# Building Multi-language Reports in Power BI

Power BI provides Internationalization and localization features which make it possible to build multi-language reports. For example, you can design a Power BI report that renders in English for some users while rendering in Spanish, French, German or Dutch for other users. If a company or organization has the requirement of building Power BI reports that support multiple languages, it's no longer necessary to clone and maintain a separate PBIX project file for each language. Instead, they can increase reuse and lower report maintenance by designing and implementing multi-language reports.

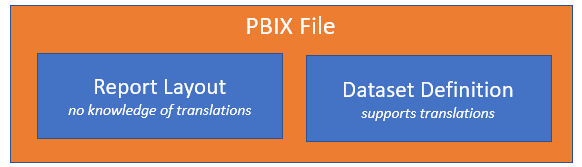
## Overview of Multi-language Report Design

Power BI provides the features required to design and implement multi-language reports. However, the path to success is not overly intuitive. The purpose of this article is to explain how to use the Power BI features for Internationalization and localization from the ground up and to provide the guidance for building reports that support multiple languages.

The primary feature in Power BI used to build multi-language reports is known as **translations**. Power BI inherited this feature from its predecessor, Analysis Services, which introduced translations to add localization support for the data model associated with a tabular database or a multidimensional database. In Power BI, translations support has been implemented at the dataset level.

A translation represents the property for an object in a data model that's been translated for a specific language. Consider a simple example. If your data model contains a table with an English name of **Products**, you can add translations for the **Caption** property of this table object to provide alternative names for when the report is rendered in a different language. The object types in a Power BI dataset that support translations include tables, columns, measure and hierarchies. In addition to the **Caption** properties which tracks an object's name, Power BI dataset objects also support adding translations for two other properties which are **Description** and **DisplayFolder**.

Keep in mind that the Power BI support for translations only applies to datasets. The Power BI report designer does not support adding translations. That means you cannot localize any text content you add to a report layout. For example, if you add to a textbox or a button a Power BI report and type in a literal text value, that text value cannot be localized. Therefore, you must avoid using textboxes and buttons with literal text when designing multi-language reports. Page tabs in a Power BI report are also problematic because their display names cannot be localized. Therefore, you must design multi-language reports so that page tabs are never displayed to the user.



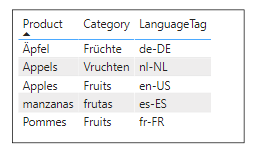
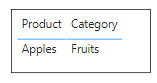
A little later in this article you learn about the low-level details of building multi-language reports in Power BI Desktop. At this point, however, it's possible to make a high-level observation. For someone experienced with report building in Power BI Desktop, the challenge of learning how to build multi-language reports isn't as much about learning ***what to do*** but more about learning ***what not to do***. There are lots of popular Power BI report design techniques that cannot be localized and therefore cannot be used when building multi-language reports.

Dataset translations are used to localize object properties such as the names of tables, columns and measures. Since these translations are created and maintained as part of the metadata for a dataset, they are also known as **metadata translations**. While metadata translation help to localize the names of tables and columns, they don't offer any assistance when it comes to localizing text values in the data itself. If your dataset has a **Products** table, how do you localize the product names in exists in individual **Products** table records?

While adding metadata translations to your dataset is an essential first step, it does not provide a complete solution by itself. A complete solution will not only localize the names of tables and columns, it will localize the content stored as text-based values in table rows such as product names. Therefore, the use of dataset translation to localize metadata must be complimented by a design which supports **content translation**.

Long before Microsoft introduced Power BI, software developers have been building multi-language applications that support content translation. After two decades of designing and refining various database designs to support content translation, several common design patterns have emerged as industry best practices. The design pattern for content translation that is best suited for Power BI dataset is row replication.

Consider a simple example of using the row replication design pattern to implement content translation. Let's say the **Products** table contains two text columns named **Product** and **Category** and you'd like your report to support five different languages including English, Spanish, French, German and Dutch. For each product in the **Products** table, you need to generate 5 records where each record contains the product name and category translated to a specific language. Whenever the report is loaded, a filter is applied to **LanguageTag** column users only see rows for one of the supported languages.



