# Translations Builder User Guide

Translations Builder is an external tool created for Power BI Desktop to assist dataset authors and report authors with tasks associated with creating translations and building multi-language reports. As a user, you can install Translations Builder and use it together with Power BI Desktop to build and test datasets and reports that support multiple languages.

While this document has been designed to explain the features and limitations of Translations Builder, it is recommended that you also read through the [**guidance document**](https://github.com/PowerBiDevCamp/TranslationsBuilder/blob/main/Docs/Building%20Multi-language%20Reports%20in%20Power%20BI.md) and that you work through the [**hands-on lab exercises**](https://github.com/PowerBiDevCamp/TranslationsBuilder/blob/main/Labs/Hands-on%20Lab%20-%20Building%20Multi-language%20Reports%20for%20Power%20BI.md). These learning resources will complement the contents you’ll find here.

There are three types of translations that come into play when localizing Power BI datasets and building reports that support multiple languages. Translations Builder helps to create and manage the first two types of translations which are **metadata translations** and **report label translations**. Translations Builder does not provide assistance with implementing **data translations**. For more conceptual background on translations used with Power BI, please read [**Understanding the Three Types of Translations**](https://github.com/PowerBiDevCamp/TranslationsBuilder/blob/main/Docs/Building%20Multi-language%20Reports%20in%20Power%20BI.md#understanding-the-three-types-of-translations).

### Translations Builder Limitations

To use Translations Builder effectively, you should be aware of its scope and understand a few important limitations.

#### Supported Dataset Editing Scenarios

Translations Builder has been designed to work with Power BI datasets running locally in Power BI Desktop. Translations Builder does not support connecting to Power BI datasets in the Power BI Service or connecting to older dataset formats used by Analysis Services. It’s possible to move beyond these limitations by extending Translations Builder as a developer. See the [**Translations Builders Developer Guide**](https://github.com/PowerBiDevCamp/TranslationsBuilder/blob/main/Docs/Developer%20Guide.md) for more information. Translations Builder is similar to Power BI Desktop in that it can only be installed on the Windows operating system.

#### Saving Your Changes in Power BI Desktop

While Translations Builder is designed to directly update datasets loaded into memory by Power BI Desktop, it is not able to persist changes to a dataset to the underlying PBIX project files on its own. Therefore, it is important to return to Power BI Desktop and save your changes after adding and editing translations with Translations Builder.

#### Supported Languages and Locales

When the Power BI Service loads a report, it creates a user context that includes a **language ID** and a **locale identifier**. The Power BI Service parses the language ID and the locale identifier together into string-based value known as the **culture identifier**. For example, a culture identifier of **en-US** represents a user who speaks English (**en**) in the United States (**US**). A culture identifier of **fr-FR** represents a user who speaks French (**fr**) in the France (**FR**).

Translations Builder supports the following set of languages, each with a specific culture identifier.

|  |  |  |  |
| --- | --- | --- | --- |
| Afrikaans [af-ZA] | Filipino [fil-PH] | Italian [it-IT] | Serbian [sr-Latn-BA] |
| Arabic [ar-001] | **Finnish [fi-FI]** | **Japanese [ja-JP]** | **Slovak [sk-SK]** |
| Bulgarian [bg-BG] | **French [fr-FR]** | **Korean [ko-KR]** | **Slovenian [sl-SI]** |
| Catalan [ca-ES] | **German [de-DE]** | **Latvian [lv-LV]** | **Somalian [so-SO]** |
| Chinese [zh-CN] | **Greek [el-GR]** | **Napali [ne-NP]** | **Spanish [es-ES]** |
| Croatian [hr-HR] | **Hebrew [he-IL]** | **Norwegian [nb-NO]** | **Swedish [sv-SE]** |
| Czech [cs-CZ] | **Hindi [hi-IN]** | **Persian [fa-IR]** | **Thai [th-TH]** |
| Danish [da-DK] | **Hungarian [hu-HU]** | **Polish [pl-PL]** | **Turkish [tr-TR]** |
| Dutch [nl-NL] | **Icelandic [is-IS]** | **Portuguese [pt-PT]** | **Ukrainian [uk-UA]** |
| English [en-US] | **Indonesian [id-ID]** | **Romanian [ro-RO]** | **Vietnamese [vi-VN]** |
| Estonian [et-EE] | **Irish [ga-IE]** | **Russian [ru-RU]** |  |

While Translations Builder supports all the languages shown above, it is important to understand that it only supports a single culture identifier per language. For example, you can add the language **Spanish [es-ES]** to your PBIX project. But you cannot add the language Spanish with the different culture identifier such as **es-MX** for Spanish in Mexico.

