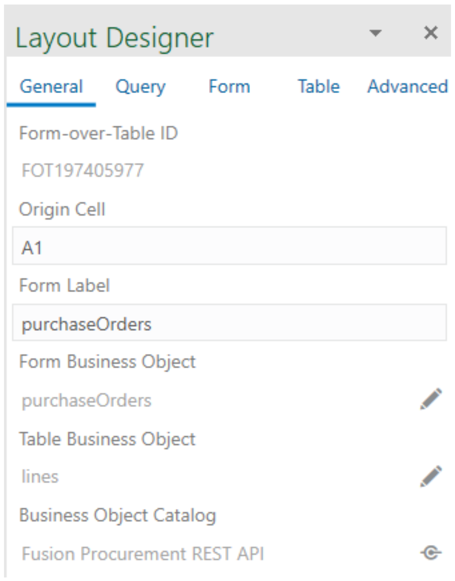
For each Table layout you create, you can include a column from the immediate parent layout to help your users keep track of the parent record for each row in the table. For example, if ScheduleNumber is displayed as a column in the schedules layout, you can add it to the distributions layout to see which distributions are associated with this schedule.

To enable the creation of items in third-level and deeper layouts in a dependent layout hierarchy, 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. See Support for Creating Items.

Let's use the example in this section to create a hierarchy of dependent layouts that mirrors your business object hierarchy.

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**.
   b. When prompted, provide the service description document.
   c. Choose the parent business object. For example, select purchaseOrders to create a layout for purchaseOrders over lines (the first two levels in our hierarchy) and click **OK**.
   d. Select **Form-over-Table Layout** as the new layout and click **OK**.

   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.

   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 levels in your business object 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**.
   c. Choose the business object that's third in the hierarchy, for example, schedules, and click **OK**.
   d. Select **Table Layout** as the new layout and click **OK**.

e. Repeat steps **a** to **d** to create Table layouts for all the remaining levels in your business object hierarchy. Continuing our example, create a Table layout for distributions.

A Table layout is created for each level in the business object 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.

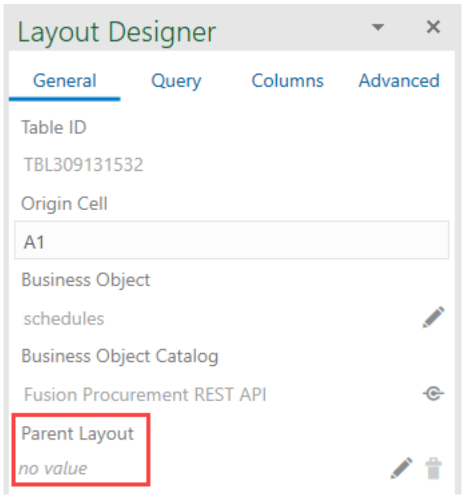
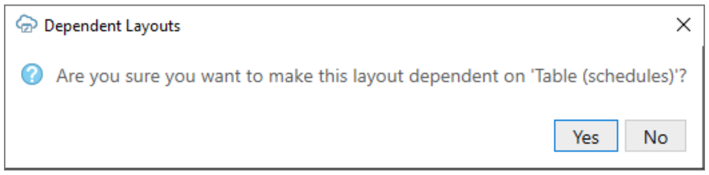


Table ID
TBL309131532
Origin Cell
A1
Business Object
schedules
Business Object Catalog
Fusion Procurement REST API
Parent Layout
no value

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:



Are you sure you want to make this layout dependent on 'Table (schedules)'?

   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.

Once a layout has its **Parent Layout** defined, you'll notice a **Child Layout** field in its parent's Layout Designer. Also, search specifications (defined in the Query tab) are no longer available for any dependent layout that is not the primary layout. Here's the Layout Designer for the schedules Table layout, whose parent is purchaseOrders-lines and child is distributions.