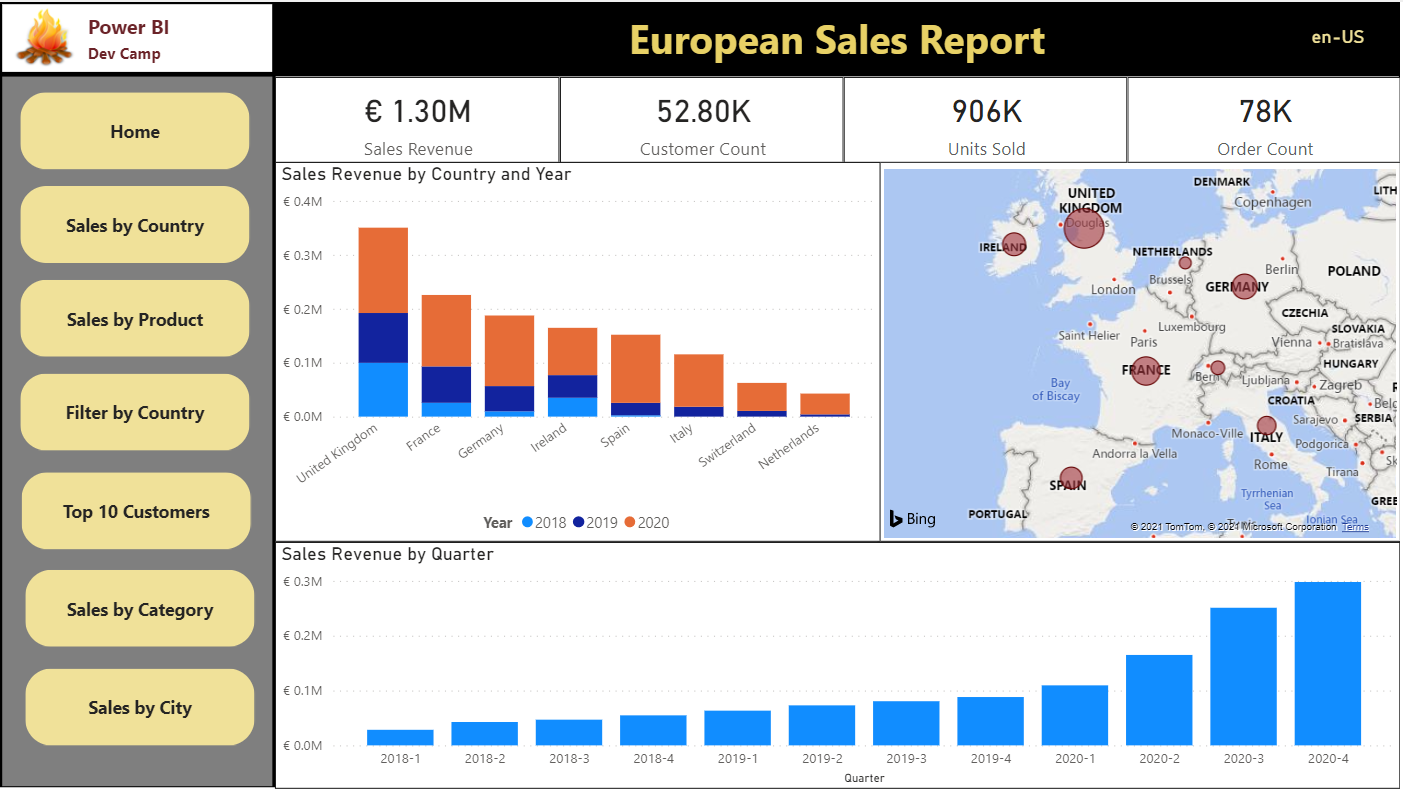
Now that you understand the fundamental concepts involved with building multi-language reports, it's time to discuss how to structure the multi-language PBIX development process. Here are the three phases that we will work through in this article to build a multi-language report.