Even if you intend to build multi-language reports for users who speak Spanish in Mexico, you are required to add the language as **Spanish [es-ES]**. Keep in mind that the multi-language reports you build can still load with a cultural identifier of **es-MX**. The DAX code generated by Translations Builder for implementing report label translation only uses the first part of the culture identifier so it would not make a different whether the report is loaded with a culture identifier of **es-ES** or **es-MX**.

For any project that requires English, you must add the language as **English [en-US]**. For any project that requires French, you must add the language as **French [fr-FR]** and so on. You can inspect list of languages shown above to see which culture identifier is used for each of the supported languages.

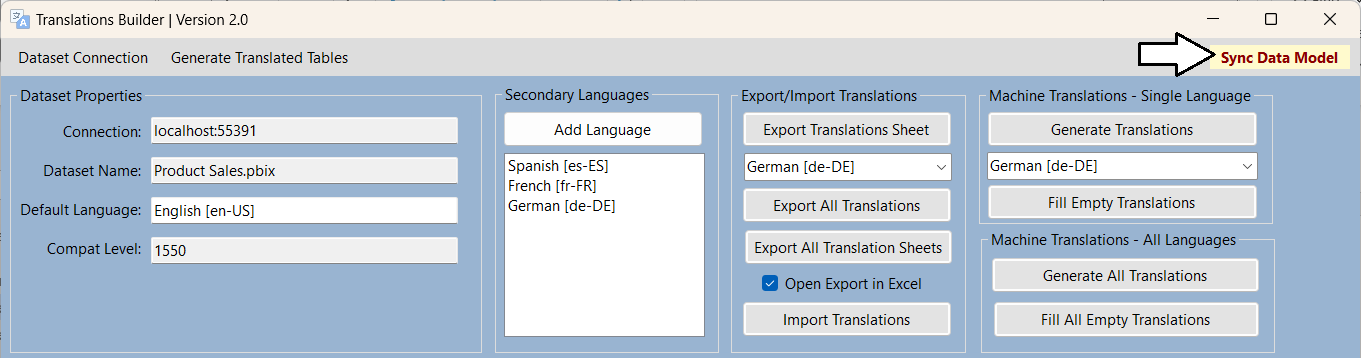
#### Import and Export using CSV Formatted Files

Translations Builder 2.0 uses the CSV file format to import and export translations sheets. Due to CSV files using comma separated values, Translations Builder does not support adding commas (**,**) to translations as they will cause errors when generating or reading translations sheets during import and export operations.

In future versions of Translations Builder, we are considering adding support for other file formats for import/export operations such as the RESX file format and a JSON-based file format.

#### Syncing Translations Builder with Power BI Desktop

Translation Builder reads



### Translations Builder Configuration Options

Before using Translations Builder, it is recommended you configure a few settings that are tracked on a per user basis. You can start by clicking the **Configure Settings…** from the **Dataset Connection** menu to display the **Configuration Options** dialog.

Graphical user interface, application, Word

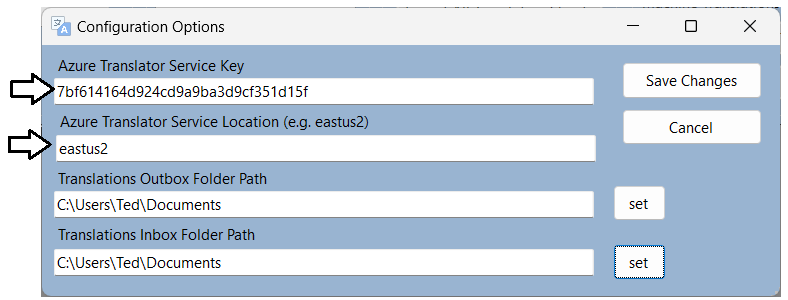
Description automatically generated

The **Configuration Options** dialog should appear like the one shown in the following screenshot.

