

Oracle® Cloud

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



F35242-01
July 2020



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

F35242-01

Copyright © 2018, 2020, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

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

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

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

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

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

Contents

1	What's New in Release 2.3.0	
	New and Changed Features	1-1
	Known Limitations	1-1
2	Get Started with the Oracle Visual Builder Add-in for Excel	
	What is the Oracle Visual Builder Add-in for Excel?	2-1
	Key Concepts, Components, and Terms	2-1
	How to Begin with the Oracle Visual Builder Add-in for Excel	2-1
3	Install the Oracle Visual Builder Add-in for Excel	
	Check for Updates	3-3
	Best Practices for an Upgrade	3-4
	Install from the Command Line	3-4
4	Create Layouts in an Excel Workbook	
	Create a Table Layout in an Excel Workbook	4-1
	Work with Service Path Parameters in a Table Layout	4-3
	Create a Form-over-Table Layout in an Excel Workbook	4-4
	Manage Layout Capabilities	4-7
5	Edit Service Descriptions and Business Objects	
	Create a Service Description in Oracle Visual Builder	5-4
	Add a Business Object to an Existing Catalog	5-4
	Override a Business Object's Base Path	5-4
	Configure Pagination for a Business Object	5-5
6	Configure Search Options for Download	
	Use Search to Limit Downloaded Data	6-2

	Use REST Finders to Limit Downloaded Data	6-3
	Use Search Parameters to Limit Downloaded Data	6-3
	Download Behavior in Layouts that Use Search and REST Finders	6-5
7	Use Lists of Values in an Excel Workbook	
	Configure a List of Values for a Business Object Field	7-2
	Configure Filters for a List of Values	7-4
	Configure Search for a List of Values	7-4
	Configure a Cascading List of Values in a Layout	7-5
	Clear Cache for a List of Values	7-7
8	Custom Actions	
	Use Custom Actions	8-1
	Use Custom Actions in a Form Row of a Form-over-Table Layout	8-4
	Edit Custom Actions	8-5
	Service Description for Custom Actions	8-5
9	Appearance of an Integrated Excel Workbook	
	Reset Workbook Styles	9-1
	Choose Field Formats	9-2
10	Upload Changes	
	Upload Changes from a Table Layout	10-1
	Upload Changes from a Form-Over-Table Layout	10-1
	Upload Changes Using Batch Requests	10-2
	Upload Changes for Custom Actions	10-3
11	Use Multiple Layouts for Multi-level Business Objects	
	Delete a Dependent Layout	11-5
12	Use Macros in an Integrated Excel Workbook	

13 Publish an Integrated Excel Workbook

14 View and Edit Data in an Excel Workbook

Download Data to the Workbook	14-2
Edit Downloaded Data in the Workbook	14-3
Understanding Data Validation	14-4
Understanding Read-Only Behavior	14-5
Create New Rows to Upload to the REST Service	14-5
Delete Data from the REST Service	14-7
Upload Changes to the REST Service	14-7
View and Edit Data in a Form-over-Table Layout	14-8
Create a Parent Row in a Form-over-Table Layout	14-8
Perform Custom Actions in a Table or Form-over-Table Layout	14-9
Manage Data for Layouts in a Dependent Hierarchy	14-12
Data Consistency	14-13

15 REST Service Support

16 Internationalization

Change the Add-in's Language	16-1
Refresh All Field Titles	16-2

17 Security Best Practices

18 Troubleshoot Excel Workbooks

Network Monitor	18-1
Logging	18-2
Logging Console	18-2
Diagnostic Report	18-2

19 Migration

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

Audience

Developing Integrated Spreadsheets Using Oracle Visual Builder Add-in for Excel is intended for Oracle Visual Builder users who want to create and publish Excel workbooks that integrate with the enterprise applications that they use. This guide is also helpful to users who need to work with published Excel workbooks.

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*

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

What's New in Release 2.3.0

Here's an overview of new features and enhancements added to Oracle Visual Builder Add-in for Excel in Release 2.3.0. Take note also of some limitations when using the add-in.

New and Changed Features

- Perform coordinated download, upload, and clear operations by linking layouts across multiple "dependent" layouts for business objects with multi-level hierarchies. See [Use Multiple Layouts for Multi-level Business Objects](#).
- Option to hide the Design tools, so users who don't need access to those tools can disable them during installation. See [Install the Oracle Visual Builder Add-in for Excel](#).
- Control capabilities to perform various standard and custom actions in a layout by enabling or disabling capabilities supported in a layout. See [Manage Layout Capabilities](#).
- Configure a business object to use a base path different from the one shared by business objects in the "Catalog". See [Override a Business Object's Base Path](#).
- Add filters to a list of values, so your users can filter a field's list of values. See [Configure Filters for a List of Values](#).
- Perform custom actions on a form row in a Form-over-Table layout. See [Use Custom Actions in a Form Row of a Form-over-Table Layout](#).

Known Limitations

Be aware of these limitations when using the Oracle Visual Builder Add-in for Excel.

- Review [View and Edit Data in an Excel Workbook](#) for guidelines on viewing and editing data in an integrated Excel workbook.
- Review [REST Service Support](#) to understand 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

Get Started with the Oracle Visual Builder Add-in for Excel

The Oracle Visual Builder Add-in for Excel is an add-in for Microsoft Excel that lets you interact with data from REST web services. It 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.

What is the Oracle Visual Builder Add-in for Excel?

The Oracle Visual Builder Add-in for Excel integrates Microsoft Excel spreadsheets with REST services to retrieve, analyze, and edit business data from the service. You can 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	An entity such as employees. 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.

How to Begin with the Oracle Visual Builder Add-in for Excel

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.

The following image shows a published worksheet that is integrated with an REST service that manages employees:

	A	B	C	D	E	F	G	H	I
1	Change	Status	First Name	Last Name*	Email*	Hire Date	Job Title	Salary	Manager Id
2		Update Succeeded	Sophia	Ren	sphren@example.com	2012-02-09	Member Technical Staff	65,435.00	101
3		Create Succeeded	Lisa	Roe	lroe@example.com	2006-04-23	Member Technical Staff	6,700.00	120
4	Update		Dave	Brown	davbro@example.com	2017-07-04	Member Technical Staff	28,000.00	102
5			John	Sieve	jsieve@example.com	2012-05-11	Vice President	76,543.00	102
6			Julia	Nayer	jnayer@example.com	2005-07-16	Member Technical Staff	3,200.00	120

The user has updated one row and created a new row with employee data. These changes have been successfully uploaded to the service, as indicated by messages in the Status column. The user has also updated data in another row that has yet to be uploaded, as indicated by the Update message in the Change column.

The image also features the Table layout, which is one of two types of layouts that the add-in can create in an Excel worksheet. The second type is the Form-over-Table layout, which you can configure for services where a parent-child relationship exists between a parent business object and child business objects in the service. You can create one type of layout per worksheet in your Excel workbook. That is, each worksheet in the Excel workbook can include a layout.

Subsequent sections in this guide describe how you install, configure, publish, and use an integrated Excel worksheet. You can integrate Excel workbooks with services that provide a service description in the OpenAPI format. See [REST Service Support](#).

For Excel specifications and limits, see [Microsoft documentation](#).

3

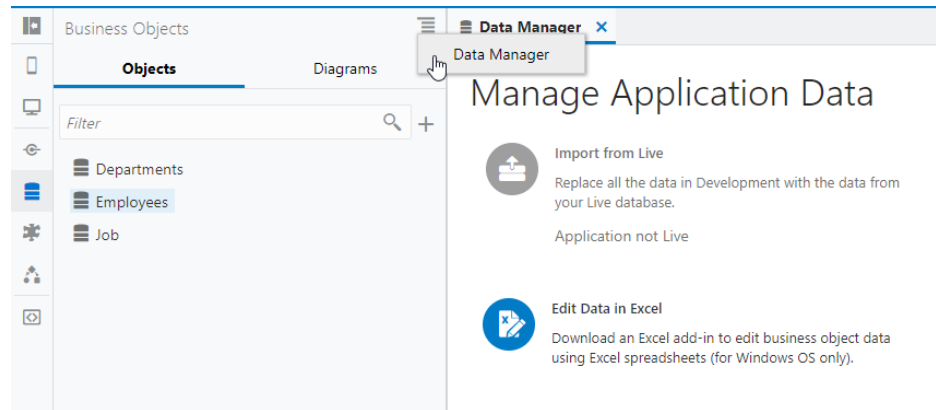
Install the Oracle Visual Builder Add-in for Excel

Install the Oracle Visual Builder Add-in for Excel by downloading the installer from your visual application's Data Manager.

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 \(Doc ID 2474783.1\)](#)).

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. Before you install the add-in, download the `vbafe-installer.exe` installation file from your Visual Builder instance.
 - 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**.

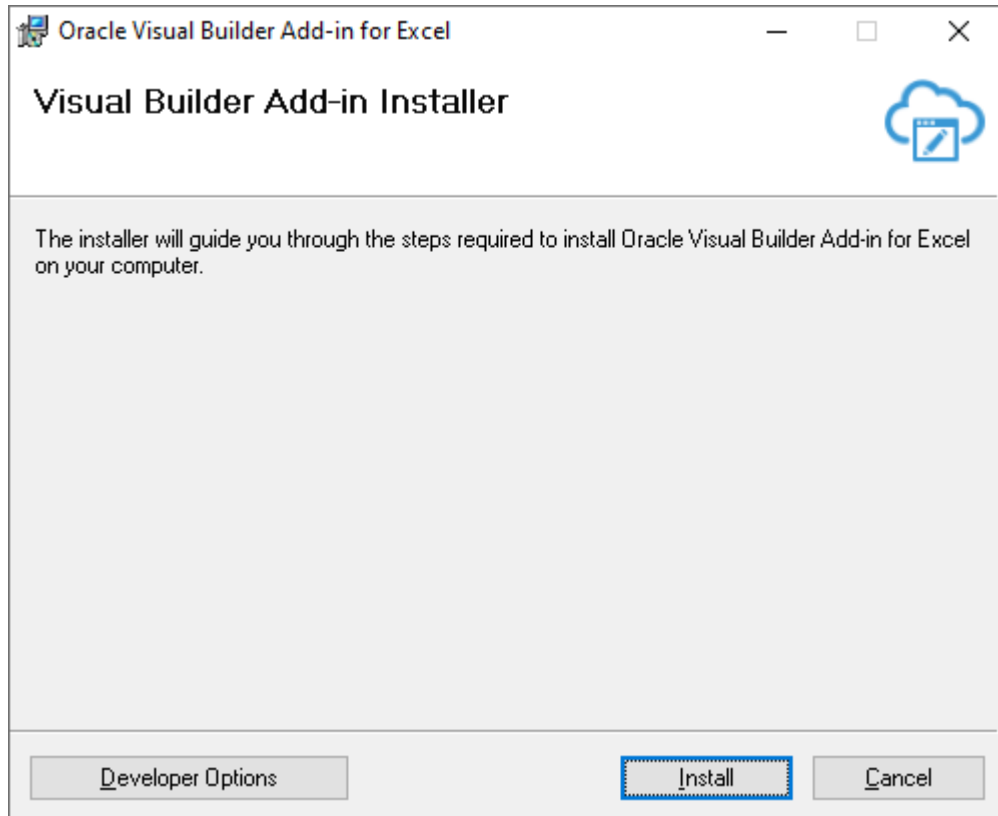


2. Quit Excel before you begin installation.
3. Double-click the `vbafe-installer.exe` file that you downloaded previously to launch the installation wizard.
4. When you install the Oracle Visual Builder Add-in for Excel, you get a full set of commands to design layouts, publish workbooks, and more. If you don't need access to the 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**.



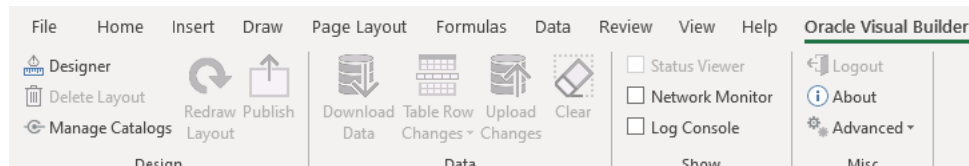
The add-in requires the Microsoft .NET Framework and Visual Studio Tools for Office Runtime to be installed on your computer. If this software is not present, both are installed.

 **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.

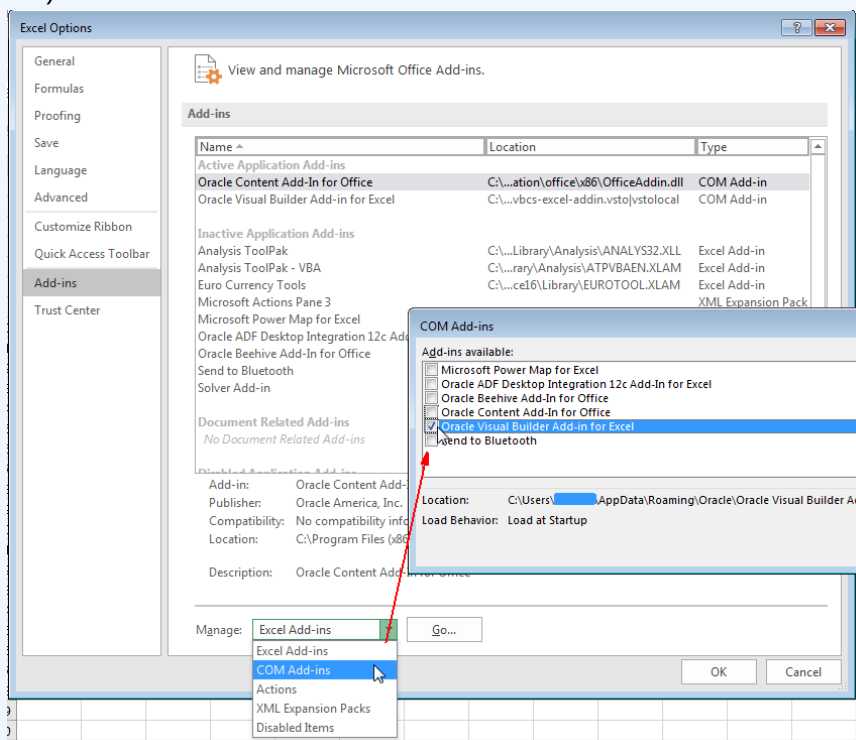
6. Once installation is complete, click **Close**.
If you run into issues, check the `%TEMP%\vbafe\vbafe-installer-log.txt` installation log.
7. Start Excel and open a new or existing workbook.

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.



Note:

The add-in is enabled by default after installation. You can disable and re-enable it using the **Oracle Visual Builder Add-in for Excel** check box in the COM Add-ins window (click Excel's **File > Options > Add-Ins**).

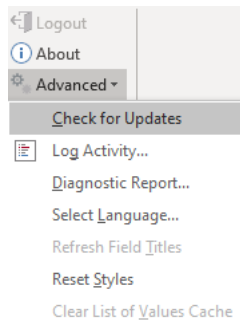


- To verify your installation, you can download and run the Visual Builder Add-in for Excel - Client Health Check Tool (see [How to use Visual Builder Add-in for Excel - Client Health Check Tool \(Doc ID 247792.1\)](#)).

Check for Updates

After you install the Oracle Visual Builder Add-in for Excel, it's a good idea to check whether a newer version of the add-in is available for you to install. You'll need an Oracle.com account to sign in and download a new version.

- 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

When you want to upgrade your existing installation of Oracle Visual Builder Add-in for Excel, follow these recommendations for a clean upgrade.



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 the 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
<code>/help</code>	Displays a list of supported switches with description.

Table 3-1 (Cont.) Oracle Visual Builder Add-in for Excel Installer Command-Line Switches

Switch	Description
<code>/quiet</code>	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.
<code>/designer 0 1</code>	Use this switch as follows: <ul style="list-style-type: none">• 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.
<code>/log path</code>	Runs the installer and directs the log output to the specified log file. The default log file location is <code>%TEMP%\vbcs\vbcs-installer-log.txt</code> .
<code>/roaming 0 1</code>	Use this switch as follows: <ul style="list-style-type: none">• 0 to install the add-in to the local application data folder (<code>%localappdata%\Oracle\Oracle Visual Builder Add-in for Excel</code>). Use <code>/roaming 0</code> 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 (<code>%appdata%\Oracle\Oracle Visual Builder Add-in for Excel</code>). This is the default installation location.