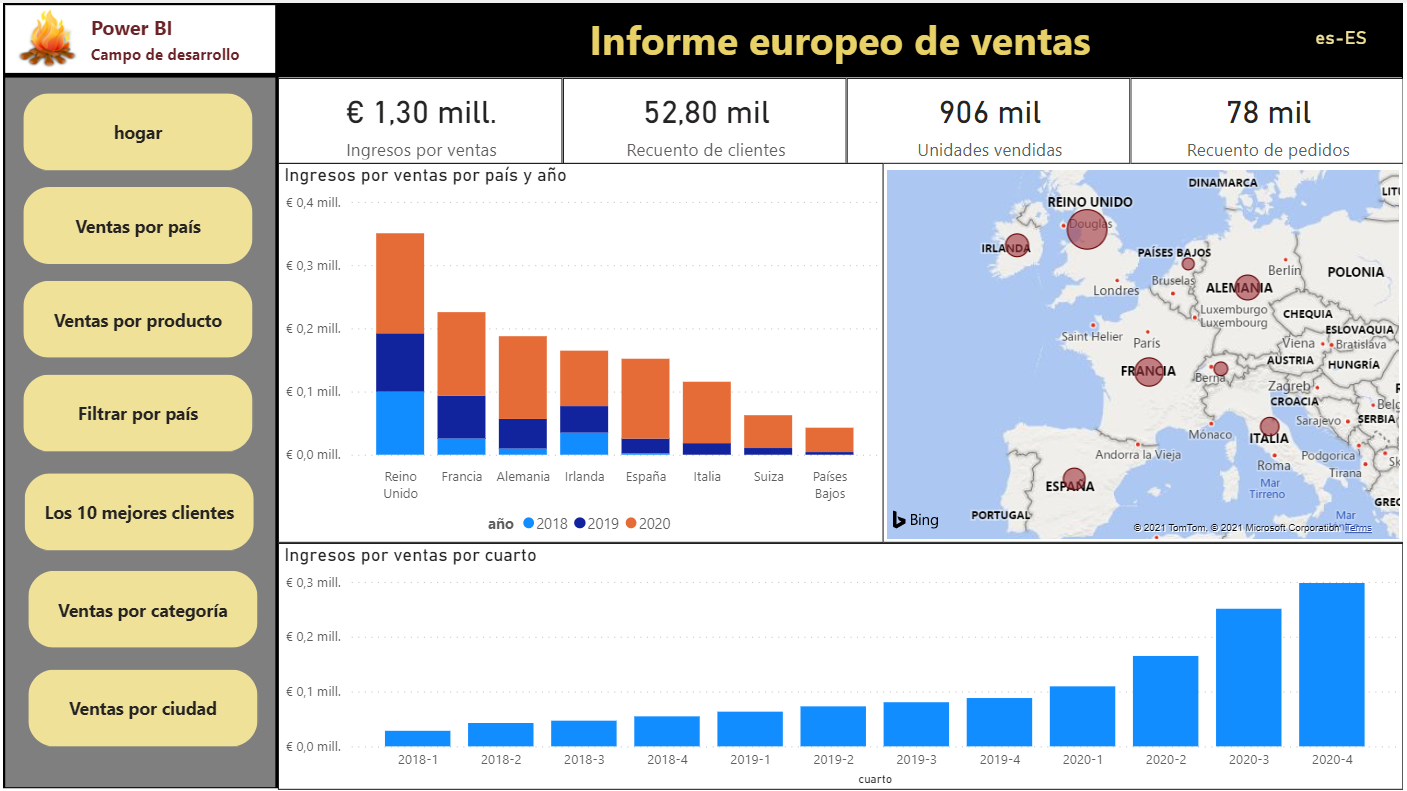
1. Use Power BI Desktop to prepare the PBIX project file to support translations
2. Add metadata translations to the PBIX project file
3. Design and implement content translation strategy

Developer sample

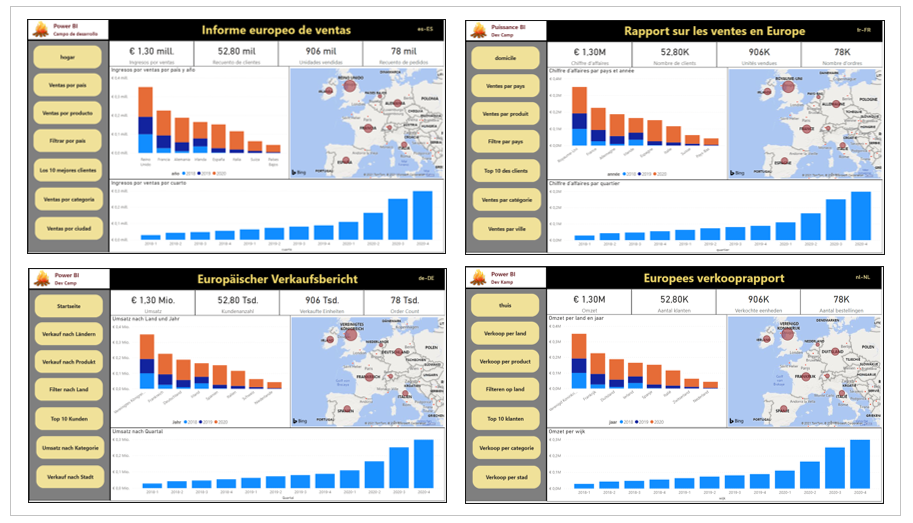
EuropenProductSales.pbix built as report designed to support Report English, Spanish, French, German and Dutch. Includes metadata transaltions and content translations.