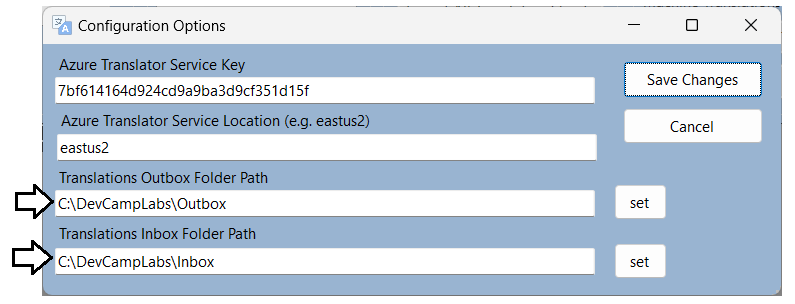
Table

Description automatically generated with low confidence

To enable support for generating machine translations with the Azure Translator Service, you must supply configuration values for the Azure Translator Service Key and Azure Translator Service Location. If you have an Azure subscription, you can learn how to obtain this key and its location by reading [Obtaining a Key for the Azure Translator Service](https://github.com/PowerBiDevCamp/TranslationsBuilder/blob/main/Docs/Obtaining%20a%20Key%20for%20the%20Azure%20Translator%20Service.md).



The **Configuration Options** dialog also allows you to assign local folder paths for the **Outbox** and **Inbox**. The **Outbox** folder is where Translation Builder generates files for export operations. The **Inbox** folder is where Translations Builder looks for files when you want to perform an import operation. By default, both the **Outbox** and **Inbox** are configured to use the Documents folder for the current user. However,

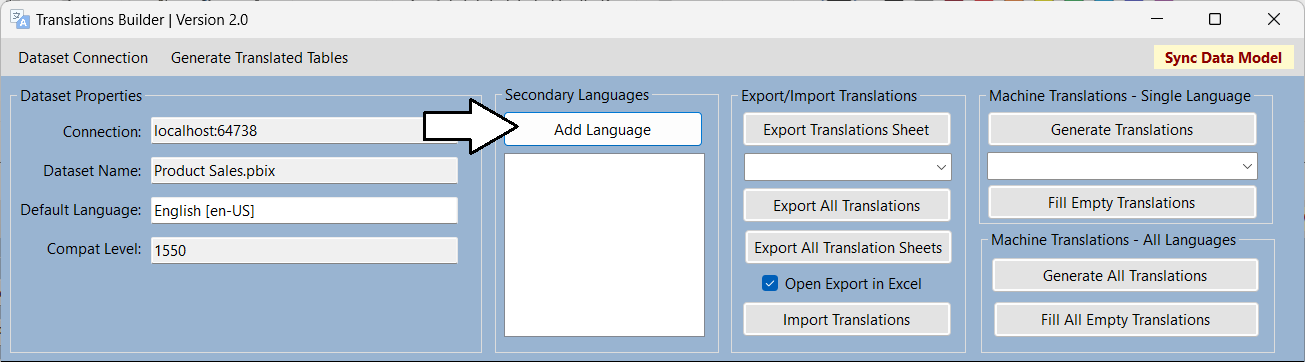


### Adding Metadata Translations

In general, you start any PBIX project involving translations by adding one or more secondary languages. Once you have added the languages you require, then you can begin to add translations.

#### Adding Secondary Languages

You will use the **Add Language** button to add new languages.