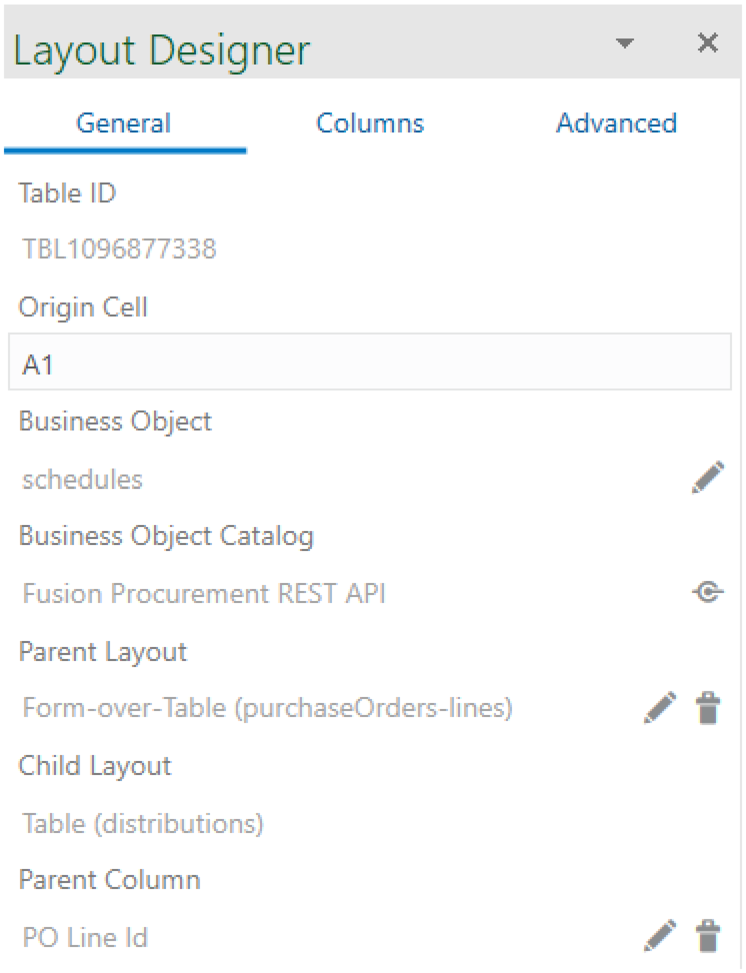


Table ID

TBL1096877338

Origin Cell

A1

Business Object

schedules

Business Object Catalog

Fusion Procurement REST API

Parent Layout

Form-over-Table (purchaseOrders-lines)

Child Layout

Table (distributions)

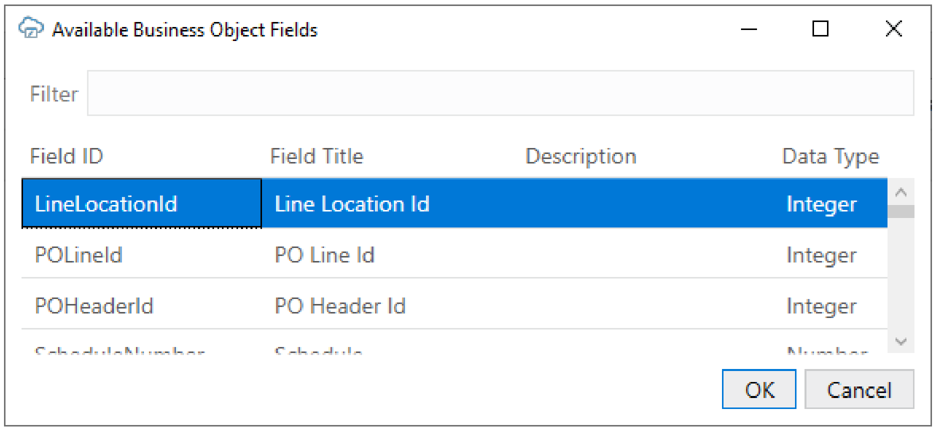
Parent Column

PO Line Id

4. If you plan to enable the creation of new items in third-level or deeper layouts (for example, schedules and distributions), choose one or more columns from the parent layout to display in the layout.

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

   a. From the worksheet (distributions in our example), click the Add Parent Column icon (+) in the Layout Designer's General tab.
   b. From the Available Business Object Fields window, select a field from the parent layout and click **OK**. The field must be exposed as a column in the parent table. If you don't see the field you want to use, you'll need to add it as a column in the parent table.


