Translate text with Azure Translator service

1. Introduction

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/1-introduction

Introduction

Completed

- 1 minute

There are many commonly used languages throughout the world, and the ability to exchange information between speakers of different languages is often a critical requirement for global solutions.

The Azure Translator provides an API for translating text between 90 supported languages.

After completing this module, you'll be able to:

- Provision an Azure Translator resource
- Understand language detection, translation, and transliteration
- Specify translation options
- Define and run custom translations

2. Provision an Azure Translator resource

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/2-provision-translator-resource

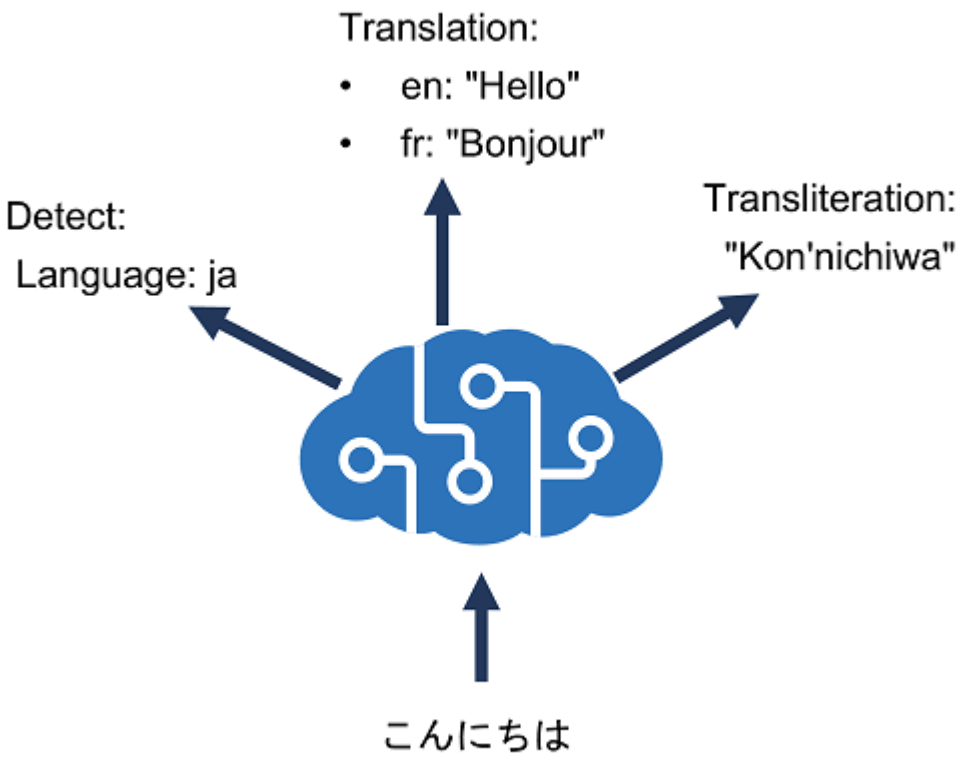
Provision an Azure Translator resource

Completed

- 2 minutes

Azure Translator provides a multilingual text translation API that you can use for:

- Language detection.
- One-to-many translation.
- Script transliteration (converting text from its native script to an alternative script).



Azure resource for Azure Translator

To use the Azure Translator service, you must provision a resource for it in your Azure subscription. You can provision a single-service Azure Translator resource, or you can use the Text Translation API in a Foundry Tools resource.

After you provision a suitable resource in your Azure subscription, you can use the **location** where you deployed the resource and one of its **subscription keys** to call the Azure Translator APIs from your code. You can call the APIs by submitting requests in JSON format to the REST interface, or by using any of the available programming language-specific SDKs.

> **Note**
>
> The code examples in the subsequent units in this module show the JSON requests and responses exchanged with the REST interface. When using an SDK, the JSON requests are abstracted by appropriate objects and methods that encapsulate the same data values. You'll get a chance to try the SDK for C# or Python for yourself in the exercise later in the module.

3. Understand language detection, translation, and transliteration

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/3-understand-language-detection-translation-transliteration

Understand language detection, translation, and transliteration

Completed

- 3 minutes

Let's explore the capabilities of **Azure Translator**. These capabilities include:

Language detection

You can use the **Detect** function of the REST API to detect the language in which text is written.

For example, you could submit the following text to the `https://api.cognitive.microsofttranslator.com/detect?api-version=3.0` endpoint using curl.

Here's the text we want to translate:

```
{ 'Text' : 'こんにちは' }
```

Here's a call using curl to the endpoint to detect the language of our text:

```
curl -X POST "https://api.cognitive.microsofttranslator.com/detect?api-version=3.0" -H "Ocp-Apim-S
```

The response to this request looks as follows, indicating that the text is written in Japanese:

```
[
  {
    "language": "ja",
    "score": 1.0,
    "isTranslationSupported": true,
    "isTransliterationSupported": true


   }
]
```

Translation

To translate text from one language to another, use the **Translate** function; specifying a single **from** parameter to indicate the source language, and one or more **to** parameters to specify the languages into which you want the text translated.