Live demo at <https://multilanguagereportdemo.azurewebsites.net>



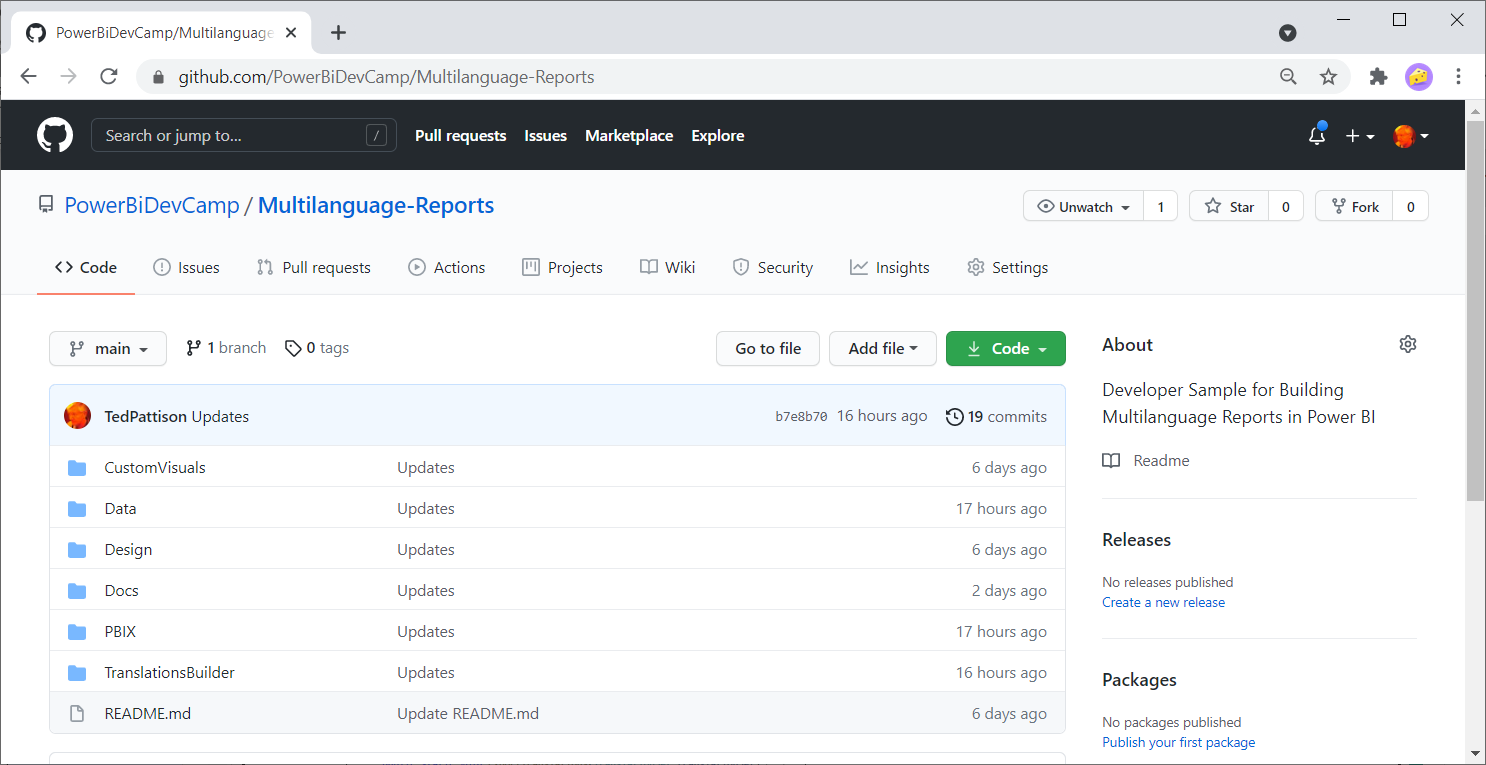


Xxx



All sample code for this developer samples available for download

<https://github.com/PowerBiDevCamp/Multilanguage-Reports>



## Designing Multi-language Reports

Preparing a PBIX Project for Translations

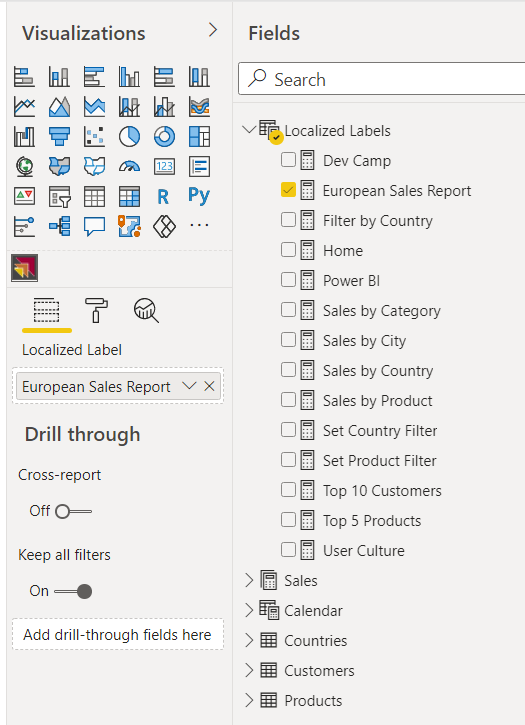
You might have a simple scenario where you are creating a PBIX project file that contains both a dataset and a report. This is the scenario discussed in this article and implemented in the developer sample. However, you can use the exact same strategies to build multi-language in more complicated scenarios where datasets and reports are defined in separate PBIX project files.