4

Create Layouts in an Excel Workbook

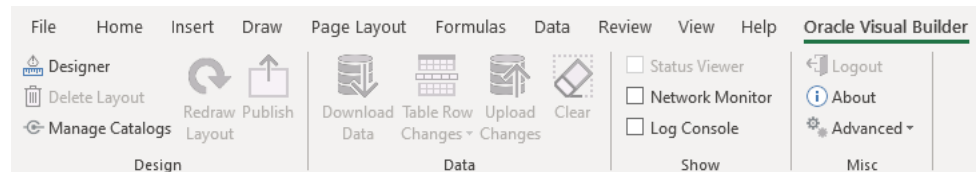
The Oracle Visual Builder Add-in for Excel lets you create different layouts to work with your data in an Excel worksheet. You can create one type of layout per worksheet in your Excel workbook. That is, each worksheet in the Excel workbook can include a layout.

Create a Table Layout in an Excel Workbook

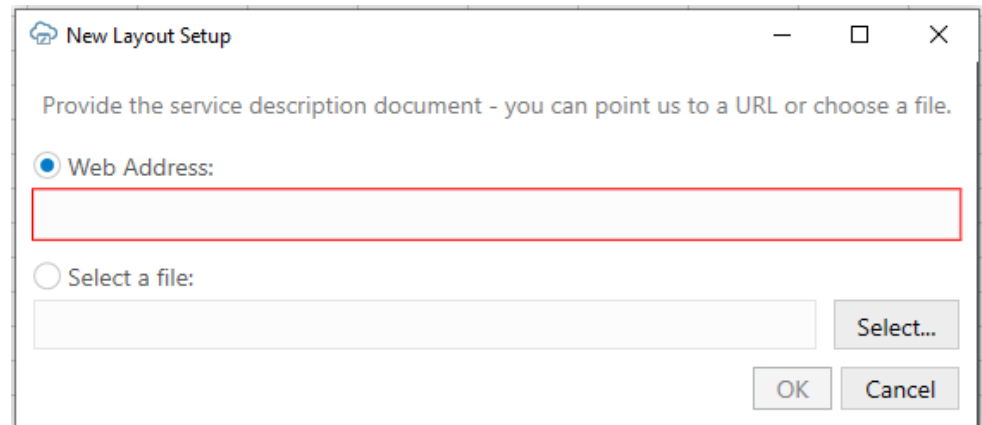
Create a Table layout in the Excel worksheet when you want to view and edit data from a REST service in a tabular format.

Run Excel and create a blank workbook using the standard Excel workbook file format type (.XLSX) or the macro-enabled workbook type (.XLSM). The Oracle Visual Builder Add-in for Excel does not support other Excel formats (.XLS, and so on).

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



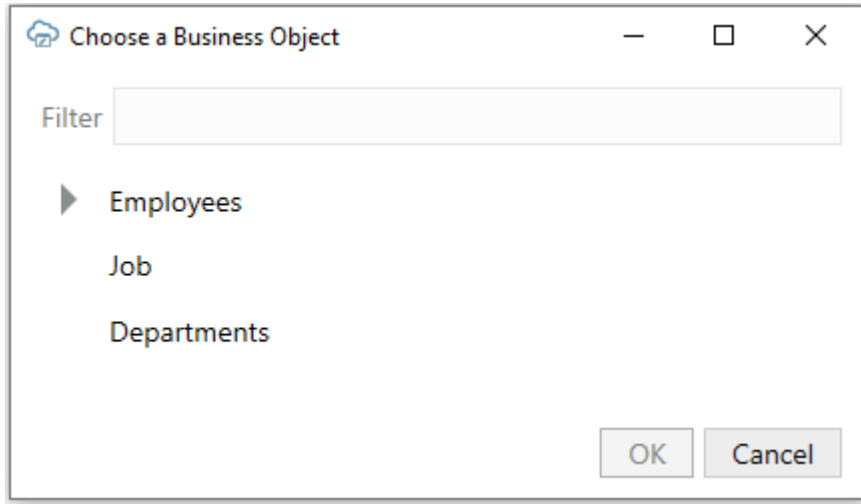
2. Click the cell where you want to locate the table.
3. In the Oracle Visual Builder tab, click **Designer**.
4. When prompted, provide the service description document. Use the **Web Address** option (the default) if you access the service description from a URL. Note that you can't provide a data 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.



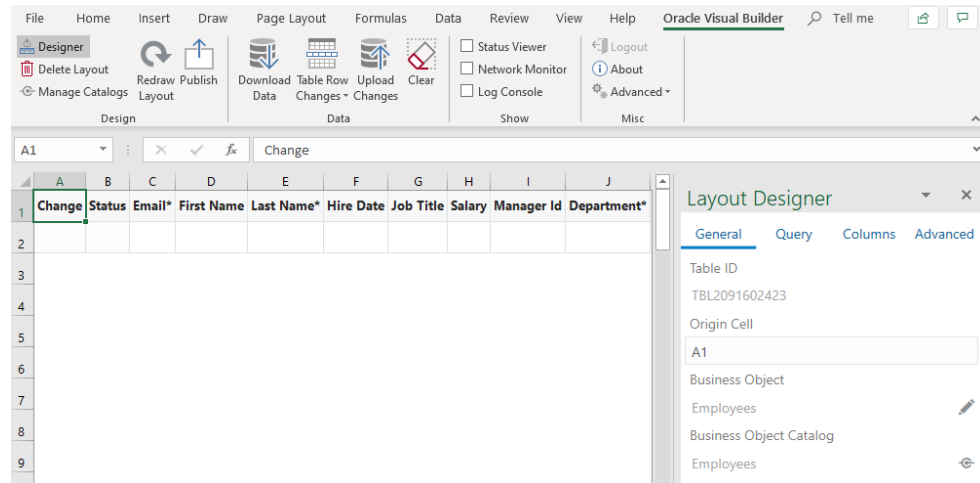
 **Tip:**

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.



5. Click the **Columns** tab to add or remove columns, or change the order in which columns appear. Review the list of columns to verify that the add-in only creates the desired columns. After making changes in the Layout Designer, click **Redraw Layout** to see your changes reflected in the worksheet.



Note:

When a table is created, most (but not all) fields are added as columns. Click the Manage Columns button (+) to add or remove columns as needed.

The workbook is now complete and ready to be published. We recommend you test the table by performing various operations, such as download, update, and upload before you publish the workbook to distribute it to users. Alternatively, you can configure the workbook further so that the add-in limits the data that it downloads, as described in [Configure Search Options for Download](#).

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, when your parameterized service path is `/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.

Service Path Parameter Editor

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

Parameters:

ExpenseReportsId

OK Cancel

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.

The following 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.

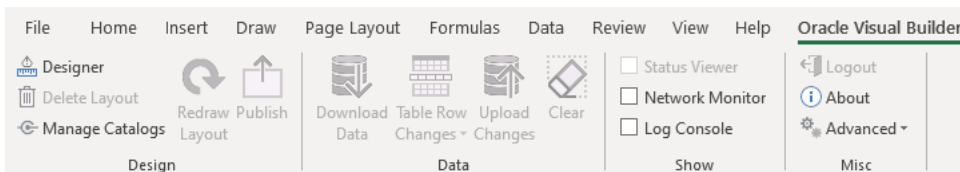
The add-in remembers the values provided at download time. These values are used again at upload time to construct the upload requests. If you upload without having done a previous download (for example, when exclusively creating new rows), you'll be prompted for the path parameter values at the beginning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

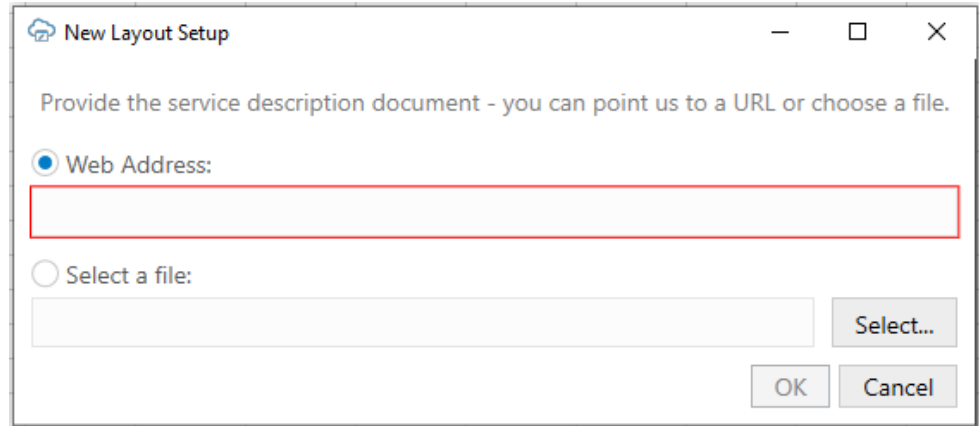
Run Excel and create a blank workbook using the standard Excel workbook file format type (.XLSX) or the macro-enabled workbook type (.XLSM). The Oracle Visual Builder Add-in for Excel does not support other Excel formats (.XLS, and so on).

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



2. Click on the cell where you want to locate the form and table.
3. In the Oracle Visual Builder tab, click **Designer**.
4. 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

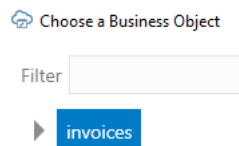
file on your computer.



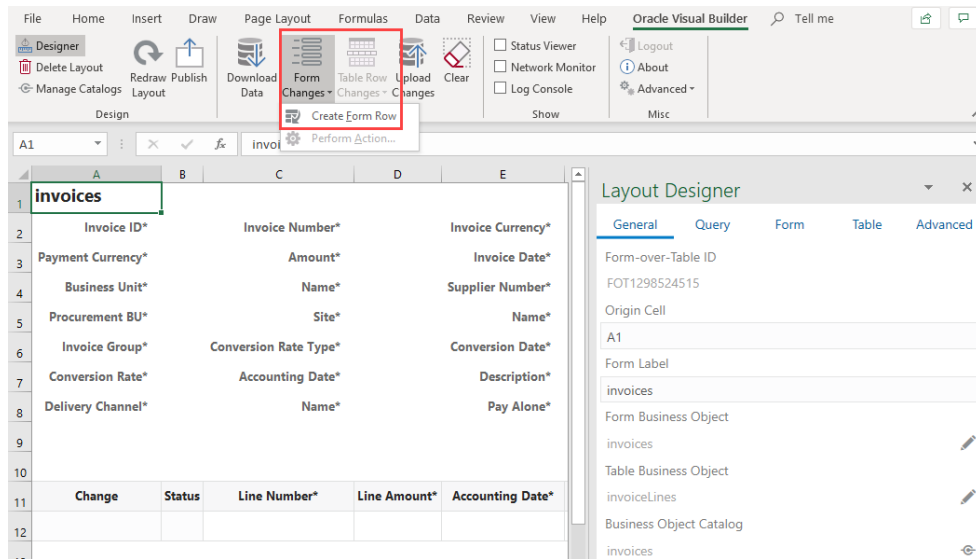
 **Tip:**

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:



5. Choose a business object, such as `invoices` shown in the previous image, where a parent-child relationship exists, choose **Form-over-Table Layout** in the New Layout Setup window that appears, and click **OK**. 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 illustrated in the following image:



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](#)).

6. 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. If, for example, you do not specify a value for the Search field or Finder property, as described in [Configure Search Options for Download](#), 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.

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 designer, right-click a field or column to see choices for changing the order.

Configuration is now complete and you can publish the workbook. We recommend you test the workbook before you publish it to distribute it to users. Alternatively, you can configure the workbook further so that the add-in limits the data that it downloads, as described in [Configure Search Options for Download](#).

Service Requirements for Form-Over-Table Layouts

A parent-child relationship at the service level requires:

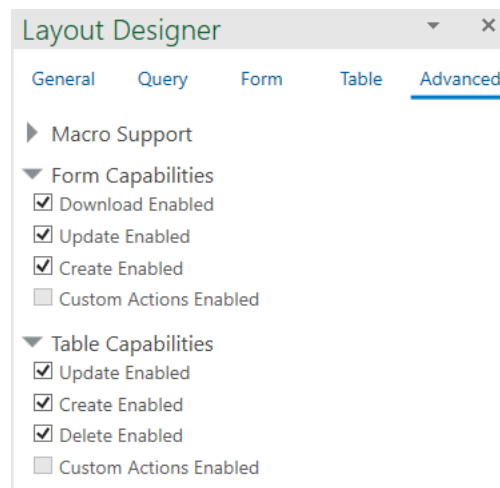
- A parent service path, for example, `fscmRestApi/resources/latest/invoices`
- A child service path with a parameter, for example, `fscmRestApi/resources/latest/invoices/{invoice-id}/child/invoicelines`

In your workbook, both business objects must be declared in the same service. 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.

Manage Layout Capabilities

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

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



3. Select or deselect options as required. If the business object doesn't support an action, it won't be available for selection (like **Custom Actions Enabled** in the example).
4. Click **Redraw Layout**.

5

Edit Service Descriptions and Business Objects

The Oracle Visual Builder Add-in for Excel provides editors where you can modify a workbook's service description, including its definition for 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; note that any changes that 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:

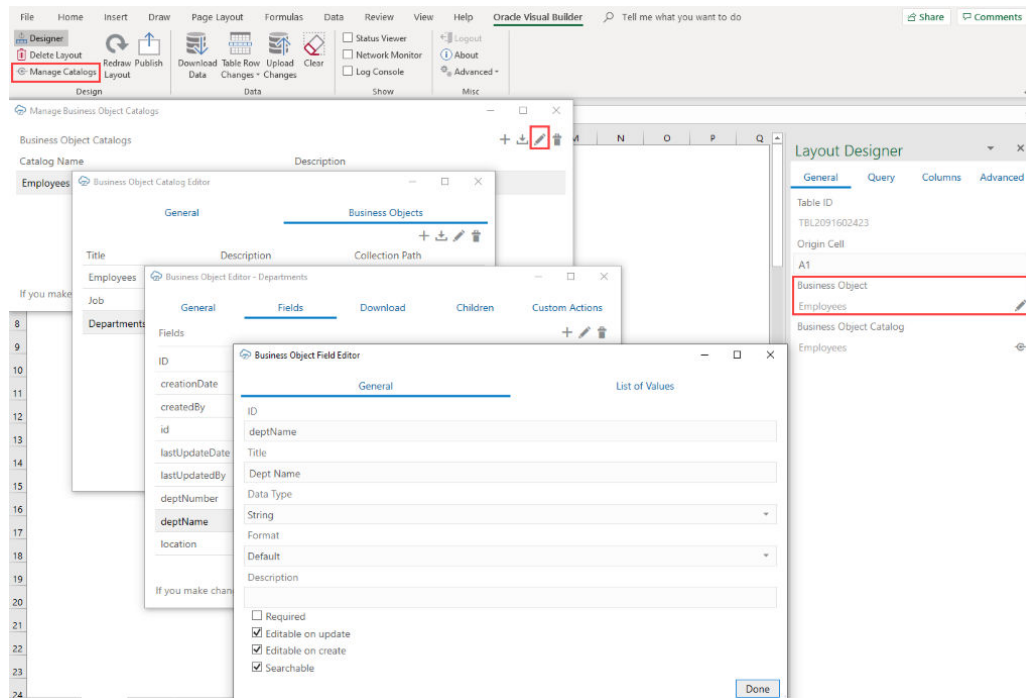
- Edit field titles
- Configure the [list of values](#) for a field
- Change when a field is editable
- Configure [pagination](#)
- Adjust the field data types (Advanced)

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 the overall user experience.