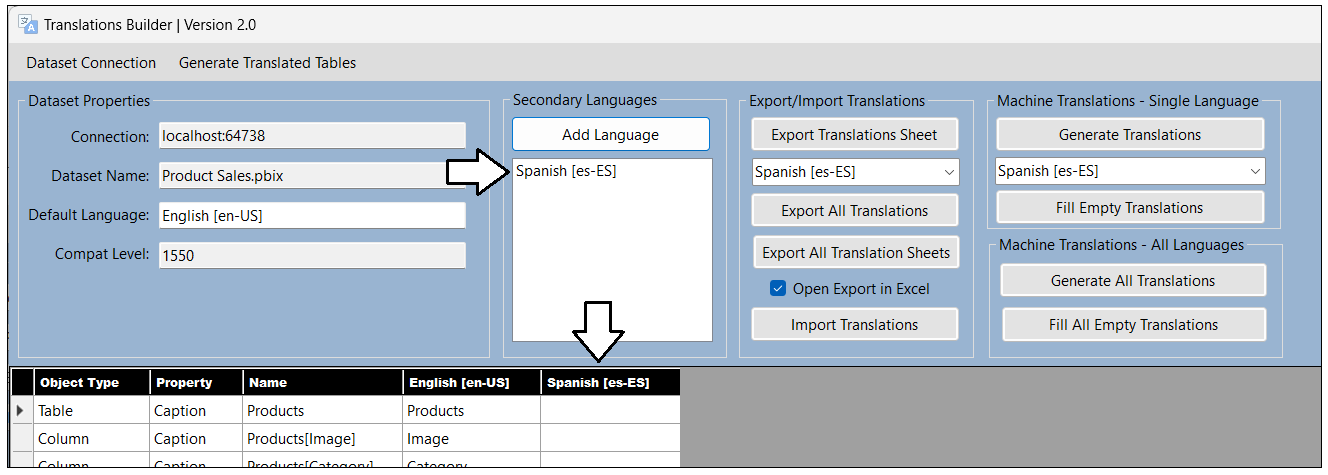


Click the **Add Languages** button will open the **Add Language** dialog. In this dialog you can select a language and then click the **Add Language** button.

Graphical user interface

Description automatically generated

Once the new language has been added, it should be displayed in the list box just underneath the **Add Languages** button in the **Secondary Languages** section. The new language should also get its own new column in the translation grid.



Note the **Add Language** dialog supports multiple selection if you’d like to add more than one language in one operation.

Graphical user interface, application

Description automatically generated

If you’d like to remove a language from your PBIX project, you can do so by right-clicking the column header for that language in the translation grid and clicking the **Delete Secondary Language** menu command.

Graphical user interface, text

Description automatically generated

When deleting a language, you’ll be prompted to confirm the operation as it deletes all the translations for that language.

Graphical user interface, application

Description automatically generated

#### Creating and Editing Translations by Hand

The key value proposition of Translations Builder is that is allows a content creator to view, add and update metadata translations using a two-dimensional grid. This ***translations grid*** simplifies the user experience because it abstracts aways the low-level details or reading and writing metadata translation associated with a dataset definition. Users work with the translation grid to view, add and edit metadata translations in a manner that is similar to working with data inside an Excel spreadsheet.

You can start by selecting a cell in a secondary language column in the translation grid. Selecting a cell will turn it blue.

Table

Description automatically generated

Once the cell for a secondary language translation is selected, you can begin typing.

Table

Description automatically generated

Once you’ve typed the translation, pressing ENTER will move the cell out of edit mode and move down to the cell below.

Table

Description automatically generated

Note you can even use the **{F2}** key to toggle a cell with content in and out of edit mode.

It’s easy to forget to save your changes in Power BI Desktop. Be aware that any changes made by Translations Builder are just made to the data model loaded in memory. None of your changes are saved back to the PBIX project file until you save in Power BI Desktop.

Graphical user interface, application

Description automatically generated

#### Testing Metadata Translations in the Power BI Service

One of the issues that makes working with translations a bit more complicated is that you cannot test your work with metadata translations in Power BI Desktop. Instead, you must test your work in the Power BI Service in a workspace associated with a Premium capacity. After you have added secondary languages and translations with Translations Builder and you have saved your changes to the underlying PBIX file, you can then publish the PBIX project from Power BI Desktop to the Power BI Service for testing.