Plan localization from the start. Much harder to work with pre-existing PBIX created without localization in mind

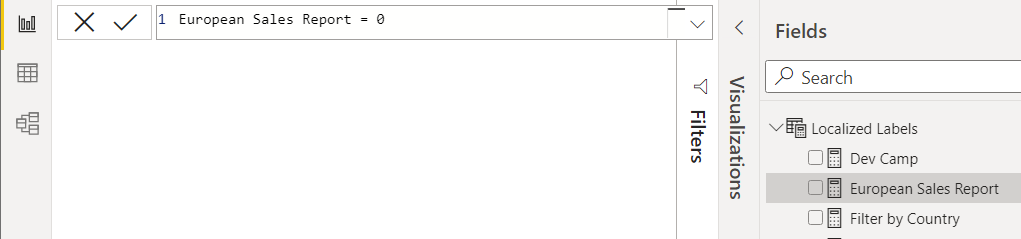
Plan for content growth. Some languages have content wider than English. Include padding for translated content

Avoid report design techniques that do not support localization. Don’t add literal text in visuals such a textboxes or button. Don’t display page tabs - their content cannot be localized.

Create a new table in data model named Localized Labels. Add a measure for any string content that needs to be localized

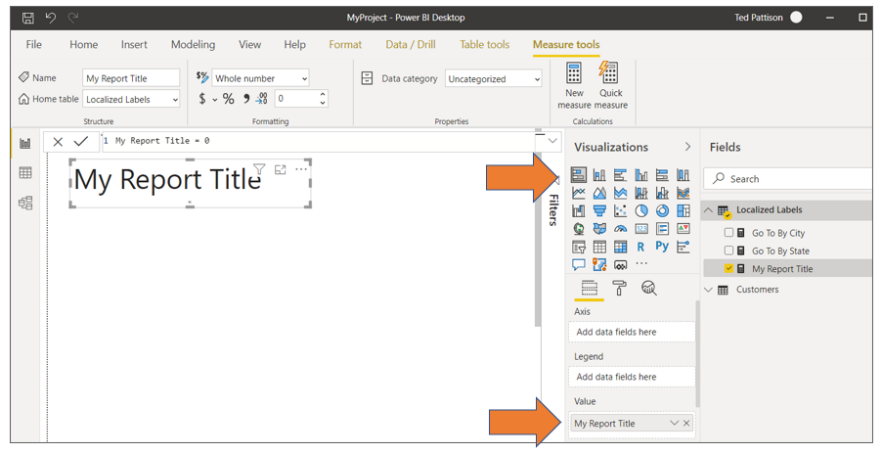


Set measure expressions to 0

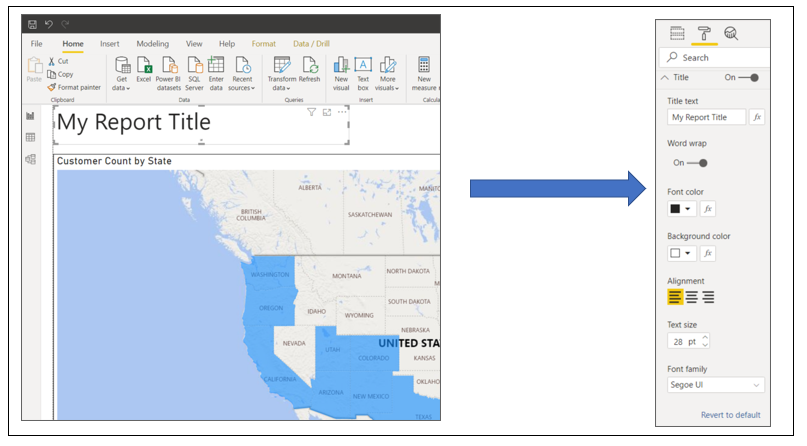


Surfacing a Localized Label.