Note:

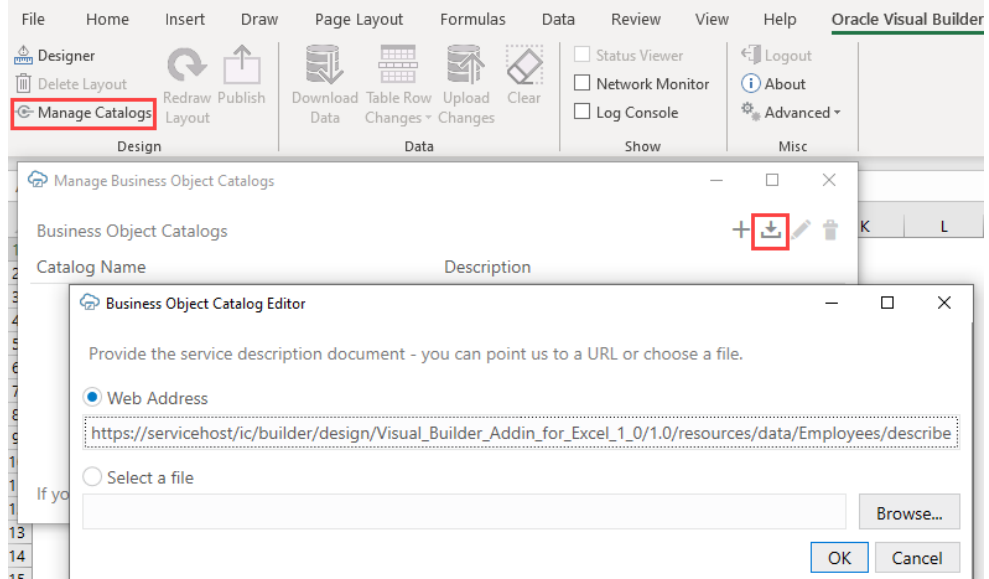
When you edit the service description in your workbook, you're only telling the add-in how the service operates, you are not telling the service what it should do. Service behavior cannot be changed from the workbook, so any changes you make to the service description in the workbook must be compatible with the service.

When you click **Manage Catalogs** in the main Oracle Visual Builder tab, the Catalog editors show progressively. That is, you first access the editor to edit the catalog. You then access subsequent editors for business objects and business object fields that the catalog provides access to. 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.

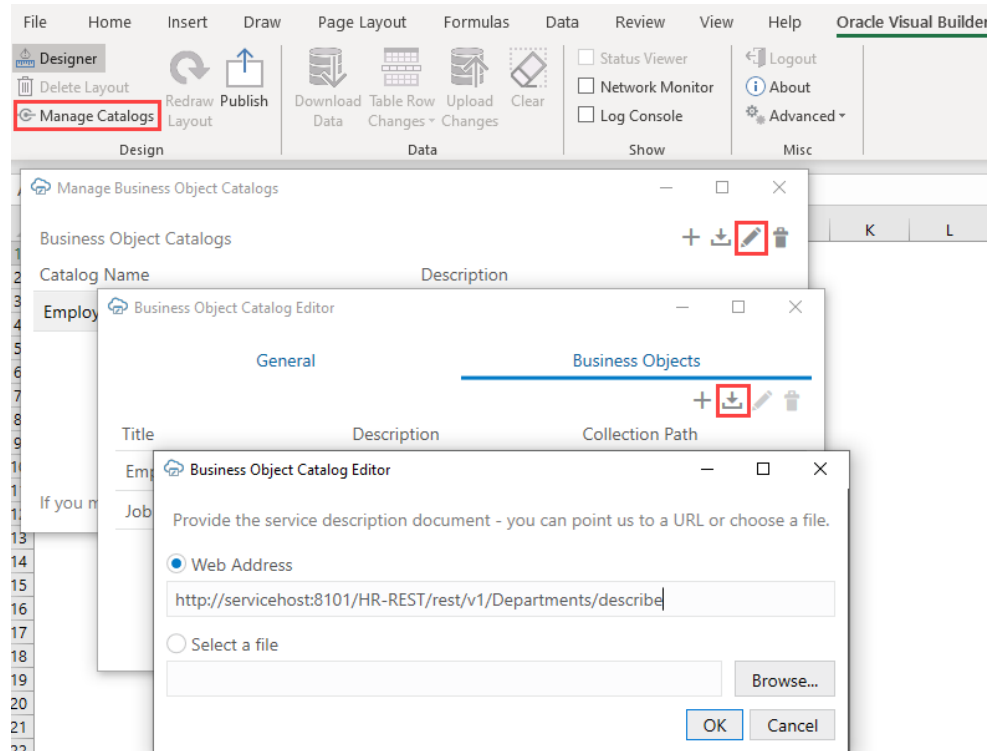


To import a service description document, you have two options:

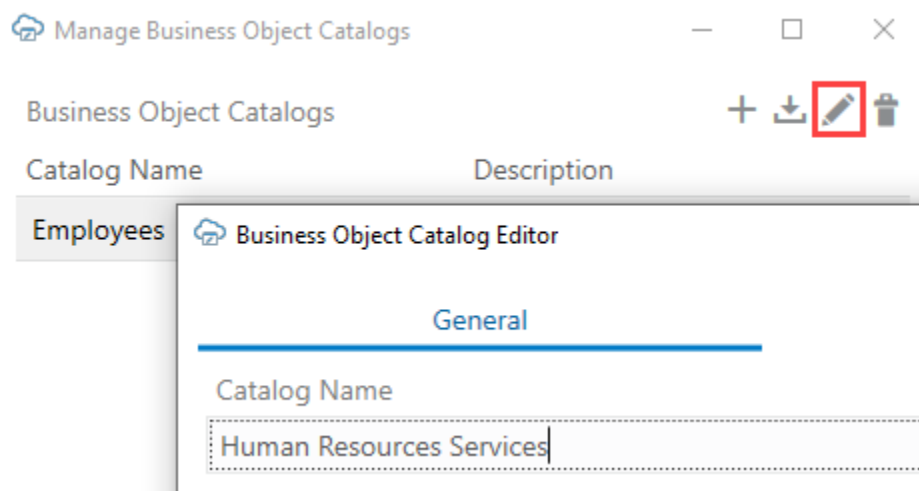
- If you want to create a new catalog, particularly if the OpenAPI document has multiple business objects, use Manage Catalogs.



- 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 Catalog Editor (see [Add a Business Object to an Existing Catalog](#)). This option is particularly useful when configuring a list of values.



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, the following image shows a catalog where 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 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 an OpenAPI-compliant service description 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**.

Business Object Editor - Countries

General Fields Download Children Custom Actions

Title
Countries

Description

Parent Business Object
Top-level business object (no parent)

Base Path Override ?
/123RestApi-context-root/resources/v1

Collection Path
/Countries Edit

Item Path
/Countries/{Countries_Id} Edit

If you make changes, redraw the associated layouts Done

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 slow. Attempting to download all 10,000 rows in one request might result in a timeout error. So, the solution is to 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
Salary	Manager Id	Department	Key
65,435.00	101	Research
28,000.00	102	Accounting
76,543.00	102	Accounting
3,200.00	120	Accounting
24,000.00	100	Accounting
17,000.00	100	Accounting
17,000.00	102	Accounting

Business Object Editor - Employees

Layout Designer

General Query Columns Advanced

Table ID
TBL726348314

Origin Cell
A1

Business Object
Employees

General Fields Download Children

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

The following list describes a number of the properties that the Download tab allows you configure values for:

- **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.
- **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.

6

Configure Search Options for Download

Configure the search options for a given layout. The configured search options are used when the user clicks **Download** in the Oracle Visual Builder tab.

The add-in provides search options that you configure for each layout. The 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 only configure Search and Search Parameters properties.

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 REST Finder property if the Oracle business object REST API service you connect to supports finders. If you select a value for the REST Finder property, the add-in uses it during download. A Finder that an Oracle business object REST API service supports may have parameters. If the Finder that you choose for the REST Finder property specifies parameters, the add-in prompts users to provide parameter 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 the values supported by the REST Finder property before it prompts the user to enter values for the Search property. Once the user enters the requested input values, the add-in downloads data based on the values that the user input.

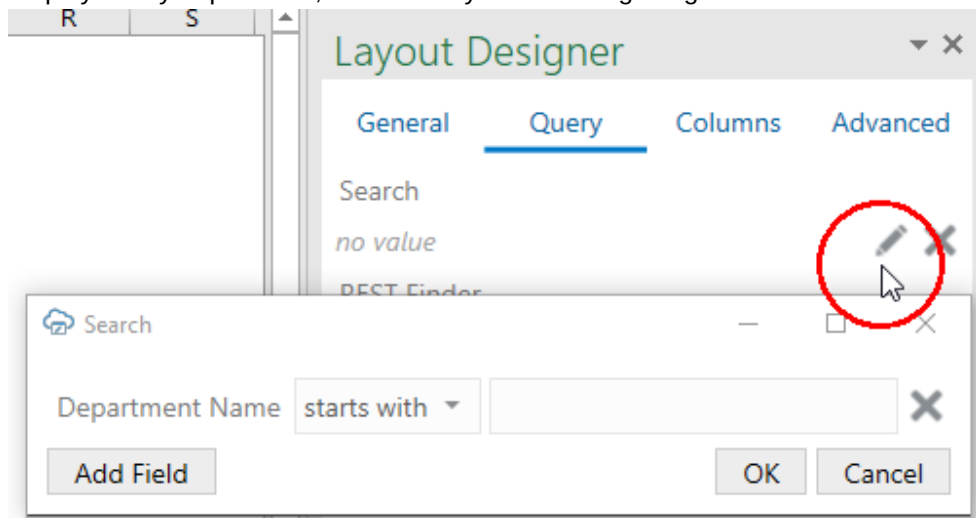
Tip:

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.

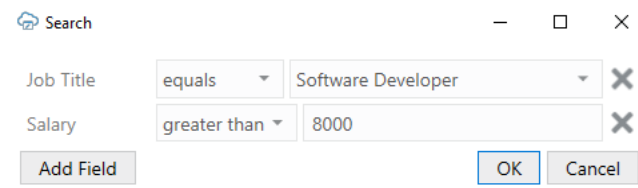
Use Search to Limit Downloaded Data

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 Query tab of the Layout Designer, 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 by the following image.



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:



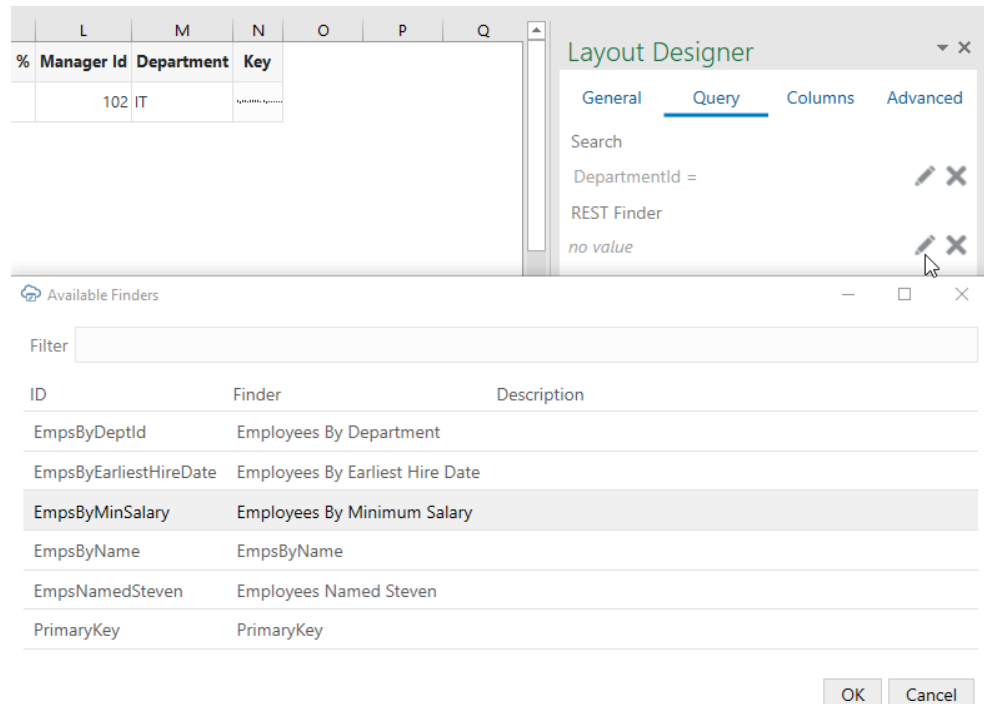
All searchable fields are available for you to choose from. You can adjust which fields show by selecting or deselecting **Searchable** in the Business Object Field Editor. Make sure that the service supports searching on that field.

4. Click **OK**.

Use REST Finders to Limit Downloaded Data

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

1. In the Excel ribbon, click **Designer**.
2. In the Query tab of the Layout Designer, click the Edit icon next to the REST Finder property, as shown in the following image. The service must be configured to expose the options that appear in the Available Finders window.



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 the following 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 add search parameters to the layout to include in the request to download data from the REST service.

Many REST services support the addition of various parameters in the query string portion of the URL, such as the following example.

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


In this example:

- "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:

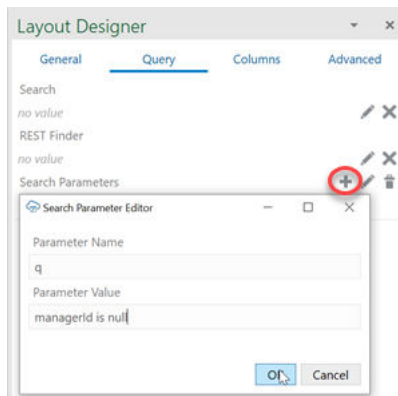
```
q=((firstName LIKE '*es*' ) or ((hireDate< "2001-01-13") and (department = 10)))
```

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 Query tab of the Layout Designer, 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.

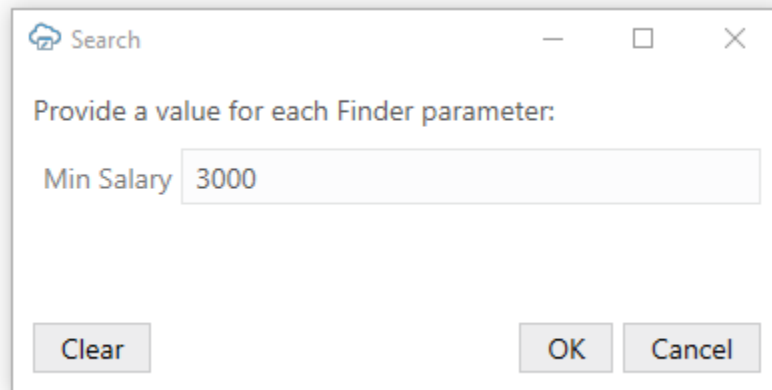
Download Behavior in Layouts that Use Search and REST Finders

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

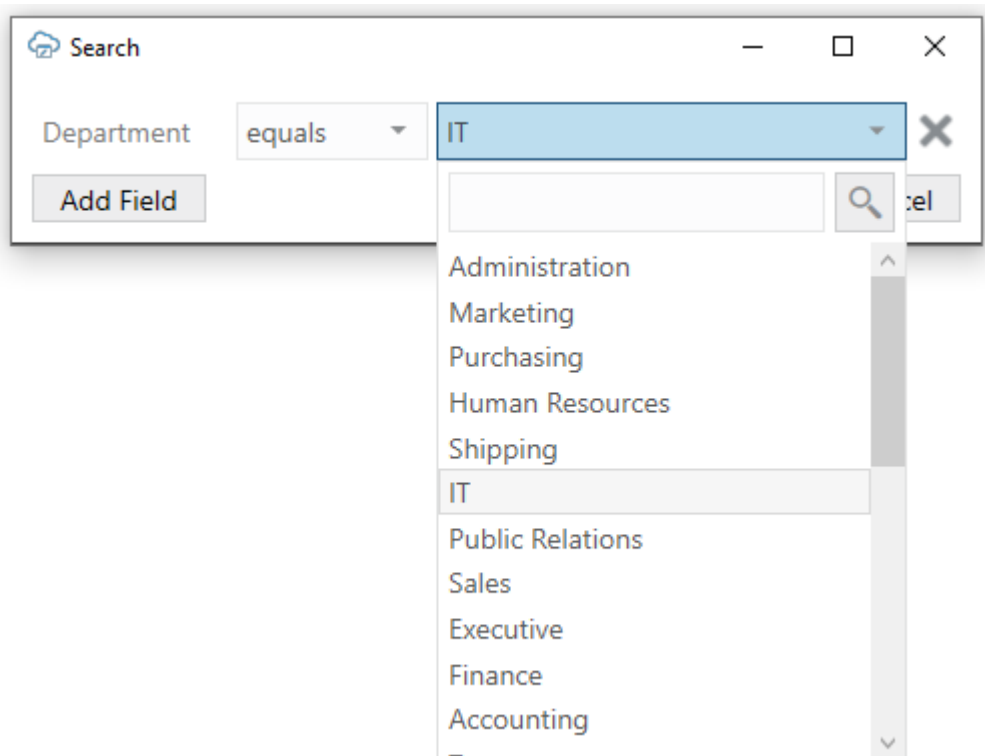
- If there are no parameters, there is no prompt.
- If there is exactly one parameter, a prompt appears that allows the user to specify a value for the parameter.
- If the finder has more than one parameter, a prompt appears that allows the user to specify values for each parameter.

