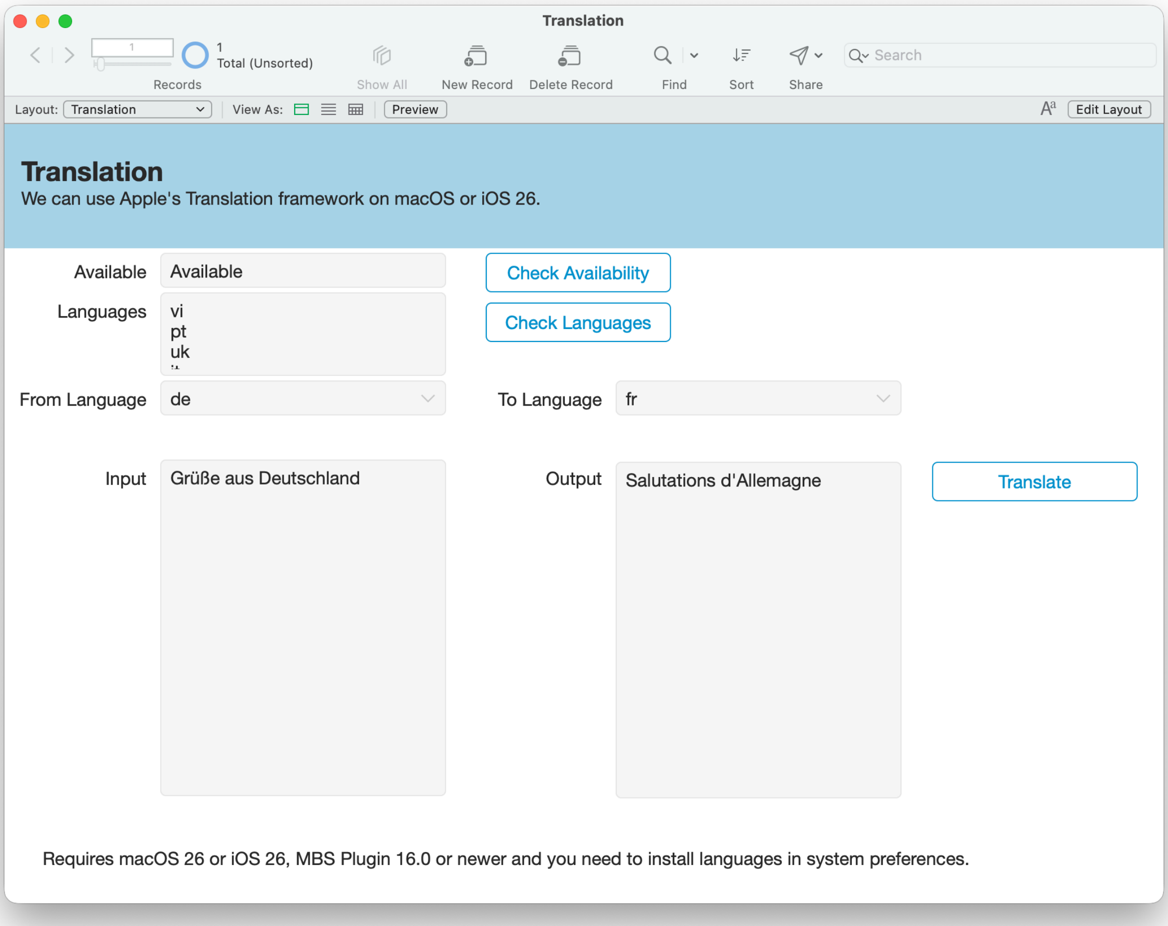
Translation Framework in FileMaker

Since macOS 26 (and iOS 26) you can use the Translate framework to perform on-device translation services. You can use this with the upcoming MBS FileMaker Plugin 16.0, currently in beta test. We also back ported it to macOS 15 and iOS 18 for most of the functionality.

Available?