Available Business Object Fields

Filter

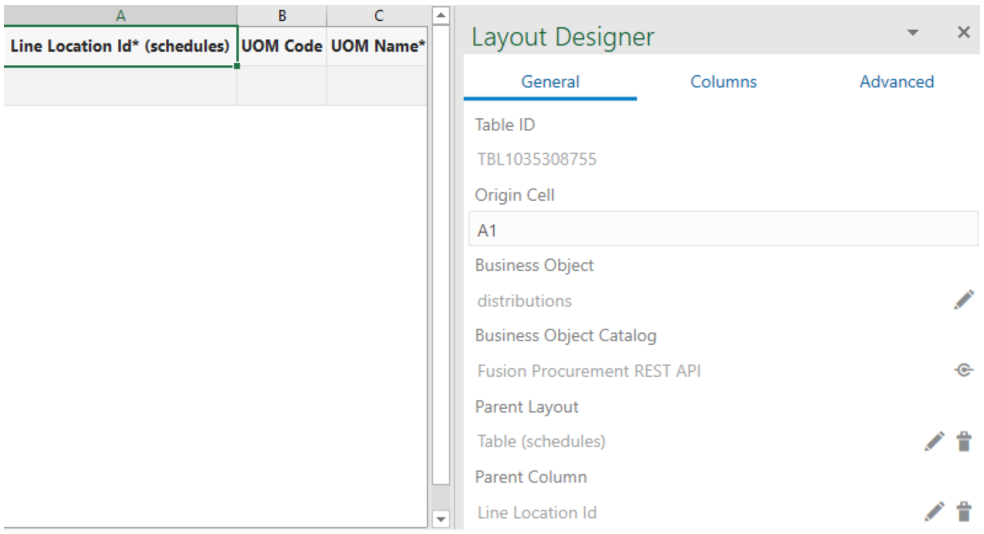
Field ID	Field Title	Description	Data Type
LineLocationId	Line Location Id		Integer
POLineId	PO Line Id		Integer
POHeaderId	PO Header Id		Integer
ScheduleNumber	Schedule		Number

c. Repeat steps **a** and **b** to add additional parent columns to the child layout as required.

d. Repeat steps **a** to **c** to add parent columns to other Table layouts in the hierarchy.

5. Click **Redraw Layout** for each of the modified worksheets in your hierarchy.

   A column for each parent field added is created in the child layout, for example, the **Schedule (schedules)** column as shown here:


A	B	C
Line Location Id* (schedules)	UOM Code	UOM Name*

Layout Designer

Table ID
TBL1035308755
Origin Cell
A1
Business Object
distributions
Business Object Catalog
Fusion Procurement REST API
Parent Layout
Table (schedules)
Parent Column
Line Location Id

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*.

Support for Creating Items

To enable the creation of items in third-level and deeper layouts in a dependent layout hierarchy, 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.

The parent columns you choose must uniquely identify the parent row and must be exposed in the parent table. For example, to create a new distribution, a user must be able to specify an existing line and schedule with which to associate the new item. Therefore, you would want to add columns such LineNumber and ScheduleNumber to both the distributions and schedules layouts.

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

Add a parent column to a dependent layout using the Add Parent Column icon (+) in the Layout Designer's General tab.

The field must be exposed as a column in the parent table. If you don't see the field you want to use in the Available Business Object Fields window, you'll need to add it as a column in the parent table.

Repeat the step to add additional columns from the parent layout and then redraw the layout when you're done. The parent columns appear after the Status column in the layout in the order you added them. The table header for the parent column uses the format "<*field title*> (<*parent business object title*>)" such as "Line Number (lines)".

To clear the set of parent columns, use the Clear Parent Columns icon (🗑) next to the Parent Columns property.

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.