 **Note:**

Only Oracle business object REST API services support REST 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 illustrated in the following image.



7

Use Lists of Values in an Excel Workbook

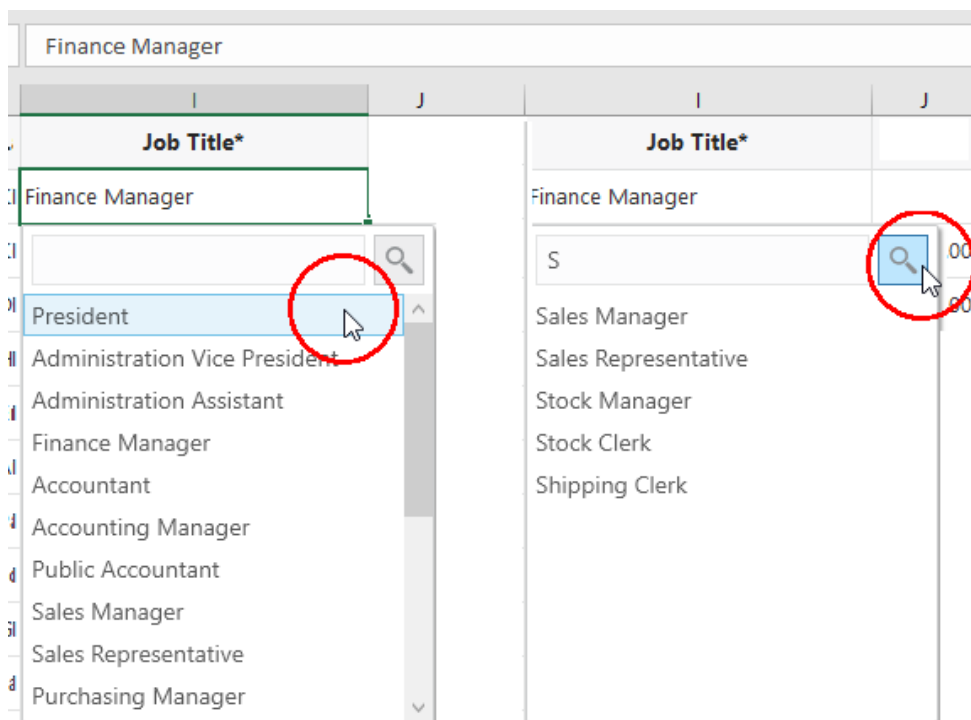
The Oracle Visual Builder Add-in for Excel enables you to associate a **list of values** with a business object field.

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 (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 available display values; the user can then select one of these values. 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](#). The following 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`.



The add-in supports multiple display fields for one list of values. In that case, values of all display fields are concatenated and the concatenated value is shown in a single Excel cell.

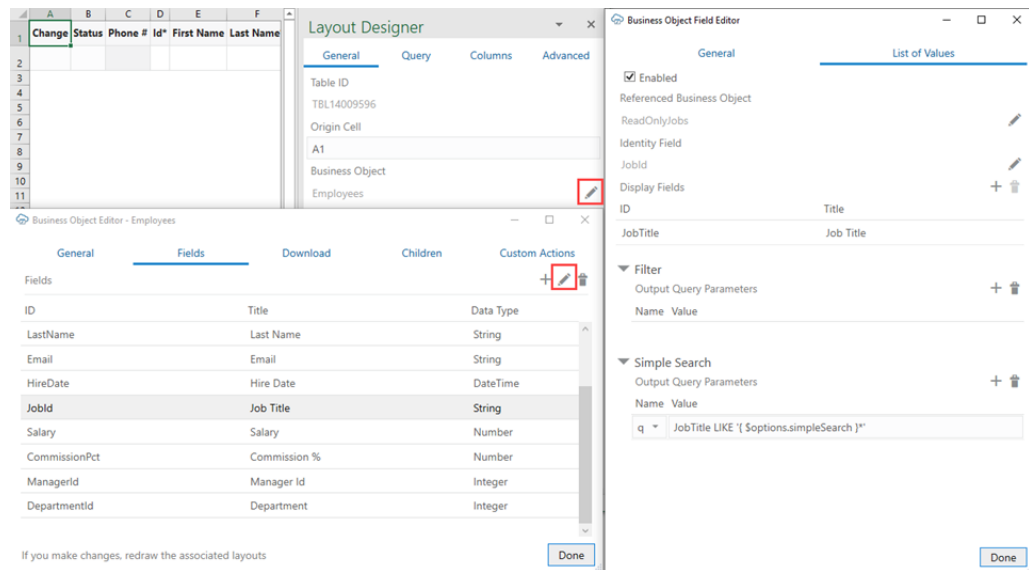
Note:

The list of values' data is cached in the workbook. 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 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.



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.
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.

For more detail about using the add-in's editors to modify the business objects that your workbook uses, see [Edit Service Descriptions and Business Objects](#).

Configure Filters for a List of Values

Define filters using name-value query parameters to let your users filter a field's list of values.

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 REST 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['DepartmentId'] }`

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.



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

Define search parameters to let your users enter a string to search for a value from a list of values.

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

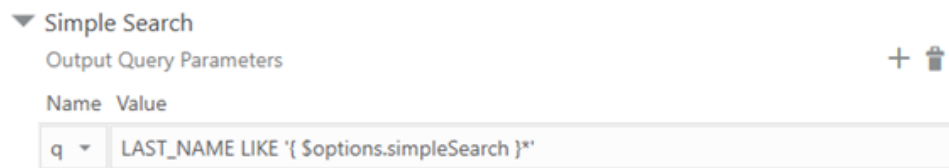
add-in does not validate the values, so if the user enters invalid information, the add-in will make invalid requests that return errors.

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 }*'
```



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, you want to 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 Cascading List of Values in a Layout

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

For 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, 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.

When the service meets the following requirements, the add-in detects and wires up the cascading list of values automatically:

- The cascading list of values resource must be configured using static links with Finders. For example, for Countries, States, and Cities, the item links for the parent business object must be configured as follows:

```
"links": {
  "CountryView1": {
    "operationRef": "http://servicehost/Countries/describe#/paths/~1Countries/get",
    "x-lov": {...}
  },
  "StateView1": {
    "operationRef": "http://servicehost/States/describe#/paths/~1States/get",
    "x-lov": {...},
    "parameters": {
      "finder": "ByCountryFinder%3BCurrentCountry%3D{CountryId}"
    }
  },
  "CityView1": {
    "operationRef": "http://servicehost/Cities/describe#/paths/~1Cities/get",
    "x-lov": {...},
    "parameters": {
      "finder": "ByCountryAndStateFinder%3BCurrentCountry%3D{CountryId}%2CCurrentState%3D{StateId}"
    }
  }
}
```

- For the POST/PATCH request to process attributes in order, the attributes must have dependencies configured properly. In our example, the State's dependencies array must be ["Country"] and City's dependencies array must be ["Country", "State"].

In the Excel worksheet, the add-in exposes the corresponding business object fields as table columns (Countries, States, and Cities using our example). You can see and configure these columns much like other columns; no extra configuration is required. To implement a cascading list of values in your worksheet, do not remove any column from the cascading list of values columns.

The add-in also exposes each field's Finder parameters as part of the Filter section in the Business Object Field Editor. Based on our example metadata, where State depends on Country and City depends on Country and State, the following name-value parameters will show in the Filter section of the corresponding field's list of values:

Table 7-1 Example Name-Value Parameters in Filter

Field	Name	Value
StateId	finder	ByCountryFinder%3BCurrentCountry%3D{CountryId}
CityId	finder	ByCountryAndStateFinder%3BCurrentCountry%3D{CountryId}%2CCurrentState%3D{StateId}

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 before you publish the workbook for data entry
- 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.

8

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 invoice business object might support a custom action called "approve". Users could download many invoices and approve many of them in a single upload.

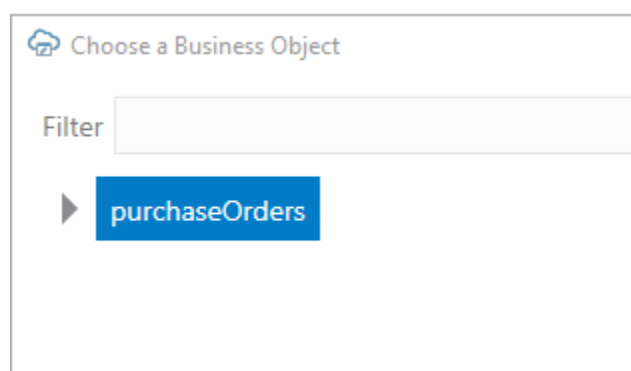
Use Custom Actions

When your REST API exposes a custom action, let's see how you can invoke the custom action in your Excel workbook, for example, a custom action called "Close" that's exposed by a purchase order business object to close purchase orders.

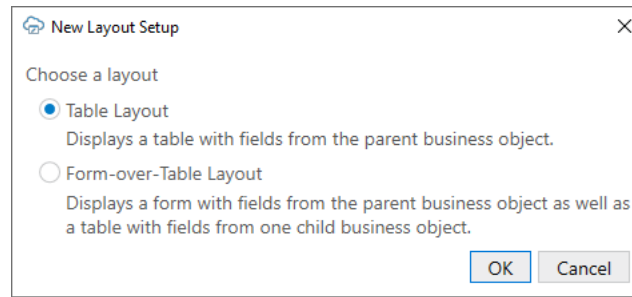
 **Note:**

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. For other limitations, see [Limitations, Known Issues, and Other Notes](#).

1. Create a layout for the custom action, for example, a Table layout for the "Close" 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, `purchaseOrders`.

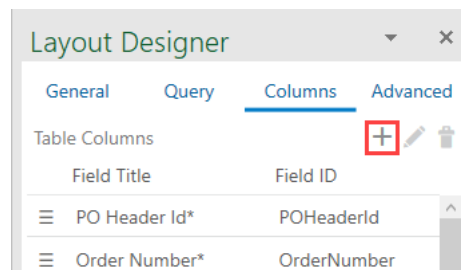


- d. Select **Table Layout** and click **OK**.

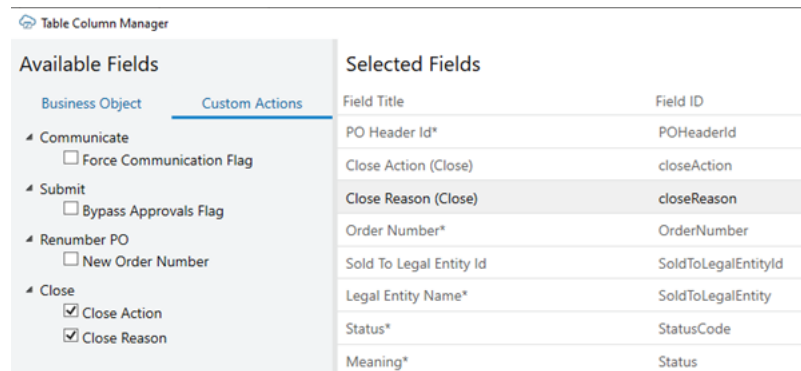


A table layout that includes column headers and a placeholder data row is created in the Excel workbook.

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 Close Action and the Close Reason payload fields.
 - a. In the Layout Designer, click **Columns**, then click Manage Columns (+).



- b. Click **Custom Actions** in the Table Column Manager.
 - c. In the Selected Fields pane, select the location where you want the payload fields' columns to be added. The columns will be inserted before the selected field.
 - d. In the Available Fields pane, select the payload fields (**Close Action** and **Close Reason**).



- e. Click **Done**.
 - f. Click **Redraw Layout**.

The newly added columns show in the worksheet table.

3. Mark rows for the custom action.

- a. Click **Download Data** to download data to the table.
- b. Update cells that correspond to the custom action's payload fields, in our example, cells that correspond to the **Close Action (Close)** and the **Close Reason (Close)** columns.

You can mark cells for action simply by entering values in cells that correspond to custom action columns. Alternatively, you can click the **Table Row Changes** menu and select **Mark for Action**.

	A	B	C	D	E	F
4	Change Status		PO Header Id*	Close Action (Close)	Close Reason (Close)	Order Number*
5	Close		100,000,012,676,405	Rejected	Over budget	1000364
6			420,601			1000138
7			100,000,012,303,183			1000241
8			100,000,015,118,920			1000454
9			423,537			1001756

 **Tip:**

To mark several rows for the custom action, enter values in the custom action column's cells (for example, *Rejected* in the *Close Action (Close)* column and *Over budget* in the *Close Reason (Close)* column), then copy and paste the values to other rows.

 **Note:**

When marking cells for action, if you enter a value in a custom action column's cell before making other edits in that row, the row is automatically marked for that custom action. For example, in the following image, adding *Rejected* in cell D2 marks the row for the *Close* custom action. Note how fields that don't correspond to a custom action in that row (*PO Header Id* and *Order Number*) can no longer be edited. Editing fields that don't involve custom actions in row 3 marks the row as an *Update*, but now, the custom action cells in that row (corresponding to the *Close Action (Close)* and *Close Reason (Close)* columns) cannot be updated.

	A	B	C	D	E	F
1	Change Status		PO Header Id*	Close Action (Close)	Close Reason (Close)	Order Number*
2	Close		100,000,012,676,405	Rejected	Over budget	1000364
3	Update		420,601			1000138
4			100,000,012,303,183			1000241
5			100,000,015,118,920			1000454
6			423,537			1001756

- c. If you used the **Mark for Action** option and only one custom action is defined for the business object, you're prompted to confirm. If more than one custom action is defined, you're prompted to select from a list of custom actions.

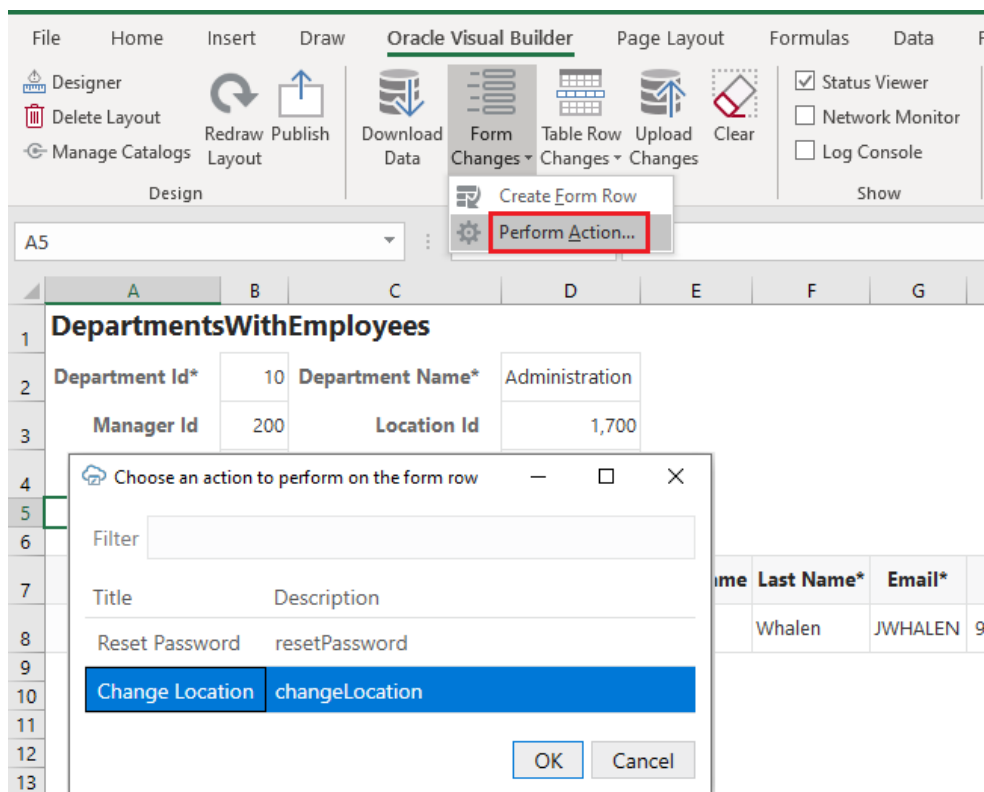
The selected table rows are updated in the Change column, and the Status Viewer indicates that the rows are pending the selected action. Up to this point, no REST requests have been made and the custom actions have not been performed. Data entry validation is performed at this point on custom action field columns for the current action.

- 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](#).

Use Custom Actions in a Form Row of a Form-over-Table Layout

When the REST service in your workbook exposes custom actions, you can perform those actions on a Form row in a Form-over-Table layout.

- 1. In a Form-over-Table layout, click **Download Data**.
- 2. Click the **Form Changes** menu and select **Perform Action**. You can perform a custom action only on an existing row (not on a pending Create row).



If multiple custom actions are found, you are prompted to select an action, as shown here. Otherwise, you are prompted to confirm the action.

- 3. If payload fields are required, provide a value for each payload field.

4. Click **Perform Action**.

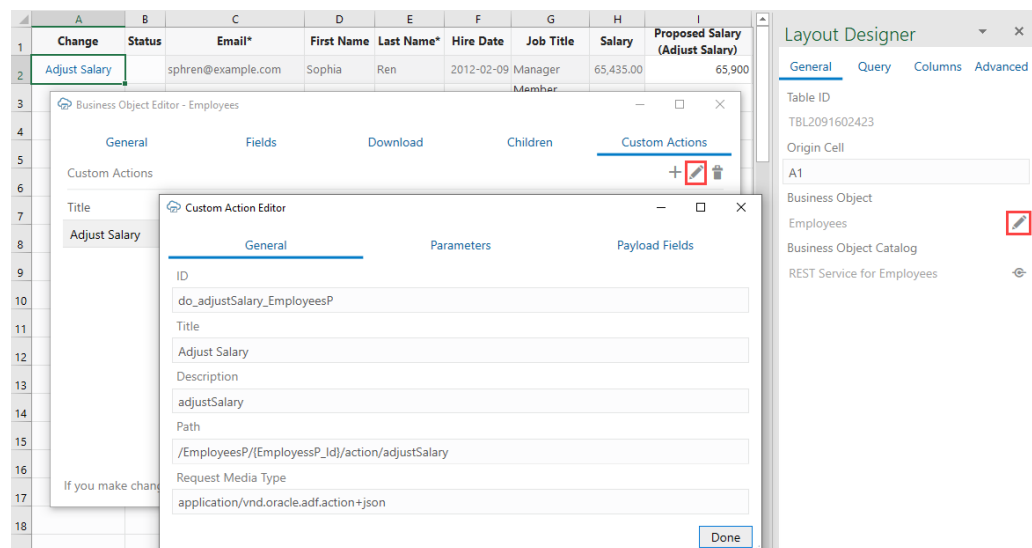
The result of the custom action is returned. You can also view details in the Status Viewer.

5. To refresh data in the form row, click **Download Data**. Form row data is not automatically refreshed after a custom action is performed, even if the action was successful.

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. Other properties of the custom action should generally be left as is. Payload fields can be edited by clicking the Edit button or double-clicking an entry in the list.



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 `close` custom action would appear in the `paths` collection:

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

"/purchaseOrders/{purchaseOrders_Id}/action/close": {
  "parameters": [
    {
      "$ref": "#/components/parameters/purchaseOrders_Id"
    }
  ],
  "post": {
    "summary": "close",
    "description": "close",
    "operationId": "do_close_purchaseOrders",
```

```

    "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": {
              "closeAction": {
                "type": "string",
                "nullable": true
              },
              "closeReason": {
                "type": "string",
                "nullable": true
              }
            }
          },
          "additionalProperties": false
        }
      }
    }
  }
}