You can surface localized label using one of core Power BI visuals. Add Stacked Barchart visual and add localized label in Values data roles,

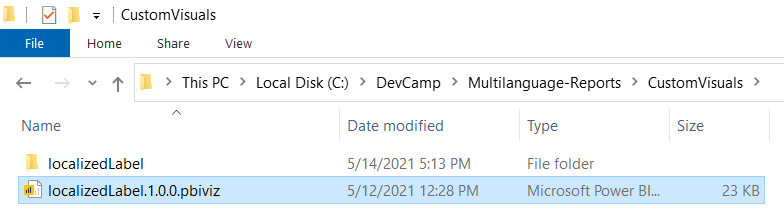


Using Format pane to configure label display properties. Design experience for this technique is limited

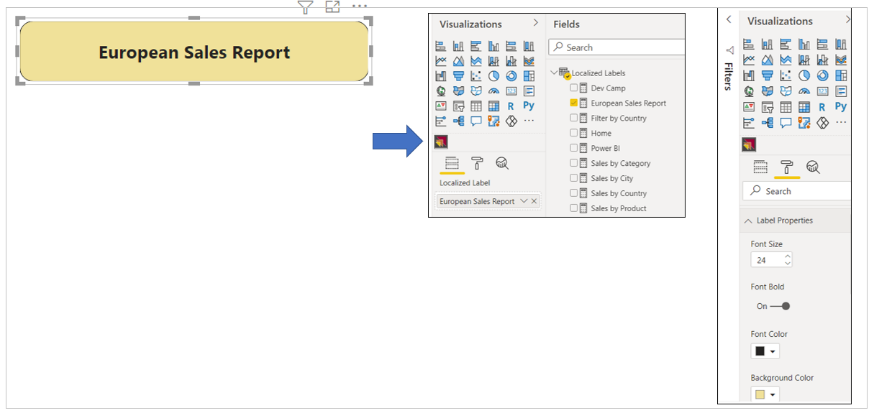


Developing a Custom Visual: LocalizedLabel

Using a custom visual to surface localized labels. Sample provides custom visual project

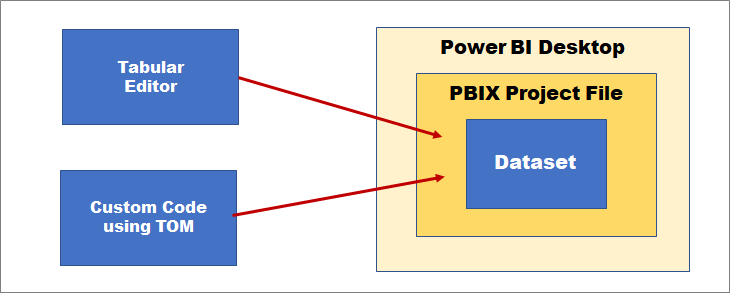


Provides more flexibility in configuring display properties



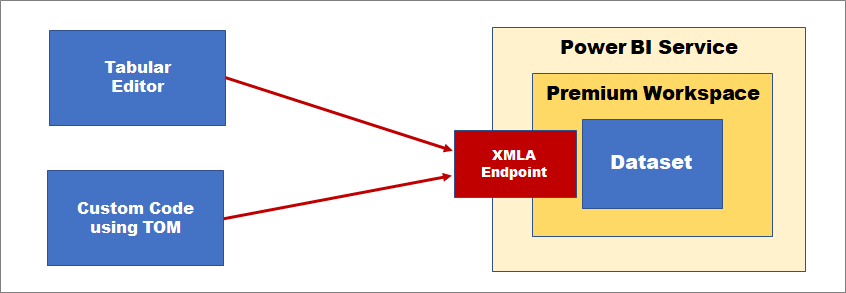
## Adding Metadata Translations to a Dataset

Once you have internationalized a PBIX project file and prepared it for localization, the next step is to add dataset translations for each language you need to support. You can accomplish this task by hand using an External Tool called the Tabular Editor. Alternatively, you can automate the task using an Power BI dataset API known as the Tabular Object Model (TOM). With either approach you can work on a PBIX project file that is open in Power BI Desktop.