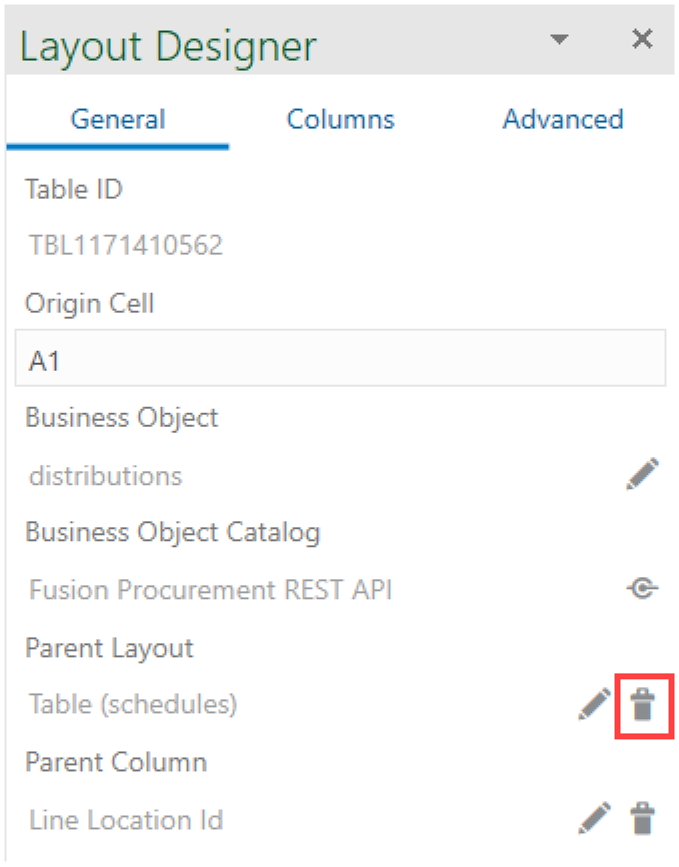


Table ID

TBL1171410562

Origin Cell

A1

Business Object

distributions

Business Object Catalog

Fusion Procurement REST API

Parent Layout

Table (schedules)

Parent Column

Line Location Id

3. When prompted, click **Yes** to remove the dependency.
4. Click **Delete Layout**, then confirm your selection.
5. On the other layouts in the workbook, click **Redraw Layout**.

12

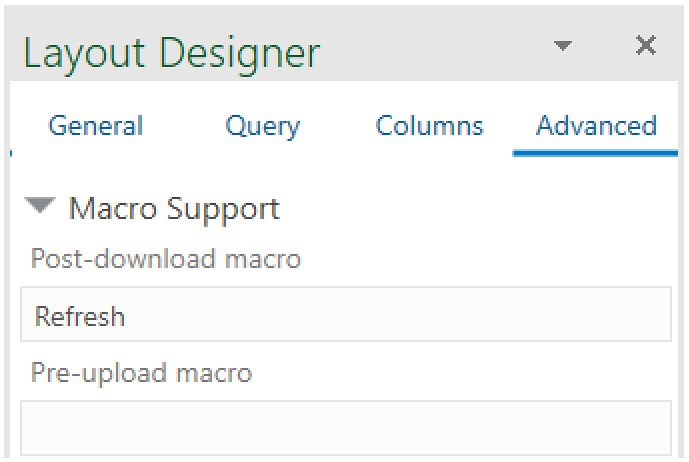
Use Macros in an Integrated Excel Workbook

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 user base 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.



Post-download macro: Refresh

Pre-upload macro:

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.