Graphical user interface, application

Description automatically generated

Once you have published the PBIX project to the Power BI Service, you can test metadata translations for secondary languages by adding the **language** parameter with the culture identifier to the end of the report URL. For example, you can test your Spanish translations by adding a **language** parameter with a value of **es-ES**.

Graphical user interface, text

Description automatically generated

When you are testing metadata translations, you should not expect that all the text on a report page will be translated into a secondary language. That’s because you will not see translations that fall into the category of report label translations or data translations. You will only see metadata translations for dataset object names such as columns and measures. For example, columns headers in a table visual will display the metadata translations for column names and measure names.

Graphical user interface, application

Description automatically generated

You will see the translated measure names in Card visuals if they are configured to display the measure name.



You might also see your metadata translations used in visual titles and in legends.

Graphical user interface

Description automatically generated with low confidence

Now that you’ve seen how to test your work when working with translations, it possible to make a high-level observation about working with Translations Builder. As you begin to work with secondary languages and translations to localize a PBIX project, you will follow the same set of steps again and again:

1. Make changes in Power BI Desktop.
2. Publish the PBIX project to the Power BI Service.
3. Test your work with a browser in the Power BI Service using **language** parameter.
4. Repeat steps 1-3 until all the translations work has been completed.

#### Creating Machine Translations using the Azure Translation Service

Once you have configured the key and location for the Azure Translator Service in the **Configuration Options** dialog, the commands for generating machine translation will appear on the right side of the main window.

Graphical user interface, application

Description automatically generated

If you want to generate machine translations for a single language, you can start by selecting the target language in the dropdown menu under the **Generate Translations** button.

Graphical user interface, application

Description automatically generated

After selecting a language, clicking **Generate Translations** will start the process of generating machine translations.

Graphical user interface, application

Description automatically generated

While Translations Builder calls across the network to generate machine translations, it displays a progress dialog.

Graphical user interface, application

Description automatically generated

Once this dialog closes, you will see all cells of the target column have been filled with machine-generated translations.

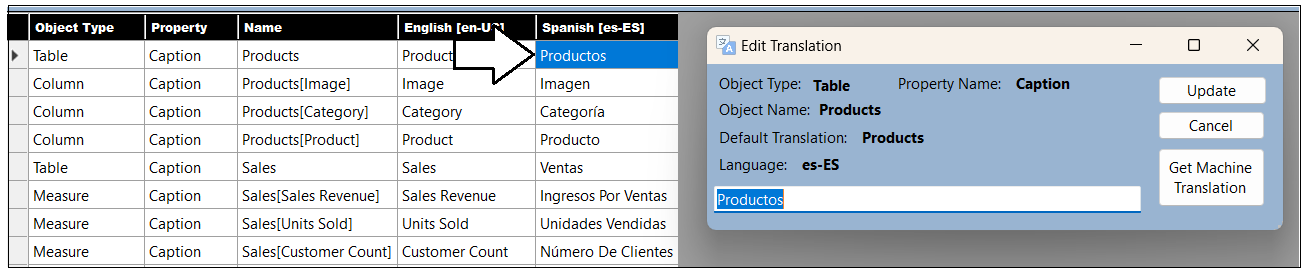
Graphical user interface, application

Description automatically generated

You have just seen that the **Generate Translations** command will create machine translations for a single secondary language. The **Generate All Translations** command below in the **Machine Translations - All Languages** section will create machine translations for all secondary languages at once one after another. Both of these commands will replace any existing translations with newly generated machine translations.

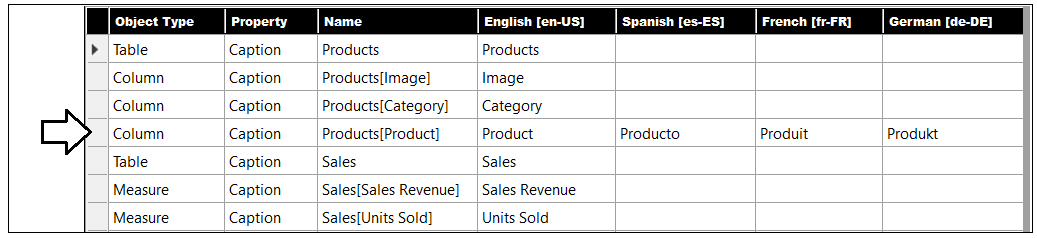
In some scenarios you might want to keep all the translations you’ve already added and just generate translations for any of the cells that are empty. You can accomplish this for one language at a time using the **Fill Empty Translations** command or all languages at once using the **Fill All Empty Translations** command.