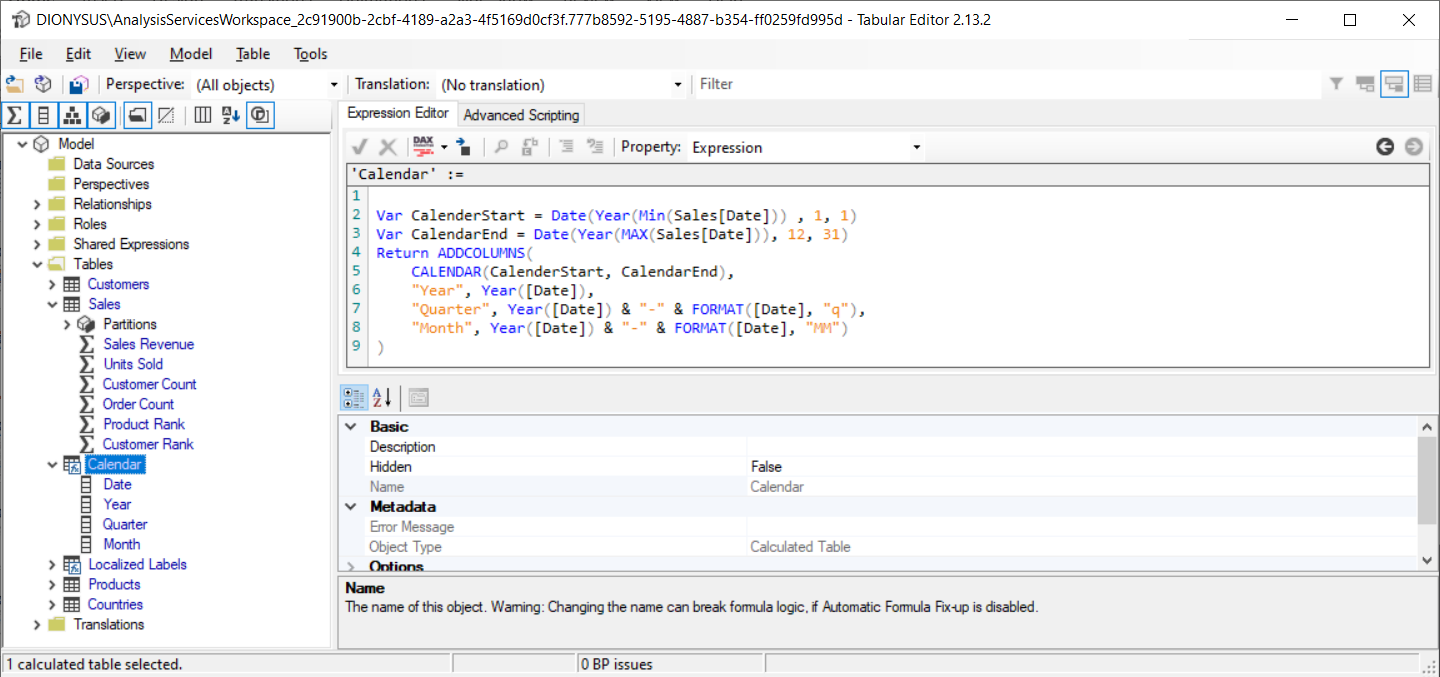
While Power BI Desktop doesn't have any explicit support to add translations to a dataset, it is still part of the process. Once you open a PBIX project file in Power BI Desktop, the dataset is loaded into memory and becomes accessible to both the Tabular Editor and to whatever custom code you write with TOM.

If often makes sense to add translations with a local setup involving Power BI Desktop as you build and ship a dataset with translations for the first time. Note that it is also possible to access datasets running in the Power BI Service using t eh exact same techniques. This screenshot show how you can continue to add and manage translations in a live dataset without needing to work with the PBIX project file.

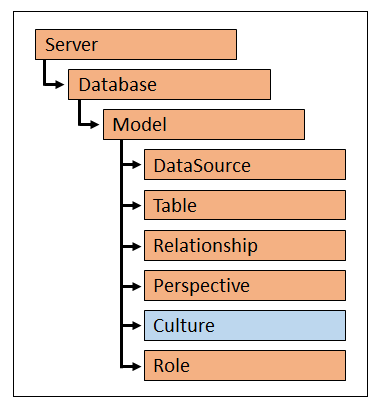


If you plan to work with dataset translations, you should become comfortable working with the Tabular Editor which can be downloaded from following URL: <https://github.com/otykier/TabularEditor/>.

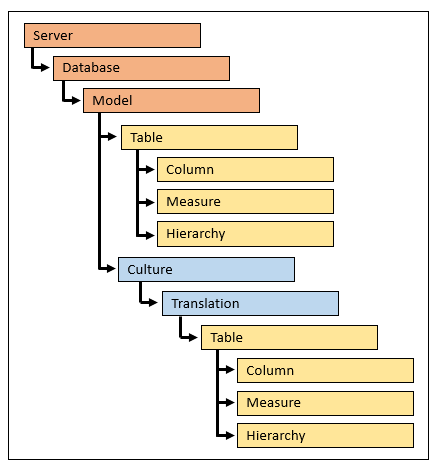
The Tabular Editor allows you to read and modify the metadata within a Power BI dataset.