```

Note the following:

- The path entry contains a path parameter for the row/item ID, for example, /purchaseOrders/{purchaseOrders_Id}/action/close"
- The end of the path entry (close) 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.

Limitations, Known Issues, and Other Notes

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.

When an Upload operation contains one or more custom action invocations, the entire operation will be performed on a row-by-row basis (one `PATCH/POST/DELETE` request made for each of the participating rows). For best performance of bulk Updates, Creates, and Deletes, avoid mixing custom action invocations into those bulk operations.

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.

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.

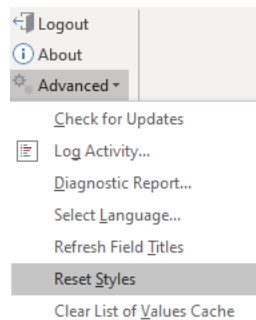
Once the styles are initialized, the add-in applies those styles to the integrated cells, according to the field's properties, at key points (such as during a data download). Typically, the styles are created once for a workbook and are never updated, but you can take advantage of the following options:

- [Reset Workbook Styles](#)
- [Choose Field Formats](#)

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 Columns tab of the Layout Designer, select the field you want to update and click the Edit icon.
2. In the Business Object Field Editor that appears, select an option in the **Format** drop-down. The default value is `Default`, which tells the add-in to apply the usual automatic styles without any override.

Remember, the format types shown to you are based on what's appropriate for the field's data type. In this image of an employee's workbook, the ID field uses the Integer data type and the available options let you format the ID with or without the thousands separator. The options also follow user preferences as defined in the Windows Region settings, so a US user will see a decimal point ($\pi = 3.14$) while a French user will see a decimal comma ($\pi = 3,14$).

The screenshot shows the 'Layout Designer' interface with the 'Columns' tab selected. The 'Business Object Field Editor' is open for the 'Id*' field. The 'Format' dropdown menu is expanded, showing three options: 'Default', 'With thousands separator and zero decimal digits (12,300)', and 'Without thousands separator and zero decimal digits (12300)'. The 'Without thousands separator and zero decimal digits (12300)' option is currently selected.

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 aborts.
3. For pending Create and Update operations, the rows are first validated locally, for example, data type, required, and so on (see [Understanding Data Validation](#)). Any failures are marked as failed and skipped (these rows do not produce requests on the business object service).
4. Pending changes are processed in the order of rows (from the top of the table to the bottom).
 - 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). For Oracle REST Data Services and other service types, there is one request per row.

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 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, and Delete 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](#).

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 [Understanding Data Validation](#)). 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:

```
{
  "closeAction": "Rejected",
  "closeReason": "Over budget"
}
```

- 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 (version 6)
 - 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.



Note:

If a row is marked for a custom action, the entire upload operation is performed row-by-row (batch requests are not used).

For information on custom actions, see [Custom Actions](#).

11

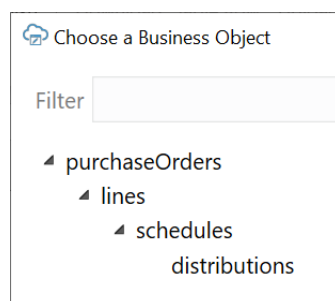
Use Multiple Layouts for Multi-level Business Objects

When business objects in your REST service have a parent-child relationship of 3 or more levels, you can link layouts in a hierarchy of dependent layouts to perform coordinated download, upload, and clear operations. Many operations triggered in a layout that's part of a hierarchy of dependent layouts work on all layouts in the hierarchy.

In a hierarchy of dependent layouts, 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). Additional Table layouts, created on separate worksheets, are based on descendant business objects, relative to the parent business object. In other words, the first Table layout uses a grandchild business object, the next one uses a great-grandchild business object, and so on. Each of these dependent layouts is linked 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.

In addition to coordinating operations such as download and upload across multiple layouts, there's another key distinction when using layouts in a dependent hierarchy: In the grandchild layout, the add-in will fetch all rows for each of the child rows and display them all together in the grandchild layout. This behavior applies to the grandchild and lower layouts in the hierarchy (basically, all Table layouts that follow the primary Form-over-Table layout).

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, **lines** manages the details of each order (such as the item to purchase, quantity, unit of measure, and price), **schedules** is used for details such as ship-to location and delivery date, and **distributions** is used for details about accounting or the project. Each purchase order may contain multiple lines, each line may be broken down into multiple schedules, and each schedule may have multiple distributions.

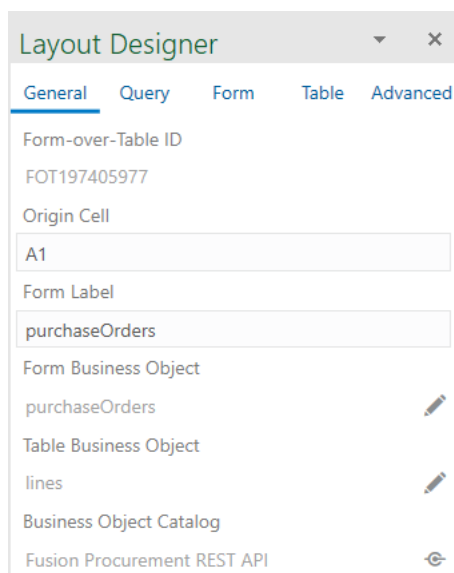
 **Note:**

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




Let's use this example to create a hierarchy of dependent layouts that mirrors your business object hierarchy. Remember, the primary layout in the hierarchy must be a Form-over-Table; the dependent layouts must be Tables.

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.



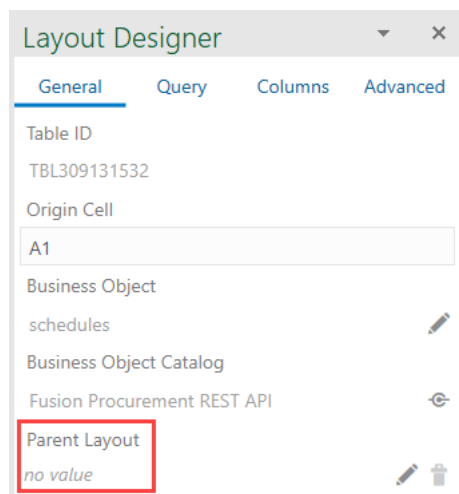
The screenshot shows the 'Layout Designer' window with the following configuration:

Property	Value	Icon
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	

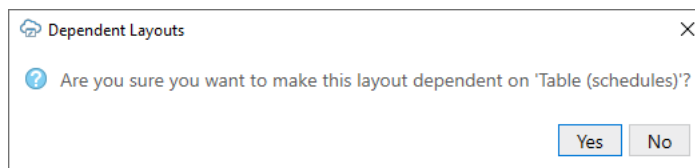
2. Create a Table layout for each of the other levels in your business object hierarchy. Ensure that you select the same business object catalog that is used by your primary Form-Over-Table layout.
 - a. Click the New Sheet icon to add a new worksheet.
 - b. Click **Designer**.
 - c. Choose the business 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.

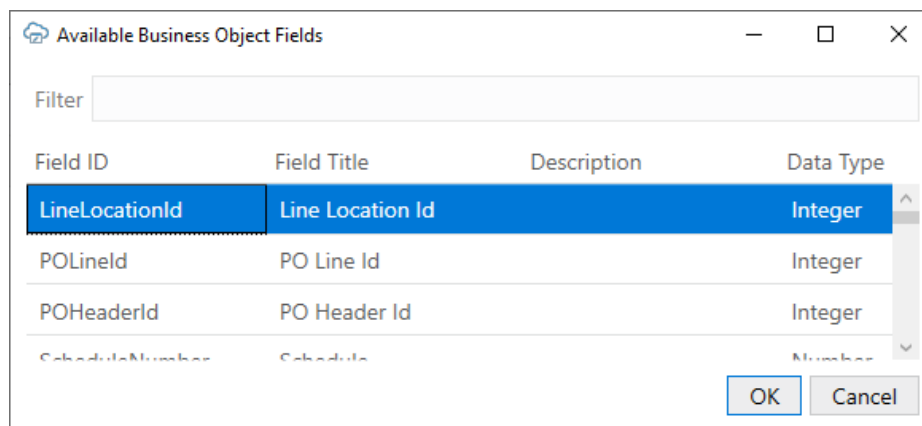
A Table layout is created for each level in the business object hierarchy. Notice the **Parent Layout** field in the Layout Designer's General tab, shown here for the schedules layout. This field shows only in layouts where the business object is a child of another business object in the same business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



3. Starting from the lowest level in the hierarchy, choose the parent for each layout.
 - 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. When prompted, click **Yes** to confirm the parent layout in the hierarchy, for example:

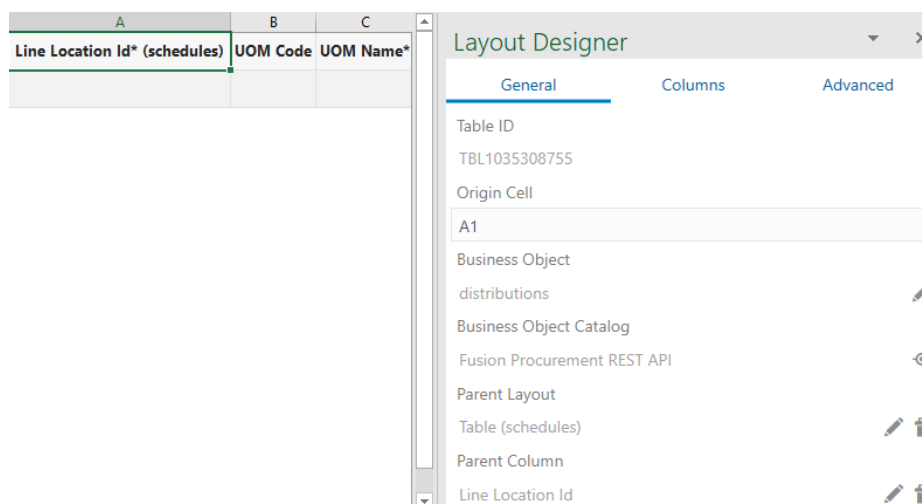


- c. For layouts that are three levels or deeper in the hierarchy (for example, schedules and distributions), if you want to support creating new rows and create is enabled in the Table's capabilities, click the Choose Parent Column icon (🔗) and select a field from the parent layout to uniquely identify the parent row for a pending create row in the child layout. The field must be exposed as a column in the parent table, for example, the Line Location Id field which exists as a column in the schedules 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.) Click **OK**.



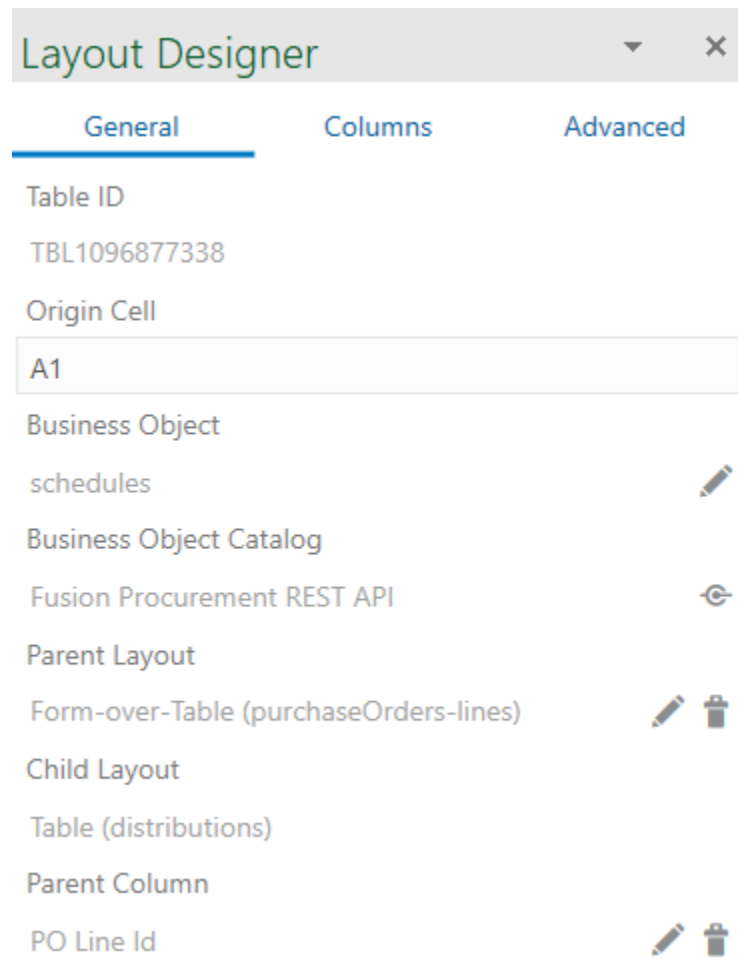
- d. Click **Redraw Layout** for both the child and parent worksheets.