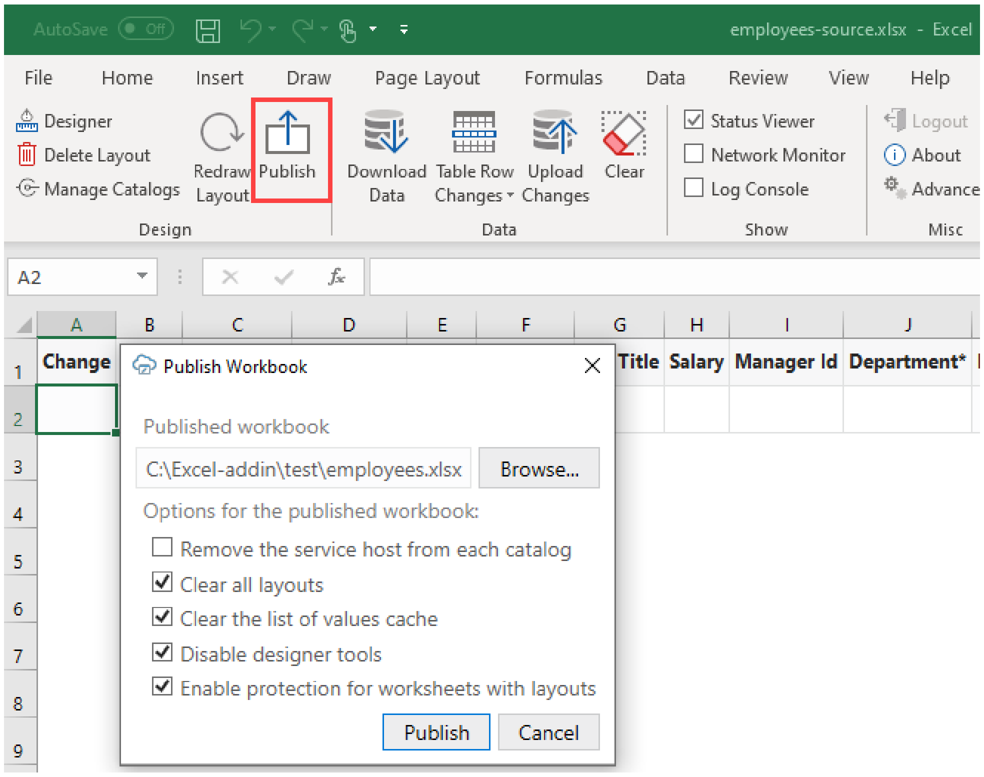
13
Publish an Integrated Excel Workbook

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**.



Publish Workbook

Published workbook

C:\Excel-addin\test\employees.xlsx [Browse...]

Options for the published workbook:

- [ ] Remove the service host from each catalog
- [✓] Clear all layouts
- [✓] Clear the list of values cache
- [✓] Disable designer tools
- [✓] Enable protection for worksheets with layouts

[Publish] [Cancel]

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.
   - 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**

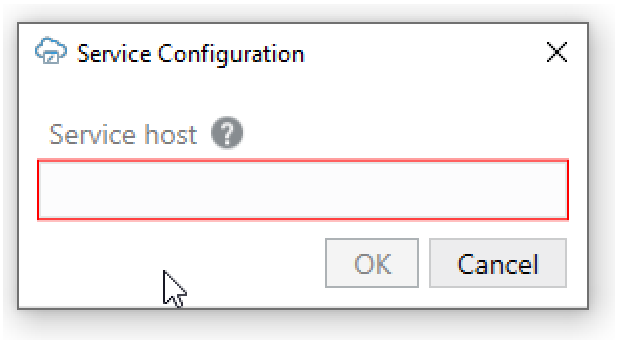
worksheet with layouts. This option enables Excel's worksheet protection for each worksheet with a layout.

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 the source workbook:

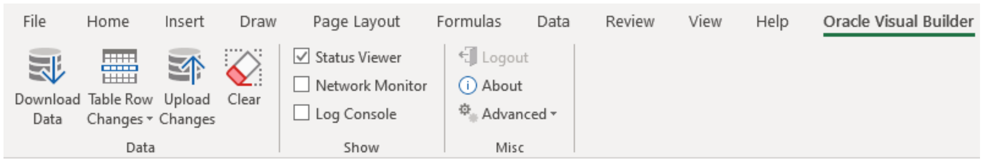
- If the service host value was removed for a workbook, users who open that workbook and perform an action that requires access to the REST service are prompted to enter the service host value, as shown here:



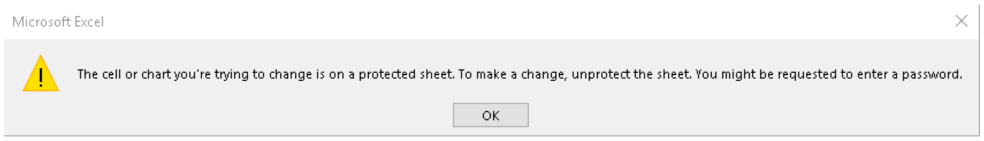
Actions that require access to the service include the Download Data and Upload Changes commands.

> **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:



The cell or chart you're trying to change is on a protected sheet. To make a change, unprotect the sheet. You might be requested to enter a password.

Worksheet protection also prevents the user from performing Excel actions that might disrupt the workbook's integration with the service.

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

The add-in provides special support for:

- Oracle business object REST API services
  - Provides search editor capabilities
  - Supports the ability to upload in batches
  - Supports row finders
- Oracle REST Data Services (ORDS)
  - Provides search editor capabilities

The add-in can also be used with other service types as long as the service behaves as the add-in expects.

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 `attachment` attribute data type, the add-in ignores it and does not create a column in the data table for this attribute type.

For more information, see the specifications for OpenAPI and JSON Schema:

- OpenApi: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md#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 specified for a schema, the Required properties default to `false`.

You can always edit the **Required for Update** and **Required for Create** values in the Business Object Field Editor (see Edit Service Descriptions and Business Objects).