As you have seen, you can select a cell and begin typing. Alternatively, you can double-click on a cell for a secondary language translation to display the **Edit Translation** dialog as shown in the following screenshot. This makes it possible to generate machine translations for a single dataset object at a time.



It can be a little tricky to display the **Edit Translation** dialog because the cell support both a click event and a double-click. If you are having trouble, select a different cell and then double-click the cell you want to edit.

It is also possible to generate machine translations for all secondary language columns in a single row.



### Adding Report Label Translations

Xxxx



xxx



Xx

Earlier you learned thatreport label translations provide localized values for text elements on a report that are not directly associated with a dataset object. Examples of report labels are the text values for report titles, section headings and button captions. Given that Power BI provides no built-in features to track or integrate report labels, Translations Builder solves this problem using the **Localized Labels** table strategy. Before introducing this strategy, let’s take a moment to discuss the problems this strategy has been designed to solve.

If you already have experience building datasets and reports with Power BI Desktop, it's critical that you learn which report design techniques to avoid when building multi-language reports. Let's begin with the obvious things which cause problems due to a lack of localization support.

* Using textboxes or buttons with hard-coded text values
* Adding a hard-coded text value for the title of a visual
* Displaying page tabs to the user

The key point here is that any hard-coded text value that gets added to the report layout cannot be localized. Consider the case where you add a column chart to your report. By default, a Cartesian visual such as a column chart is assigned a dynamic value to its **Title** property which is parsed together using the names of the columns and measures that have been added into the data roles such of **Axis**, **Legend** and **Values**.

Chart, bar chart

Description automatically generated

There is good news here. The default **Title** property for a Cartesian visual is dynamically parsed together in a fashion that supports localization. As long as you supply metadata translations for the names of columns and measures in the underlying dataset definition (e.g. **Sales Revenue**, **Country** and **Year**), the **Title** property of the visual will use the translations for whatever language has been used to load the report. The following table shows how the default **Title** property of this visual is updated for each of the these five languages.

|  |  |
| --- | --- |
| Language | Visual Title |
| English (en-US) | Sales Revenue by Country and Year |
| Spanish (es-ES) | Ingresos por ventas por país y año |
| French (fr-FR) | Chiffre d’affaires par pays et année |
| German (de-DE) | Umsatz nach Land und Jahr |
| Dutch (nl-NL) | Omzet per land en jaar |

Even if you dislike the dynamically-generated visual **Title**, you must resist the temptation to replace it with a hard-coded text value. Any hard-coded text you type into the **Title** property of the visual will be added to the report layout and cannot be localized. Therefore, you should either leave the visual **Title** property with its default value or you should use the **Localized Labels** table strategy to create report labels that support localization.

#### Creating the Localized Labels Table

Move back to Translations Builder and drop down the **Generate Translated Tables** menu.

Select the **Create Localized Labels Table** to create the Localized Labels table.

Graphical user interface, text, application

Description automatically generated

Create simple label

Show behind the scenes what has been created

Delete all and add multiple labels at once

Import labels from a file

#### Generating the Translated Localized Labels Table

Show command.

Show the two tables behind the scenes.

While the Localized Labels table is hidden from report authors, the Translated Localized Labels table is not hidden. That is what report authors use to create translated report labels

#### Surfacing Localized Labels on a Power BI Report

Measure make life easy. Over the last two years, Power BI Desktop has been extended

* Card visual
* Shape such as a Rectangle
* Button

### Adding Data Translations

When implementing metadata translations and report label translations, Translations Builder can automate a large percentage of the translations work that need to be done. Unfortunately, the same is not true for data translations. Implementing data translations will often require refactoring the underlying database or datasource to provide extra columns to provide translations on a row to row basis.

#### Understanding Translations Builder Support for Data Translations