A column for the corresponding parent field is created in the child layout, for example, the **Line Location Id (schedules)** column as shown here:



- e. Repeat steps **a** to **d** 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.



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](#)).

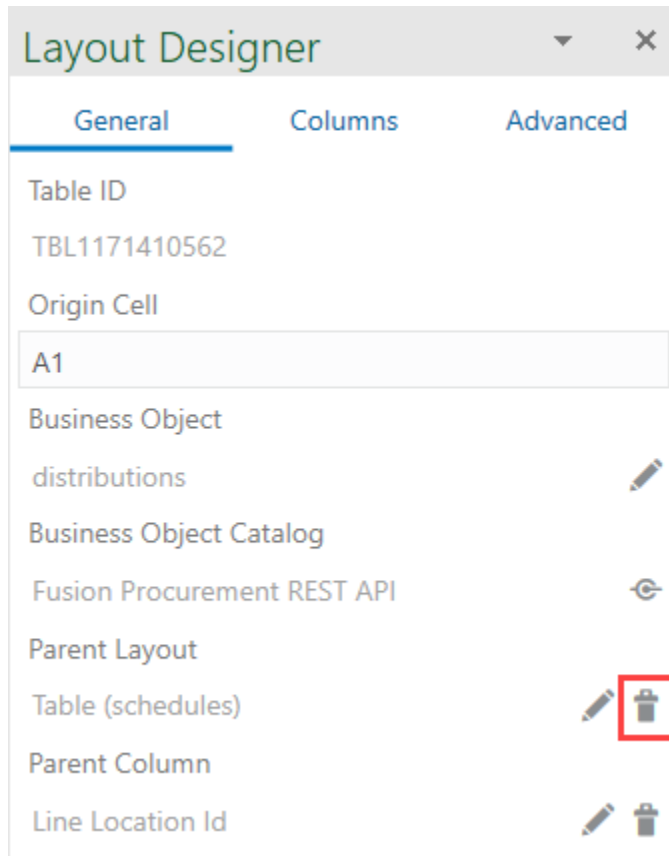
Before you publish and distribute your workbook to users, test the workbook to ensure that download, upload, or clear operations work on all layouts in the hierarchy (see [Manage Data for Layouts in a Dependent Hierarchy](#)).

Delete a Dependent Layout

When your layout is part of a hierarchy of dependent layouts, the layout cannot be deleted without first removing its dependency in the layout hierarchy.

To delete a dependent layout:

1. Open the Layout Designer of the Excel worksheet whose layout you want to delete.
2. In the General tab, click the Remove Dependency icon (🗑️) next to Parent Layout.



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

12

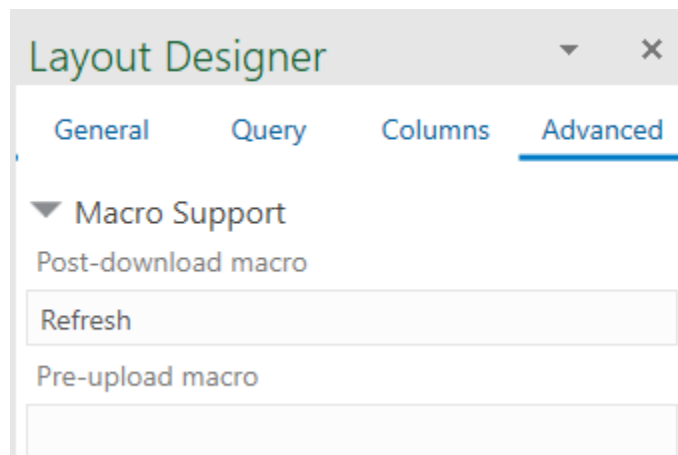
Use Macros in an Integrated Excel Workbook

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.



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

ErrorHandler:
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 configuration of an Excel workbook, you can publish it for users to do data entry work. Publishing creates a copy of the workbook with the design tools hidden and worksheet protection turned on for each worksheet with a layout.

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.
4. In the Oracle Visual Builder tab, click **Clear** for each layout in the workbook.
5. In the Advanced drop-down, click **Clear List of Values Cache**.
6. Save this source version of the workbook.

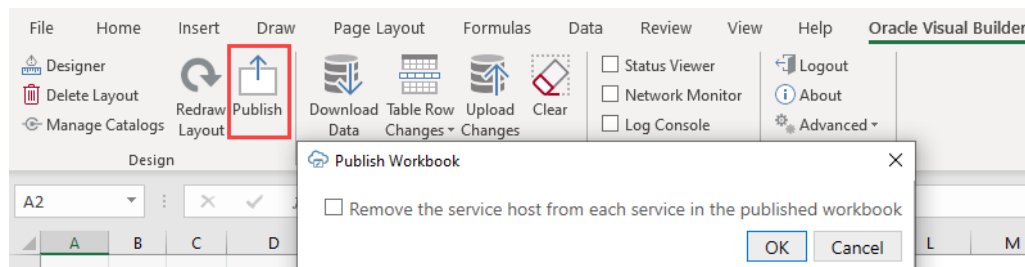
Note:

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

Once you configure and test your workbook, make a backup copy of the unpublished (source) workbook in case you need to make further configuration changes at a later date post-publication. Consider adding a file name suffix of `-src` for the source workbook. Then, remove the suffix in the published copy. Excel will not allow you to use the same file name for both workbooks. You click the **Clear** button before you publish it. Clearing data works on the current layout; if your workbook has multiple layouts, you should ideally clear each layout.

Use Excel's Inspect Workbook feature to review and remove your personal information from the workbook before you distribute it. You access the Inspect Workbook feature from Excel's File menu. Deselect the check box next to the Hidden Worksheets entry in the Document Inspector where you choose the content to inspect and potentially remove. You must not remove hidden worksheets from the Excel workbook that you distribute. The Oracle Visual Builder Add-in uses hidden worksheets to integrate the Excel workbook with the REST service.

Once you are ready to distribute the workbook to users, click **Publish**. In the Publish Workbook window that appears, select **Remove the service host from each service in the published workbook** if you want users to enter the service host when they open the published workbook, and click **OK**.

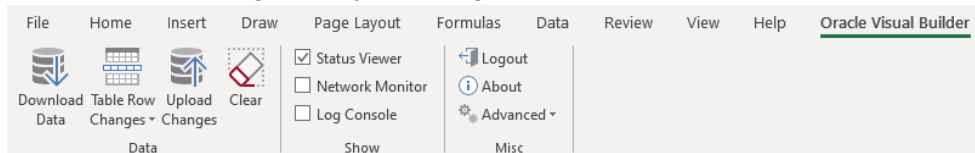


Accept the default file name and directory location values for the published workbook or select alternative values in the subsequent window, then click **Save**. You can now distribute the published workbook to users.

When a user performs an action that requires access to the REST service in a published workbook where you removed the service host value, a Service Configuration window prompts the user to enter their service host value before the action can be executed. Actions that require access to the service include the Download Data and Upload Changes commands.

There are a number of differences with the source workbook:

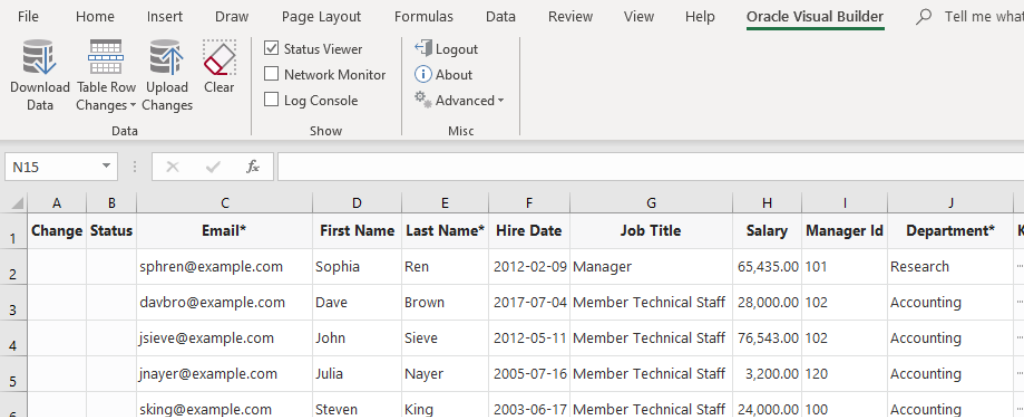
- The design tools do not appear in the Oracle Visual Builder tab of the Excel ribbon (Designer, Delete Layout, and Publish buttons), as shown in the following image.
- Excel's sheet protection is on. This mode enables true read-only behavior for cells that should be read-only. It also prevents the user from performing various other Excel actions that might disrupt the integration with the service.



14

View and Edit Data in an Excel Workbook

In Microsoft Excel, select the **Oracle Visual Builder** tab to perform operations and work with data in a workbook.



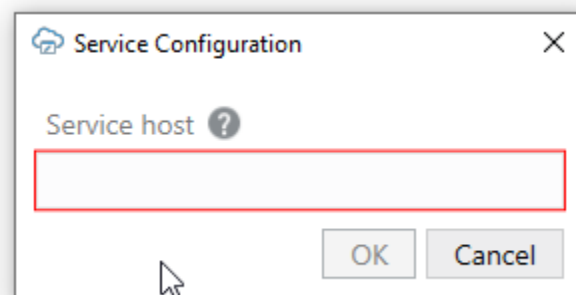
The screenshot shows the Microsoft Excel interface with the Oracle Visual Builder tab selected. The ribbon includes options like Download Data, Table Row Changes, Upload Changes, Clear, Status Viewer, Network Monitor, Log Console, Logout, About, and Advanced. Below the ribbon is a data table with the following columns: Change, Status, Email*, First Name, Last Name*, Hire Date, Job Title, Salary, Manager Id, Department*, and K.

	A	B	C	D	E	F	G	H	I	J	K
1	Change	Status	Email*	First Name	Last Name*	Hire Date	Job Title	Salary	Manager Id	Department*	K
2			sphren@example.com	Sophia	Ren	2012-02-09	Manager	65,435.00	101	Research	...
3			davbro@example.com	Dave	Brown	2017-07-04	Member Technical Staff	28,000.00	102	Accounting	...
4			jsieve@example.com	John	Sieve	2012-05-11	Member Technical Staff	76,543.00	102	Accounting	...
5			jnayer@example.com	Julia	Nayer	2005-07-16	Member Technical Staff	3,200.00	120	Accounting	...
6			sking@example.com	Steven	King	2003-06-17	Member Technical Staff	24,000.00	100	Accounting	...

For any given layout, you can:

- View existing rows
- Edit existing rows
- Create new rows
- Delete existing rows
- Perform custom actions on existing rows, for example, an "Approve" or "Reject" action for an Invoice business object. These kind of special actions can be performed on certain business objects depending on the context. For more information, see [Perform Custom Actions in a Table or Form-over-Table Layout](#).

If your action requires access to the REST service and the service host is missing (because it was removed when the workbook was published), a Service Configuration window prompts you to enter the service host value. Actions that require access to the service include the Download Data and Upload Changes commands.



(See [REST Service Support](#) for more about REST operations.)

When working with data in the Excel workbook, remember the following guidelines:

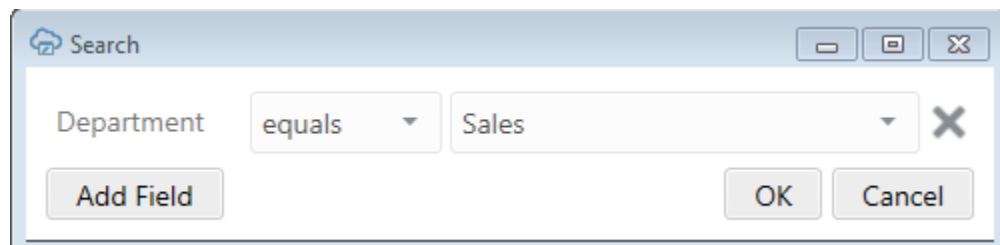
- Never edit the Key column, the last column in the table.
- Avoid using the following Excel features with the add-in. The following is a sample list of Excel features that do not work well with the add-in. Other Excel features not listed may also not work well with the add-in.
 - Do not use the Protect Sheet or Workbook features of Excel.
 - Do not attempt to re-arrange the layout of the integrated worksheet.
 - Do not use the Mark as Final command to make the Excel workbook read-only.
 - Do not delete anything from the integrated worksheet using Excel's delete features, including the Delete key.

You can remove all data from the workbook, including any pending changes that have not yet been uploaded to the REST service by clicking the **Clear** button. This button does not make any calls to the service and does not change data in the service.

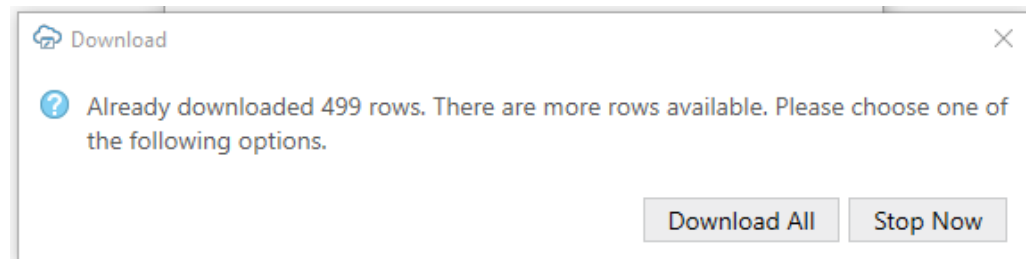
Download Data to the Workbook

Download data to the workbook using the **Download Data** button in the Oracle Visual Builder tab. The workbook prompts you for a user name and password the first time you connect to the service that the workbook is configured to use.

If you configured search options for download, a window or windows appear where users specify the value(s) to search on, as shown in the following example where data for the Sales department will be downloaded. [Configure Search Options for Download](#) describes how you configure search options for the workbook.



If the business object supports pagination, the add-in retrieves one page of rows, then prompts the user whether to attempt downloading more rows. You can view if the business object supports pagination, and change the default download limit of 499, in the Download tab of the Business Object Editor.



When download is complete, the add-in updates the table in the worksheet with data retrieved from the web application.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Salary	Commission %	Manager Id	Department	Key	
2			145	John	Russell	JRUSSEL	011.44.1344.429268	10/1/2004 12:00 AM	Sales Manager	14,000.00	0.40	100	Sales	-----	
3			146	Karen	Partners	KPARTNER	011.44.1344.467268	1/5/2005 12:00 AM	Sales Manager	13,500.00	0.30	100	Sales	-----	
4			147	Alberto	Errazuriz	AERRAZUR	011.44.1344.429278	3/10/2005 12:00 AM	Sales Manager	12,000.00	0.30	100	Sales	-----	
5			148	Gerald	Cambrault	GCAMBRAU	011.44.1344.619268	10/15/2007 12:00 AM	Sales Manager	11,000.00	0.30	100	Sales	-----	
6			149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	1/29/2008 12:00 AM	Sales Manager	10,500.00	0.20	100	Sales	-----	
7			150	Peter	Tucker	PTUCKER	011.44.1344.129268	1/30/2005 12:00 AM	Sales Representative	10,000.00	0.30	145	Sales	-----	

Status

▼ Notifications

Download completed successfully.

Downloaded 34 rows.

Elapsed time: 00:00:01

▼ Selected row status

No changes pending.