Object-typed Fields and Sub-fields

The add-in 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 Operations

Table and Form-over-Table capabilities are enabled as follows:

- Existing row updates are enabled if the item path has either a PUT or PATCH operation. For PATCH 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 (not supported for Form-over-Table)

> **Note:**
>
> You can choose to disable a layout's capability even if the business object supports the operation, by deselecting it in the Layout Designer's Advanced tab.

Language Support

For every request that the add-in makes to the REST service, the add-in 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.

Oracle REST Data Services

As with other service types, you must 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 other ORDS endpoints, you need to find or create an OpenAPI service description. See Create a Service Description in Oracle Visual Builder. For information about AutoREST, see Automatic Enabling of Schema Objects for REST Access (AutoREST) in *Oracle REST Data Services Installation, Configuration, and Development Guide*.

Only basic authentication is supported when working with ORDS services.

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

- For some row-level errors, the ORDS server does not provide a specific reason for the error.
- 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 Installation, Configuration, and Development 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.
- When using ORDS lower than version 19.2, some date-time fields are not recognized as a date-time data type due to faulty service metadata. Use the add-in’s Business Object Field Editor to correct the data type.

REST Service Support Limitations

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.

- Only REST API services that return application/json media types as response payloads are supported. Other media types (for example, XML and octet-stream responses) are not supported.
- Polymorphic business objects cannot be used to create a layout.
- 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 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.
- When providing a URL for an OpenAPI service description, `?metadataMode=minimal` is not supported.

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 Edit Service Descriptions and Business Objects.

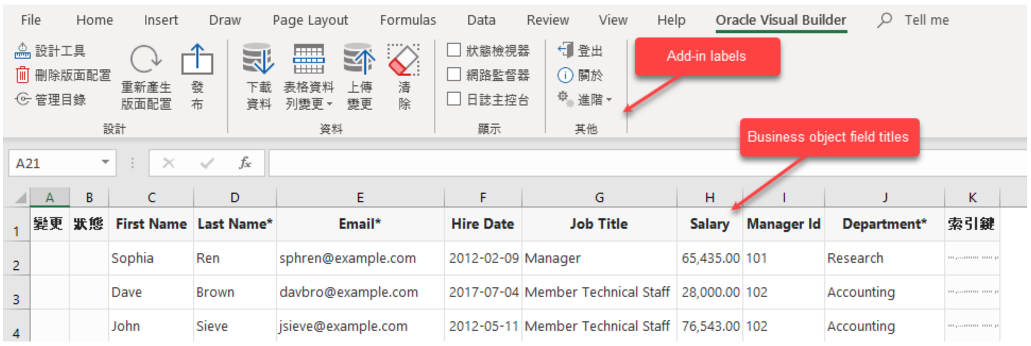
15
Internationalization

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.



	A	B	C	D	E	F	G	H	I	J	K
1	**變更**	**狀態**	**First Name**	**Last Name***	**Email***	**Hire Date**	**Job Title**	**Salary**	**Manager Id**	**Department***	**索引鍵**
2			Sophia	Ren	sphren@example.com	2012-02-09	Manager	65,435.00	101	Research	[illegible]
3			Dave	Brown	davbro@example.com	2017-07-04	Member Technical Staff	28,000.00	102	Accounting	[illegible]
4			John	Sieve	jsieve@example.com	2012-05-11	Member Technical Staff	76,543.00	102	Accounting	[illegible]

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

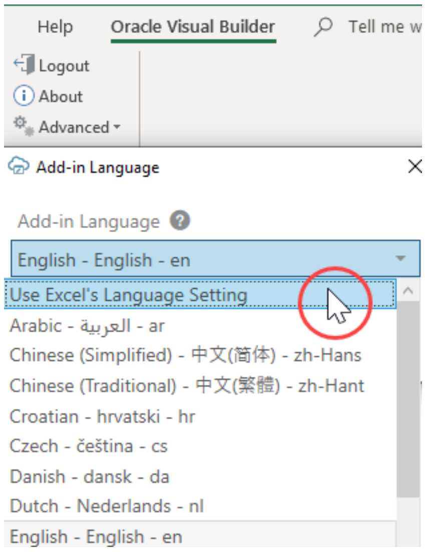
You can change the language that the Excel add-in uses. Do this if you want to evaluate your integrated workbook with different languages.

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. 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.

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

If your integrated workbook uses an Oracle business object REST API service, it is possible to refresh all field titles.