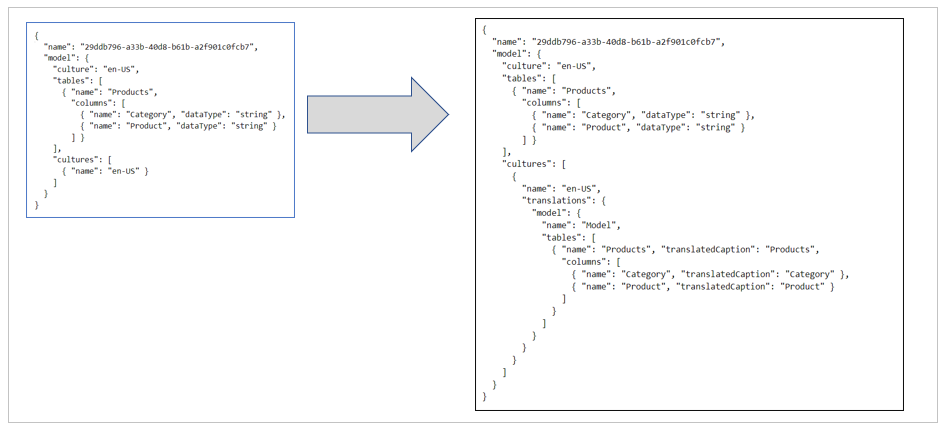
Creating models. importing and refreshing data, Assigning roles and permissions. Adding support for secondary cultures/languages



When you create a new PBIX project, it has a default culture (en-US) but contains no translations. You must add translations for each spoken language.



Aaa

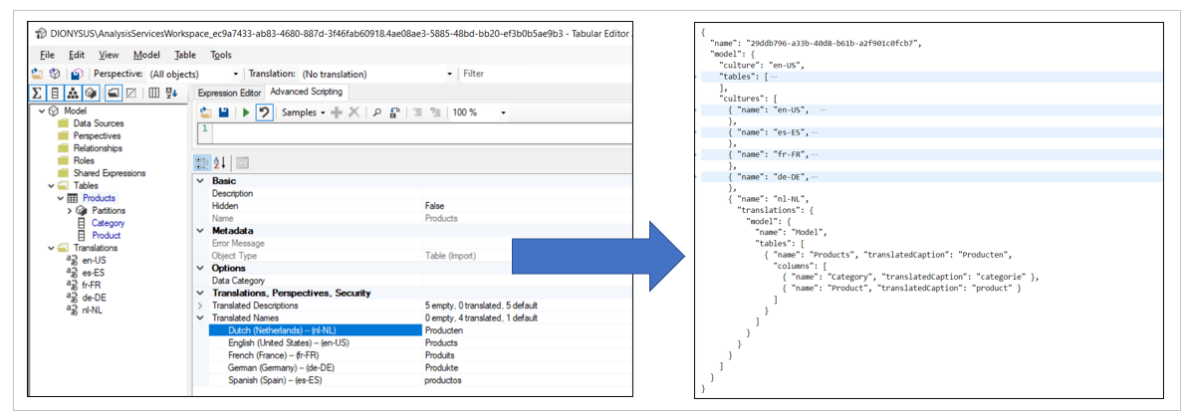


Translations must be added for each required language

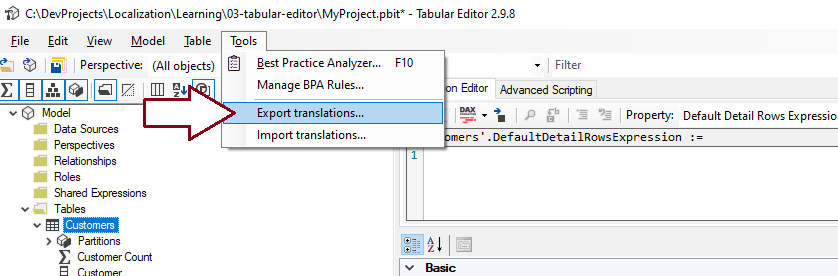
Contains translatedCaption for each translated string



Tabular Editor



Tabular Editor supports exporting/importing translations. Export metadata after populating translations for default language. Design workflow with human translators to add translations. Import updated translation back into PBIX project using Tabular Editor,



TOM programming

You can also automate the task of adding translations using the Tabular Object Model (TOM) which is an extension of Analysis Management Object (AMO) client library.

For each object, you can add translations for three properties which include **Caption**, **Description** and **DisplayFolder**.

You use the **Caption** property to add a translation for the name of an object such as a table, column, measure or hierarchy. If your data model uses display folders to organize columns and measures within tables, you need to add additional translation using the

## Embedding Reports with Specific Locales

## Designing Data Models to Support Content Translations

## Setting the Language for Current User using RLS and UserCulture

ss