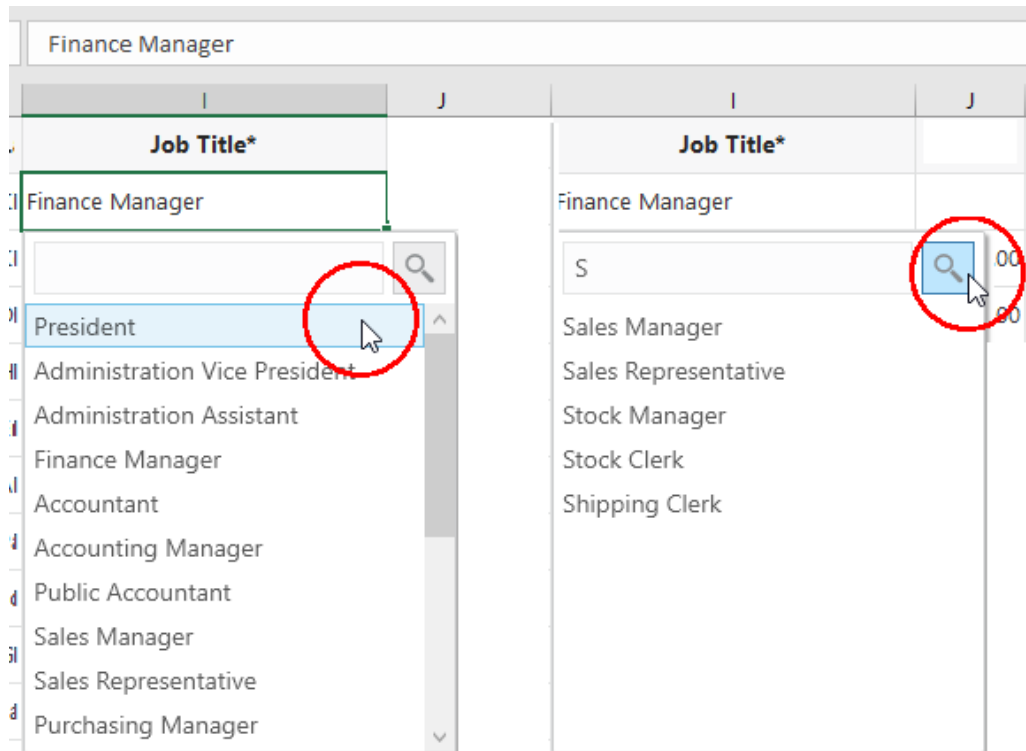
Edit Downloaded Data in the Workbook

Update data downloaded to a workbook by editing editable cells that contain the downloaded data.

The following image shows three examples where a user has edited data in the table. The Change column for the first row displays an `Update` message that indicates the user has updated this row with required and valid values. The Change column for the second row also displays an `Update` message, but its Status column displays an `Invalid` message and a red border appears around the Hire Date* column's cell to indicate that a value is required in that cell's field. Finally, in the third row, the user attempted to input `Software` in the field for Department. As `Software` is not a valid choice for this cell's list of values, a red border appears around this cell.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Salary	Commission %	Manager Id	Department	Key
2	Update		100	John	King	SKING	515.123.4567	6/17/2003 12:00 AM	Finance Manager	24,000.00			Executive	----
3	Update	Invalid	101	Neena	Kochhar	NKOCHHAR	515.123.4568		Administration Vice President	17,000.00			100 Human Resources	----
4	Update	Invalid	102	Lex	De Haan	LDEHAAN	515.123.4569	1/13/2001 12:00 AM	Finance Manager	17,000.00			100 Software	----

When columns have an associated list of values, you can select a value in a search-and-select window. In this case, the Change column displays `Update`. Alternatively, enter one or more of the starting characters for other values, then click the Search icon to filter the values based on your input. The following composite image illustrates both these scenarios. For the latter scenario, the user entered `S` in the search box next to the Search icon to find all display values that start with `S` (`Sales Manager`, `Sales Representative`, and so on).



Note:

You can alter the search behavior of the list of values using the Business Object Field Editor described in [Use Lists of Values in an Excel Workbook](#).

Understanding Data Validation

As you enter values in cells associated with layouts, the add-in validates the cell's content based on the business object field definition.

	A	B	C	D	E	F	G	H	I
	Change	Status	Email*	First Name	Last Name*	Hire Date	Job Title	Salary	Proposed Salary (Adjust Salary)
1	Adjust Salary		sphren@example.com	Sophia	Ren	2012-02-09	Manager	65,435.00	65,900
2	Update	Invalid	davbro@example.com	Dave	Brown	2017-07-04	Research	28,000.00	
3	Create	Invalid							
4			Email	hn	Sieve	2012-05-11	Member Technical Staff	76,543.00	
5			A value is required. Enter a value.	lia	Nayer	2005-07-16	Member Technical Staff	3,200.00	
6				even	King	2003-06-17	Member Technical Staff	24,000.00	

Status x

▼ Notifications

▼ Selected row status

Pending create.

Data entry errors found.

Validation occurs when you complete data entry in a cell; in other words, as soon as you enter or edit a value in a cell and move on to another cell or area. Validation also occurs at other key points:

- When a new row is added to the data table
- When a form create starts
- At the beginning of an upload.

When a row is marked for a custom action, the custom action's payload fields also receive the same validation. See [Custom Actions](#).

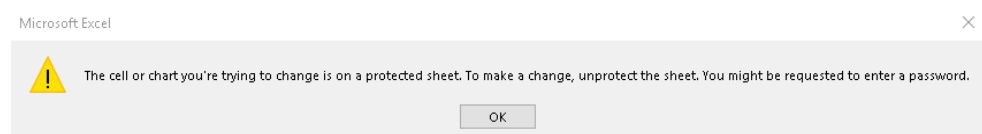
Typically, the add-in determines whether a cell's value is consistent with the expected data type. For example, you can't enter a word, say, `book`, in a field that expects a number like `2,000`. It also checks whether a required cell is missing a value and whether the value in a cell associated with a list of values is valid for that list. Required fields are designated by the workbook developer and must include a value for your changes to be uploaded successfully.

Local field properties are used for validation. Validation that is enforced by the REST service is not triggered at the points mentioned here. REST service validation is generally triggered by the requests sent during Upload (for details on Upload failure handling, see [Upload Changes to the REST Service](#)).

Understanding Read-Only Behavior

If a particular field is considered read-only, the add-in behaves as follows:

- It applies a "read-only" style to the corresponding cells to indicate that the value cannot be changed
- The cell's value is excluded from any subsequent REST requests
- The data entered by the user is not validated
- If the workbook is published with worksheet protection, any attempt to edit the cell results in a warning from Excel:



Create New Rows to Upload to the REST Service

Create new rows in the table using the **Insert Rows** option in the add-in's **Table Row Changes** menu or by using Excel options to insert a full row.

You can create new rows before or after downloading data to your workbook:

- To create new rows in a table with downloaded data, click **Table Row Changes** and select **Insert Rows**, or right-click and choose **Insert** from the Excel context menu (you can choose any Excel option to insert a row). You can create new rows either at the end of the data table or in the middle.

 **Tip:**

To insert multiple rows, select cells from multiple rows, then select **Table Row Changes > Insert Rows**. The add-in will create multiple pending Create rows for you to fill out as needed, as shown here:

	A	B	C	D	E	F	G	H	I	J
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Salary
2			100	Steven	King	SKING	515.123.4567	6/17/2003 12:00 AM	President	24,000
3	Create		209	John	Jones	JJONES		6/6/2019 12:00 AM	Accountant	12,000
4	Create	Invalid								
5	Create	Invalid								
6			101	Neena	Kochhar	NKOCHHAR	515.123.4568	9/21/2005 12:00 AM	Job Title*	ent
7			102	Lex	De Haan	LDEHAAN	515.123.4569	1/13/2001 12:00 AM		ent
8			103	Alexander	Hunold	AHUNOLD	590.423.4567	1/3/2006 12:00 AM		ent

- To create several new rows without downloading data, if create is enabled for the Table, add values in the empty row that appears below the column headers. This empty row is known as the placeholder row:

	A	B	C	D	E	F	G	H	I	J	K
1	Change	Status	First Name	Last Name*	Hire Date	Email*	Job Title	Salary	Manager Id	Department*	Key
2											
3											
4											

Once you enter data in the placeholder row (for example, in the cells for First Name), the row below becomes one where data values can be entered, and so on. This method is not available in published workbooks due to worksheet protection. Instead, you can select a cell in the first row after the table and select **Insert Rows** from the **Table Row Changes** menu.

	A	B	C	D	E	F	G	H	I	J
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Sal
2	Create	Invalid		John						
3	Create	Invalid		Jack						
4	Create		207	James	Mac	jmac@ exampl		6/6/2019 12:00 AM	Administra tion	
5	Create	Invalid		Jimmy						
6										
7										
8										

Any time a new row is created, it is validated for data entry. An **Invalid** message appears in the Status column if the row contains a cell where you are required to enter a value and are yet to do so, or you have entered an incorrect value (such as an unexpected data type). A red border also appears around the cell where a value is required or invalid. See [Understanding Data Validation](#).

Delete Data from the REST Service

Mark rows for deletion from the REST service using the **Mark for Delete** option in the **Table Row Changes** menu.

To mark a row for deletion from the service:

1. Select a cell in the table row that you want to delete from the service. If you want to select a range of table rows to mark for deletion, hold down your keyboard's Shift key and select the first and last row in the range of table rows that you want to delete.
2. Click the **Table Row Changes** menu, then **Mark for Delete**. A **Delete** message appears in the Change column and the add-in changes the style applied to the data in the table rows, as shown in the following image where three rows in the table are marked for deletion:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Salary	Commission %	Manager Id	Department	Key
107			205	Shelley	Higgins	SHIGGINS	515.123.8080	6/7/2002 12:00 AM	Accounting Manager	12,008.00		101	Accounting
108			206	William	Gietz	WGIEZT	515.123.8181	6/7/2002 12:00 AM	Public Accountant	8,300.00		205	Accounting
109	Delete		209	Bobby	Dooley	BDOOLEY		6/6/2019 12:00 AM	Accountant	12,000.00			
110	Delete		210	James	Jones	JJONES		6/6/2019 12:00 AM	Accounting Manager	14,000.00			
111	Delete		211	John	Evans	JEVANS		6/6/2019 12:00 AM	Public Accountant	12,000.00			

3. Click **Upload Changes**.
4. When prompted to confirm pending deletions, click **Yes**. The add-in sends delete requests for each row that you marked for deletion. For each successful deletion, the corresponding Excel row is removed. If the deletion fails, the Excel row remains with an error message.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job Title*	Salary	Commission %	Manager Id	Department	Key
107			205	Shelley	Higgins	SHIGGINS	515.123.8080	6/7/2002 12:00 AM	Accounting Manager	12,008.00		101	Accounting
108			206	William	Gietz	WGIEZT	515.123.8181	6/7/2002 12:00 AM	Public Accountant	8,300.00		205	Accounting
109														
110														
111														

Tip:

If you change your mind about deleting one or more rows that you have marked for deletion, select the rows and select **Unmark Pending Changes** from the **Table Row Changes** menu. Use **Unmark Pending Changes** before you upload changes from the Excel workbook; the option does not work after the changes are uploaded.

Upload Changes to the REST Service

Once you complete the changes that you want to make, click the **Upload Changes** button to upload all the changes to the REST service.

The Excel add-in performs all the requested operations, as seen in the Change column. Review the Status column to see which rows succeeded and failed. The Status column displays a **Create Failed** message or an **Update Failed** message in the cell of a row that the add-in failed to upload and the Table Row Status

appears in Excel's task pane to provide additional information on the failure, as in the following example where a required value for Hire Date was not entered. In the following example, you need to enter a hire date and click **Upload Changes** to try and upload the modified data again. Successfully deleted rows are removed from the Excel worksheet while failed attempts to delete a row result in the row remaining in the Excel worksheet. You can inspect the reason for the failure to delete in the status in Excel's task pane.

	A	B	C	D	E	F	G	H	
1	Change	Status	Id*	First Name	Last Name*	Email*	Phone #	Hire Date*	Job
104			202	Pat	Fay	PFAY	603.123.6666	8/17/2005 12:00 AM	Marketing Repr
105			203	Susan	Mavris	SMAVRIS	515.123.7777	6/7/2002 12:00 AM	Human Resourc
106			204	Hermann	Baer	HBAER	515.123.8888	6/7/2002 12:00 AM	Public Relations
107			205	Shelley	Higgins	SHIGGINS	515.123.8080	6/7/2002 12:00 AM	Accounting Man
108		Create Succeeded	207	John	Jones	JJONES		6/6/2019 12:00 AM	Administration v
109	Create	Create Failed	208	Jenny	Jones	JEJONES			Sales Manager
110			206	William	Gietz	WGIETZ	515.123.8181	6/7/2002 12:00 AM	Public Accountar

Status

▼ Notifications

Upload completed.

Create rows: 1 succeeded, 1 failed, 2 total.

Elapsed time: less than one second

▼ Selected row status

Create Failed

✖ A value is required for column: Hire Date*

How the add-in uploads modified rows depends on the type of service that your workbook uses. For workbooks that use Oracle business object REST API services, the add-in uploads the modified rows in batches. For other types of services, the add-in uploads modified rows one row at a time.



Tip:

If you change your mind about uploading changes for a particular row, select the row, then from the **Table Row Changes** menu, select **Unmark Pending Changes**. The add-in won't include the row when it sends other rows back to the service for update, creation, or deletion.

View and Edit Data in a Form-over-Table Layout

Viewing and editing data in a Form-over-Table layout is similar in many ways to viewing and editing data in a Table layout.

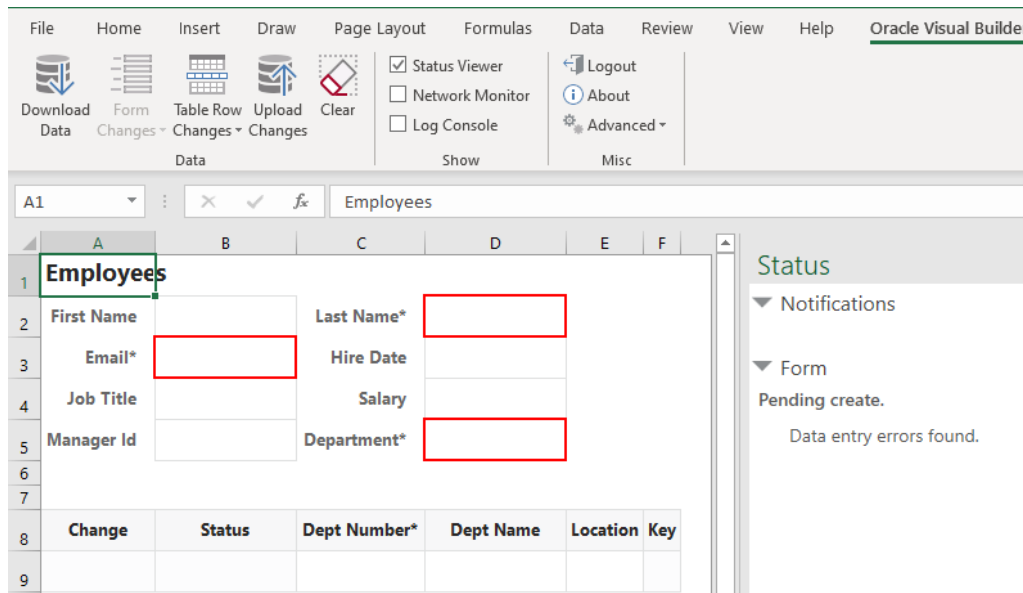
For example, you use the **Download Data** button to download data from the Form-over-Table layout's business objects. Much like table layouts, you may be prompted to do a search at the beginning of a download. Unlike a table layout, the Form-over-Table displays the first row found from the search, plus all the children of that first parent row.

You can also update the form fields in a Form-over-Table if the business object for the form fields supports update. The add-in performs data entry validation before it attempts to upload your updates. Ensure that you fix any data entry failures before the add-in uploads changes. For a form field configured to use a list of values, the behavior is the same as described for the Table layout. That is, a search-and-select window appears when you select the form field. In a published workbook or when worksheet protection is enabled, read-only form fields cannot be edited. You can make a form read-only by deselecting the **Update Enabled** check box under Form Capabilities in the Layout Designer's Advanced tab.