This functionality is handy when switching languages. Here's an example of when to reset field titles: 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.

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**.

Note:

The reset functionality relies on the `/describe` response from the Oracle business object REST API service, with the add-in sending the current language preference in the request. A given service may or may not have translations for the requested language. For any missing translations, contact the owner of the service.

16 Security Best Practices

When using Oracle Visual Builder Add-in for Excel, follow these 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.

Basic Authentication

The add-in supports basic authentication:

- When using REST service endpoints protected by basic authentication, the user is prompted for credentials when the add-in connects to the endpoint.
- When used with HTTP, basic authentication is not secure. Basic authentication should only be used with HTTPS, and preferably only in non-production environments.

JSON Web Token

In addition to basic authentication, the add-in also supports authentication for REST services exposed by Fusion applications that use the JSON Web Token (JWT) relay servlet. No configuration is required by you. The add-in automatically detects whether the Fusion application's service has the `/anticsrf` and `/tokenrelay` endpoints configured. The add-in then displays a pop-up browser window and navigates to the hosting web application's login page. When the user provides valid credentials, the pop-up automatically closes and access to the service can proceed using the token obtained during the login sequence.

Use of the JSON Web Token (JWT) relay servlet is only available for Fusion applications, as the path to the token relay service that the add-in uses is specific to Fusion applications.

Note:

In this release of the add-in, using **self-signed** certificates with the JWT relay servlet will not work. A valid certificate issued from a well-known root certificate authority should work fine with the JWT relay servlet.

TLS Security

When the add-in connects to a REST endpoint using HTTPS, the add-in enables the system default behavior for 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 4.7 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)

17

Troubleshoot Excel Workbooks

Oracle Visual Builder Add-in for Excel can generate a detailed log file and diagnostic report to help you identify and resolve issues in the Excel workbooks that you integrate using the add-in.

Before you dig deeper to isolate issues, follow these steps to fix general issues with your installation:

- 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.

Network Monitor

Use the Network Monitor window to inspect the content of REST service calls between your Excel workbook and the REST service that it connects to 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 JSON payloads that the Excel workbook and the service exchange.

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. You can select and copy information from the window. You may need this detail when troubleshooting problems with the service owner. The window shows up to 100 request-response events. Older events are discarded as new ones are added.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Network Monitor**.
   The Network Monitor window displays.
3. Repeat the steps that lead to the issue.
4. Review the details of each request and response.
5. Optionally, right-click an entry for a REST service call in the upper table, and choose the **Save request-response details as ...** option to save the information to a file.

   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.

Logging

When reporting an issue about the Excel add-in, generate a detailed log file that captures the steps that lead to the problem you want to report.

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.

The log file that you generate captures information about steps during an Excel session.

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. Please provide a diagnostic report when reporting a problem with the 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.
4. Review the content of the diagnostic report to remove any sensitive information that you do not want to share before you use it to report an issue.

18
Migration

You can migrate an Excel workbook created or modified with a previous version of the add-in to use the current version of the add-in.

Migrate 2.*x* Workbooks

Workbooks created or modified with version 2.*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 Best Practices for an Upgrade).

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.

Migrate 1.*x* Workbooks

Note:

1.*x* workbooks are deprecated. The add-in's latest version continues to support the migration of 1.*x* workbooks to the 2.*x* format. Future versions will no longer let you migrate 1.*x* workbooks.

You can migrate a workbook created or modified with version 1.*x* of the add-in to use the current version. However, once you migrate, you can no longer use the workbook with version 1.*x* of the add-in.

During migration of a 1.*x* workbook, if you fail to log in to the REST service, migration fails. Close the workbook without saving changes and try again later.

Before you install the current release of the add-in:

- Upload any pending changes from the Excel workbooks that use version 1.*x* of the add-in
- Click **Clear** for all layouts in the Excel workbook before you migrate
- Before you migrate an Excel workbook, make sure you are prepared to log in to the server configured in the Excel workbook
- If you need to change the service host (the server name that the REST service uses), be sure to do so with version 1.*x* of the add-in. You cannot change the service host during migration using version 2.*x*.

After you install version 2.*x* of the add-in, clear all layouts before using them with the latest version.

19 Third Party License

Oracle Visual Builder Add-in for Excel includes the following third-party software.

NewtonSoft.Json, Version 12.0.3

Microsoft.OpenApi, Version 1.2.0