For example, you could submit the same JSON we previously used to detect the language, specifying a **from** parameter of **ja** (Japanese) and two **to** parameters with the values **en** (English) and **fr** (French). To do this, you'd call:

```
curl -X POST "https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=ja&to=f
```

This would produce the following result:

```
[
  {"translations":
    [
      {"text": "Hello", "to": "en"},
      {"text": "Bonjour", "to": "fr"}
    ]
  }
]
```

Transliteration

Our Japanese text is written using Hiragana script, so rather than translate it to a different language, you may want to transliterate it to a different script - for example to render the text in Latin script (as used by English language text).

To accomplish this, we can submit the Japanese text to the **Transliterate** function with a **fromScript** parameter of **Jpan** and a **toScript** parameter of **Latn**:

```
curl -X POST "https://api.cognitive.microsofttranslator.com/transliterate?api-version=3.0&fromScr
```

The response would give you the following result:

```
[
    {
        "script": "Latn",
        "text": "Kon'nichiwa"
    }
]
```

4. Specify translation options

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/4-specify-translation-options

Specify translation options

Completed

- 4 minutes

The **Translate** function of the API supports numerous parameters that affect the output.

Word alignment

In written English (using Latin script), spaces are used to separate words. However, in some other languages (and more specifically, scripts) this is not always the case.

For example, translating "Smart Services" from **en** (English) to **zh** (Simplified Chinese) produces the result "智能服务", and it's difficult to understand the relationship between the characters in the source text and the corresponding characters in the translation. To resolve this problem, you can specify the **includeAlignment** parameter with a value of **true** in your call to produce the following result:


```
[
  {
     "translations":[
        {
           "text":"智能服务",
           "to":"zh-Hans",
           "alignment":{
              "proj":"0:4-0:1 6:13-2:3"
           }
        }
     ]
  }
]
```


These results tell us that characters 0 to 4 in the source correspond to characters 0 to 1 in the translation, while characters 6 to 13 in the source correspond to characters 2 to 3 in the translation.

Sentence length

Sometimes it might be useful to know the length of a translation, for example to determine how best to display it in a user interface. You can get this information by setting the **includeSentenceLength** parameter to **true**.

For example, specifying this parameter when translating the English (**en**) text "Hello world" to French (**fr**) produces the following results:


```
[
  {
     "translations":[
        {
           "text":"Salut tout le monde",
           "to":"fr",
           "sentLen":{"srcSentLen":[12],"transSentLen":[20]}
        }
     ]
  }
]
```

Profanity filtering

Sometimes text contains profanities, which you might want to obscure or omit altogether in a translation. You can handle profanities by specifying the **profanityAction** parameter, which can have one of the following values:

- **NoAction**: Profanities are translated along with the rest of the text.
- **Deleted**: Profanities are omitted in the translation.
- **Marked**: Profanities are indicated using the technique indicated in the **profanityMarker** parameter (if supplied). The default value for this parameter is **Asterisk**, which replaces characters in profanities with "*". As an alternative, you can specify a **profanityMarker** value of **Tag**, which causes profanities to be enclosed in XML tags.

For example, translating the English (**en**) text "JSON is ██████ great!" (where the blocked out word is a profanity) to German (**de**) with a **profanityAction** of **Marked** and a **profanityMarker** of **Asterisk** produces the following result:

```
[
  {
     "translations":[
        {
           "text":"JSON ist *** erstaunlich.",
           "to":"de"
        }
     ]
  }
]
```

> **Note**
>
> To learn more about the translation options, including some not described here, see the Azure Translator API documentation.

5. Define custom translations

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/5-define-custom-translations