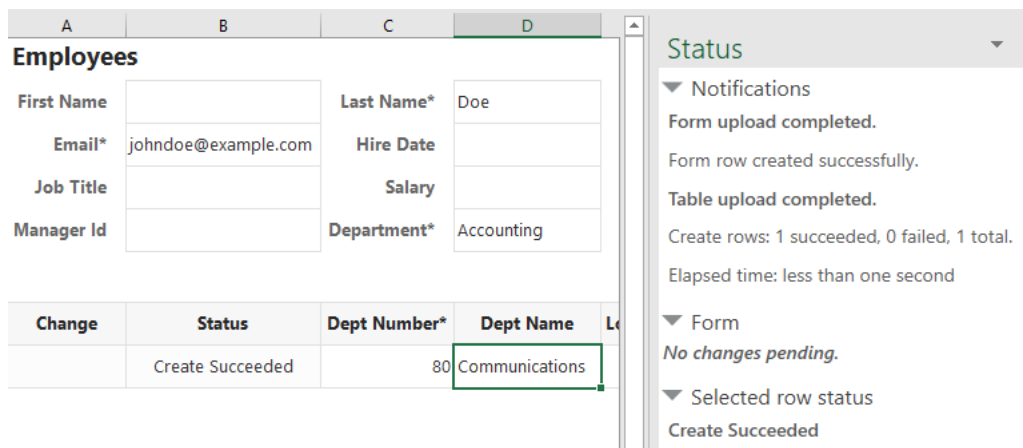
Create a Parent Row in a Form-over-Table Layout

If create is enabled for the form, you can select **Create Form Row** in the **Form Changes** menu to place the layout in create mode. When you first click **Create Form**

Row, all the form fields are blank and the child table has no rows, as shown in the following image:



You can then enter form field values, even create new child table rows. All changes are marked as a pending create, until you click **Upload Changes**. The add-in validates the data before it attempts to upload updates, and any data entry failures need to be addressed before the changes can be uploaded.



Perform Custom Actions in a Table or Form-over-Table Layout

Perform custom actions (for example, an action to close purchase orders) by downloading records of purchase orders and using the custom action on rows of data, then uploading the changes back to the service in a single upload.

As a business user, you don't need to do anything special to use custom actions in a data table. Click **Download Data** to download data to your table, then update a custom action's cell values much as you would any other field's cell value. In the following Table layout, rows marked as `Close` are those where the user has updated cell values corresponding to the `Close Action (Close)` and `Close Reason (Close)` custom action columns.

	A	B	C	D	E	F
1	Change	Status	PO Header Id	Close Action (Close)	Close Reason (Close)	Order Number
2	Close		265,134	Rejected	Over budget	PSR-265134
3	Close		265,135	Approved		PSR-265135
4	Close		265,136	Rejected	Over budget	PSR-265136
5			265,137			PSR-265137
6			265,222			PSR-265222

When you add or update a value in a custom action cell first, the row is marked for that custom action in the `Change` column. Other cells in that row that don't involve custom actions are grayed out and cannot be edited. But if you add or update a value first in a cell that doesn't involve a custom action, the row is marked as an `Update` in the `Change` column and the custom action cells in that row are grayed out (as shown in the following image). In other words, the first action you perform on a row determines that row's pending action.

	A	B	C	D	E	F
1	Change	Status	PO Header Id	Close Action (Close)	Close Reason (Close)	Order Number
2	Close		265,134	Rejected	Over budget	PSR-265134
3	Close		265,135	Approved		PSR-265135
4	Close		265,136	Rejected	Over budget	PSR-265136
5	Update		265,139			PSR-265137
6			265,222			PSR-265222

You can also mark rows for a custom action by selecting **Mark for Action** from the **Table Row Changes** menu. With this option, if only one custom action is defined, you'll be prompted to confirm. If more than one custom is defined, you'll be prompted to select from a list of available actions:

The screenshot displays the Oracle APEX interface. The top navigation bar includes tabs for File, Home, Insert, Draw, Page Layout, Formulas, Data, Review, and View. The 'Table Row Changes' menu is open, showing options: 'Mark as Changed', 'Mark for Action...' (highlighted with a red box), and 'Unmark Pending Changes'. Below the menu is a table with the following data:

1	Change Status	PO Header Id	Close Action (Close)	Close Reason (Close)	Order Number
2	Close	265,134	Rejected	Over budget	PSR-265134
3	Close	265,135	Approved		PSR-265135
4	Close	265,136	Rejected	Over budget	PSR-265136
5		265,137			PSR-265137
6		265,222			PSR-265222

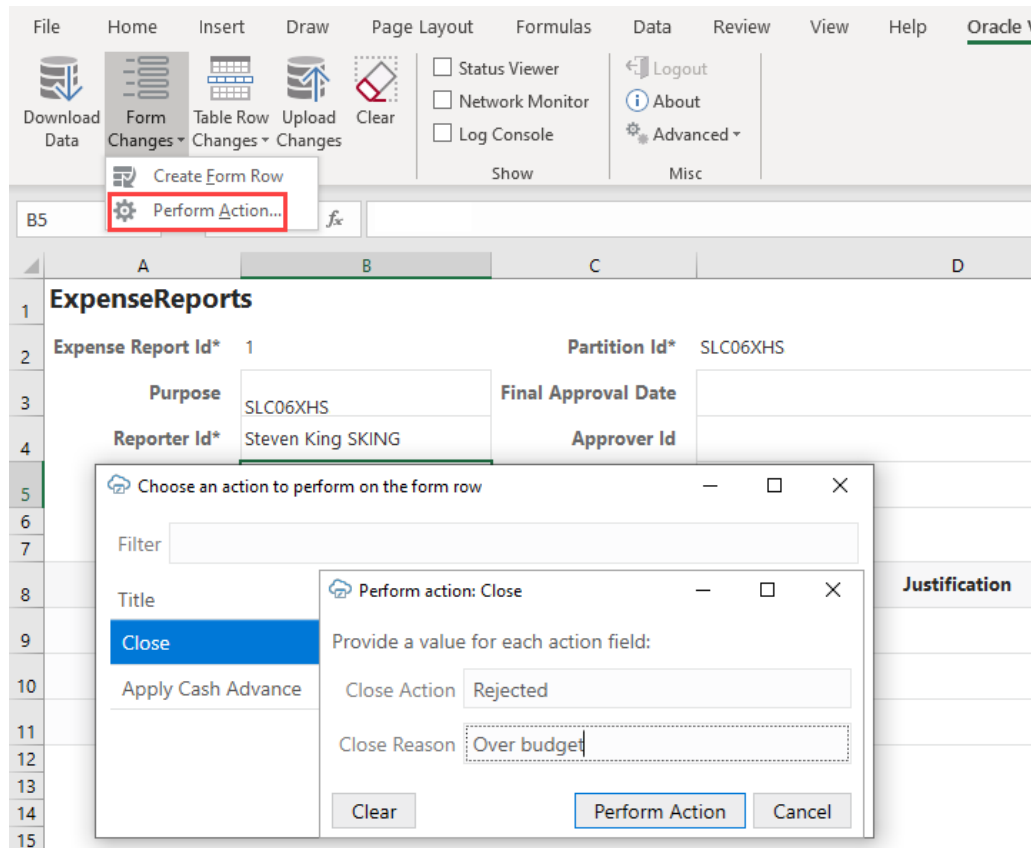
A dialog box titled 'Choose an action to perform on the selected rows' is open, showing a list of actions:

Title	Description
Communicate	communicate
Submit	submit
Renumber PO	renumberPO
Close	close

The 'Close' action is selected and highlighted in blue. The dialog also includes a 'Filter' input field and 'OK' and 'Cancel' buttons.

When a row is marked for a custom action, the action's payload fields also receive the same validation, as described in [Understanding Data Validation](#).

If you're working with a Form-over-Table layout, select **Perform Action** in the Form Changes menu, then follow the prompts to perform custom actions on a Form row, as shown here where you're prompted to select an action, then provide values for the action's fields:



A custom action can be performed only on an existing row (not on a pending Create row).

Once you perform an action on the form, it takes effect immediately, unlike an action in a table (where rows are marked for actions and the actions are performed later during an upload).

Also, form row data is not automatically refreshed after a custom action is performed, even if the action was successful. Click **Download Data** if you want to refresh data in the form row.

Manage Data for Layouts in a Dependent Hierarchy

Managing data in a layout that's part of a hierarchy of dependent layouts is similar to any download, upload, or clear operation in a Table or a Form-over-Table layout, except that the operation works on all layouts in the hierarchy.

When you click **Download Data**, **Upload Changes**, or **Clear** 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. When the operation is complete on all layouts in the hierarchy, the worksheet with the primary layout is made active again.

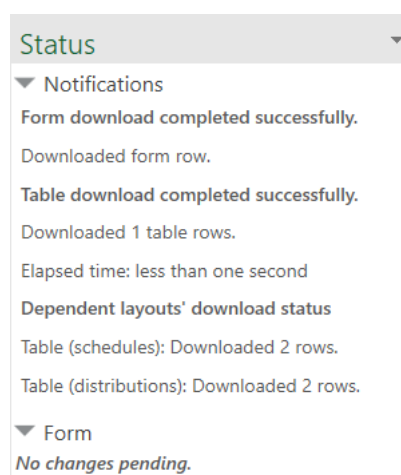
During a download operation, all items for all rows from the parent layout are downloaded at each level. (Any search specifications, if configured, apply only to the primary Form-over-Table layout.) For example, when Sheet 1 in your workbook contains Purchase Orders as the parent and Lines as the child (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 2 Schedules, the Schedules table would download 20 Lines. With this download, you can update all Schedules and upload all the changes together.

Following a download, you can edit data much as you would a Table or a Form-over-Table layout. Remember though that when you create rows in the grandchild table, you need to identify the correct child row that it belongs to. Typically, one column in the grandchild table comes from the child table. For example, let's say the Schedules table in Sheet 2 includes a PO Line ID column that comes from the Lines table in Sheet 1. By typing the correct PO Line ID in the Schedules table for new create rows, the schedules will be properly associated with the PO Lines.

When updates are ready to be uploaded, the upload operation submits all pending changes across the hierarchy of layouts.

You can view details of the operation in the primary Form-over-Table layout's Status Viewer, which 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:



Click each dependent layout to view additional details of the operation.

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.

15

REST Service Support

This chapter provides additional technical details about how the Oracle Visual Builder Add-in for Excel supports integration with REST services. It also provides information about technical known issues and limitations.

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 an 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 (see [Create Layouts in an Excel Workbook](#)). 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](#)).

Service Types

The add-in provides special support for:

- Oracle business object REST API services
 - Uses and requires REST API framework version 6
 - Provides search editor capabilities
 - Supports the ability to upload in batches
 - Supports finders
- Oracle REST Data Services (ORDS)
 - Provides search editor capabilities

See [Oracle REST Data Services](#).

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 Oracle Visual Builder Add-in for Excel 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 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>

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. 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.

- 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.
- Multi-part primary keys are supported only for Oracle business object REST API services.
- 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 will need to re-import the service description and create a new layout. Or, if the change is minor, you may choose to update the business object details to match the change in the service. See [Edit Service Descriptions and Business Objects](#).

16

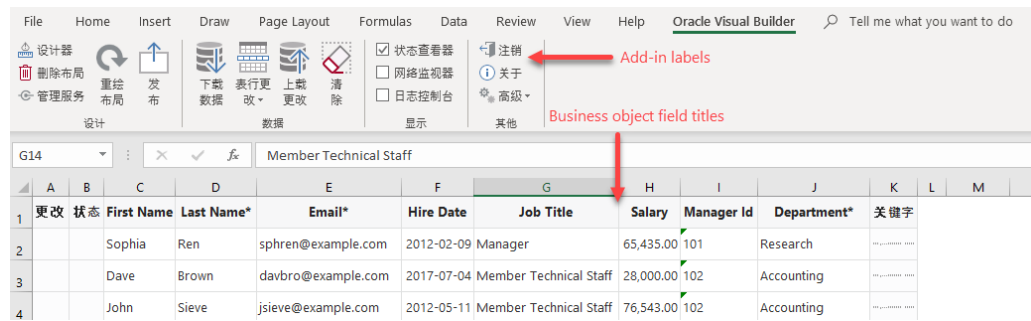
Internationalization

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

The date, date-time, and number formats used by the add-in are culture-sensitive (see also [Appearance of an Integrated Excel Workbook](#)).

When using the add-in, you work with:

- Labels visible on the Oracle Visual Builder ribbon and in various windows displayed by the add-in. These labels, known as the **add-in labels**, are owned by the add-in and are localized.
- Labels visible as column headers and field labels are known as **business object field titles**. These titles are owned by the REST service.



The add-in sends the `accept-language` header to the service on every request. The language setting specified for Excel is used for the add-in labels and for requests to the service, including the describe requests that fetches the initial business object field titles.

Note:

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

Change the Add-in's Language

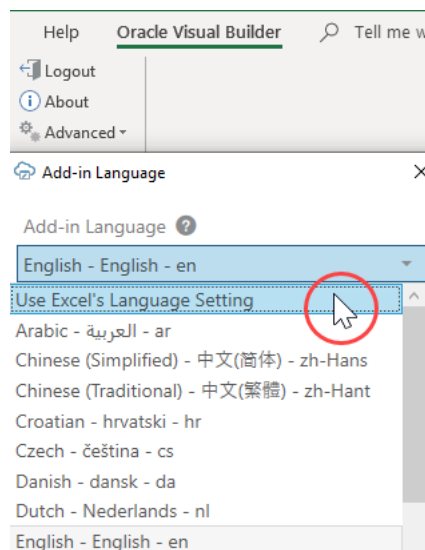
You can change the language that the Excel add-in uses. Do this if you want to evaluate your integrated workbook with different languages.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Choose **Select Language** from the Advanced drop-down.
3. In the Add-in Language drop-down list that appears, select the language you want to use. The drop-down list displays the languages that the add-in supports.

4. Click **OK**.
5. 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.

17

Security Best Practices

When using the 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.
- Also, consider using Excel's Inspect Workbook feature to review and remove your personal information from the workbook before you distribute it. Do this for unpublished workbooks. You access the Inspect Workbook feature from Excel's File menu. Clear the check box next to the Hidden Worksheets entry in the Document Inspector where you choose the content to inspect and potentially remove. You must not remove hidden worksheets from the Excel workbook that you distribute. The add-in uses hidden worksheets to integrate the Excel 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.

Troubleshoot Excel Workbooks

The 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.

In addition, consider running the Client Health Check tool to determine if your environment is configured correctly after you install the add-in. You download this tool from My Oracle Support. See [How to use Visual Builder Add-in for Excel - Client Health Check Tool \(Doc ID 2477792.1\)](#)

Additional information about troubleshooting can be found in [Troubleshooting Guide for the Visual Builder Add-in for Excel \(Doc ID 2485062.1\)](#).

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 the issues.

1. In Excel, click the **Oracle Visual Builder** tab.
2. Select **Log Console**.
The Logging Console window displays.
3. Repeat the steps that lead to the issue.
4. Review the logged messages.
5. Optionally, select **Set Level** and select a value from the drop-down list to specify the log level.

By default, the log level is set to Off. The change to the log level is temporary. The log level resets when you exit Excel.

 **Tip:**

Try the Information log level initially.

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.

19

Migration

You can migrate an Excel workbook integrated with a REST service using version 1.x of the Oracle Visual Builder Add-in for Excel to use the current version of the add-in. Once you migrate an Excel workbook to the current version, you can no longer use it with version 1.x.

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** in the Oracle Visual Builder tab 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 2.0 or later of the add-in

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

Note:

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

Third Party License

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

NewtonSoft.Json, Version 12.0.3

The MIT License (MIT)

Copyright (c) 2007 James Newton-King

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Microsoft.OpenApi, Version 1.2.0

Copyright (c) Microsoft Corporation. All rights reserved.

MIT License

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED *AS IS*, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

© 2019 GitHub, Inc.

SharpYaml, Version 1.6.5

Copyright (c) 2013-2016 SharpYaml - Alexandre Mutel

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SharpYaml is a fork of YamlDotNet <https://github.com/aaubry/YamlDotNet> published with the following license:

Copyright (c) 2008, 2009, 2010, 2011, 2012 Antoine Aubry

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.