First you may check whether the translation services are available. This basically checks only for whether you have macOS or iOS.

```
If [ MBS( "Translation.Available" ) = 1 ]
    Show Custom Dialog [ "Translation is available" ]
Else
    Show Custom Dialog [ "Translation is not available" ]
End If
```



Translation

We can use Apple's Translation framework on macOS or iOS 26.

Available	Available
Languages	vi pt uk
From Language	de
To Language	fr
Input	Grüße aus Deutschland
Output	Salutations d'Allemagne

Requires macOS 26 or iOS 26, MBS Plugin 16.0 or newer and you need to install languages in system preferences.

Supported Languages

Next you could ask for the list of supported language. Currently this list contains: vi, pt, uk, it, zh-TW, ko, en-GB, de, zh, ja, id, nl, fr, th, es, tr, pl, ar-AE, ru, en, and hi. This list may change with future updates as Apple adds more languages. To query this list, we have two variants. First you can call SupportedLanguages and pass a delegate to get the answer later when available. Or you call the synchronous variant where the plugin waits for it and returns the answer right away:

```
Set Variable [ $languages ; Value:
MBS( "Translation.SupportedLanguages") ]
```

Just because the language is supported, it may not yet be available or downloaded. For offline usage, the user needs to go to system preferences in the languages section and click download on a few language sets. You can call Status method with either the delegate for asynchronous answering or call our synchronous method to get the result with waiting. You pass the language identifiers and the framework will answer. The answer is a text with installed, supported, unsupported, unknown or an error. You get the value installed, if the local data is installed:

```
Set Variable [ $status ; Value: MBS( "Translation.Status") ]
```

Start a session

Once you decided for a language pair to translate, you can call the Constructor and start a new TranslationSessionMBS. If the second language is empty, the system will pick the best available language from the list. Like if an user in Norway has Norwegian and English in their languages list, they will get English as Norwegian is unavailable.

```
Set Variable [ $session ; Value: MBS( "Translation.StartSession"; "de";
"en") ]
If [ MBS( "Translation.IsReady"; $session ) ]
    Show Custom Dialog [ "We can translate German to English." ]
Else
    Show Custom Dialog [ "Please go to system preferences. You may
need to download language files." ]
End If
Set Variable [ $r ; Value: MBS( "Translation.Release"; $session) ]
```

The isReady function lets us check the status to make sure calling translate later doesn't raise an error. If ready, we can start translations.

Translations

Next you may want to translate. We have quick Translation.TranslateString function to return the translation directly like this:

```
Set Variable [ $session ; Value: MBS( "Translation.StartSession"; "de"; "en") ]
# translate something
Set Variable [ $translation ; Value: MBS( "Translation.TranslateString"; $session; "Hallo Leute" ) ]
# show to user
Show Custom Dialog [ $translation ]
# cleanup
Set Variable [ $r ; Value: MBS( "Translation.Release"; $session) ]
```

You can keep the session around and use it for multiple translations. If you do a lot of translations, you may better use the Translation.Translations function to pass in a JSON with a lot of texts.

```
Let([
    session = MBS( "Translation.StartSession"; "de"; "en");
    json = "[]";
    json = JSONSetElement ( json ; "[+].sourceText" ; "Hallo Leute" ; JSONString );
    json = JSONSetElement ( json ; "[+].sourceText" ; "Man sieht sich bei der Konferenz!" ; JSONString );
    text = MBS( "Translation.Translations"; session; json);
    r = MBS("Translation.Release"; session)
]; text)
```

When we run this, we receive back a JSON with the answers:

Example result:
```
[
  {
    "targetText" : "Hello people",
    "sourceLanguage" : "de",
    "targetLanguage" : "en",
    "sourceText" : "Hallo Leute"
  },
  {
    "sourceText" : "Man sieht sich bei der Konferenz!",
    "sourceLanguage" : "de",
    "targetText" : "See you at the conference!",
    "targetLanguage" : "en"
  }
]
```

Optional you can include clientIdentifier value in the request JSON and then this is returned back as identifier with the response. This way you can easily match which translation belongs to which item.

Please try the new functions with FileMaker on macOS 15/26 and iOS 18/26 and see what you can do with automatic translation.