Define custom translations

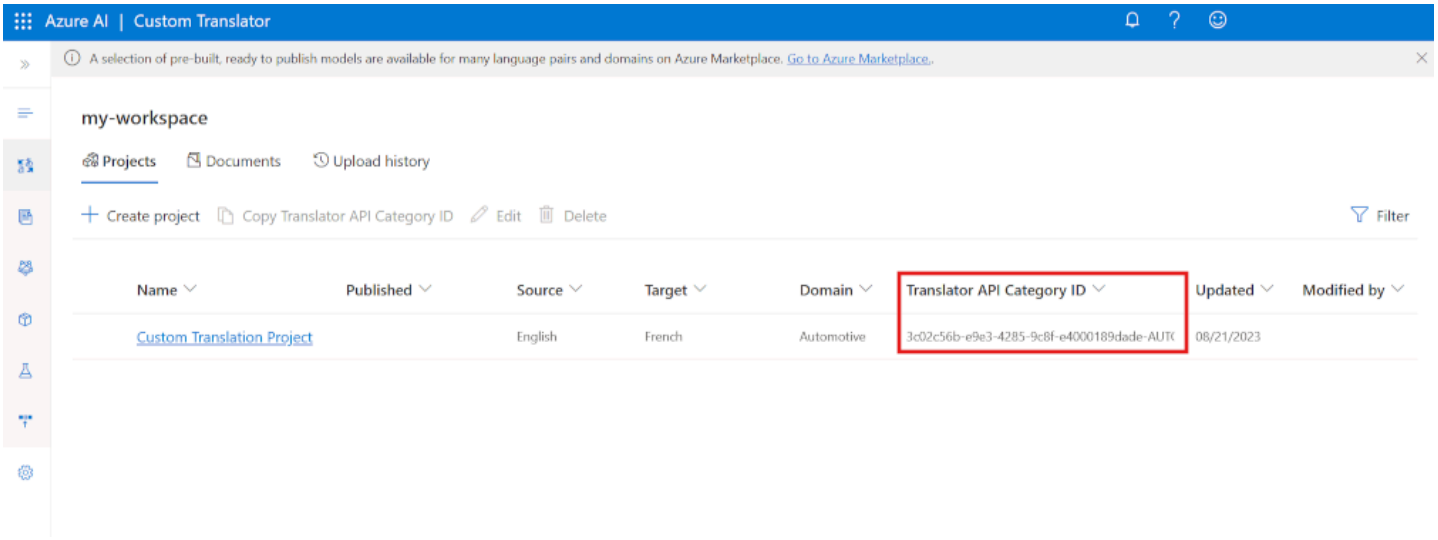
Completed

- 5 minutes

While the default translation model used by Azure Translator is effective for general translation, you may need to develop a translation solution for businesses or industries in that have specific vocabularies of terms that require custom translation.

To solve this problem, you can create a custom model that maps your own sets of source and target terms for translation. To create a custom model, use the Custom Translator portal to:

1. Create a workspace linked to your Azure Translator resource.
2. Create a project.
3. Upload training data files and train a model.
4. Test your model and publish your model.
5. Make translation calls to the API.



Your custom model is assigned a unique **category Id** (highlighted in the screenshot), which you can specify in **translate** calls to your Azure Translator resource by using the **category** parameter, causing translation to be performed by your custom model instead of the default model.

How to call the API

To initiate a translation, you send a **POST** request to the following request URL:

```
https://api.cognitive.microsofttranslator.com/translate?api-version=3.0
```

Your request needs to include a couple of parameters:

- `api-version`: The required version of the API.
- `to`: The target language to translate to. For example: `to=fr` for French.
- `category`: Your **category Id**.

Your request must also include a number of required headers:

- `Ocp-Apim-Subscription-Key`. Header for your client key. For example: `Ocp-Apim-Subscription-Key=<your-client-key>`.
- `Content-Type`. The content type of the payload. The required format is: `Content-Type: application/json; charset=UTF-8`.

The request body should contain an array that includes a JSON object with a `Text` property that specifies the text that you want to translate:

```
[
    {"Text":"Where can I find my employee details?"}
]
```

There are different ways you can send your request to the API, including using the C#, Python, and curl. For instance, to make a quick call, you can send a POST request using curl:

```
curl -X POST "https://api.cognitive.microsofttranslator.com/translate?api-version=3.0&from=en&to=
```

The request above makes a call to translate a sentence from English to Dutch.

Response returned

The response returns a response code of `200` if the request was successful. It also returns a response body that contains the translated text, like this:

```
[
    {
        "translations":[
            {"text":"Waar vind ik mijn personeelsgegevens?","to":"nl"}
        ]
    }
]
```

If the request wasn't successful, then a number of different status codes may be returned depending on the error type, such as `400` (missing or invalid query parameters). See Response status codes for a full list of codes and their explanation.

> **Note**

For more information about custom translation, see Quickstart: Build, publish, and translate with custom models.

6. Exercise - Translate text with the Azure Translator service

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/6-exercise-translate-text

Exercise - Translate text with the Azure Translator service

Completed

- 30 minutes

In this exercise, you build an app that translates text between languages.

Note

To complete this lab, you need an **Azure subscription**.

Launch the exercise and follow the instructions.

Launch Exercise

Tip

After completing the exercise, if you've finished exploring Foundry Tools, delete the Azure resources that you created during the exercise.

7. Module assessment

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/7-knowledge-check

Module assessment

Completed

- 2 minutes

8. Summary

https://learn.microsoft.com/en-us/training/modules/translate-text-with-translator-service/8-summary

Summary

Completed

- 1 minute

In this module, you learned how to:

- Provision an Azure Translator resource
- Understand language detection, translation, and transliteration
- Specify translation options
- Define custom translations

To learn more about Azure Translator, see the Azure Translator documentation.