# Translate Speech

Azure AI Speech includes a speech translation API that you can use to translate spoken language. For example, suppose you want to develop a translator application that people can use when traveling in places where they don't speak the local language. They would be able to say phrases such as "Where is the station?" or "I need to find a pharmacy" in their own language, and have it translate them to the local language. In this exercise, you'll use the Azure AI Speech SDK for Python to create a simple application based on this example.

While this exercise is based on Python, you can develop speech translation applications using multiple language-specific SDKs; including:

- Azure AI Speech SDK for Python
- Azure AI Speech SDK for .NET
- Azure AI Speech SDK for JavaScript

This exercise takes approximately **30** minutes.

> **NOTE** This exercise is designed to be completed in the Azure cloud shell, where direct access to your computer's sound hardware is not supported. The lab will therefore use audio files for speech input and output streams. The code to achieve the same results using a mic and speaker is provided for your reference.

## Create an Azure AI Speech resource

Let's start by creating an Azure AI Speech resource.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. In the top search field, search for **Speech service**. Select it from the list, then select **Create**.
3. Provision the resource using the following settings:
    - **Subscription**: *Your Azure subscription*.
    - **Resource group**: *Choose or create a resource group*.
    - **Region**:*Choose any available region*
    - **Name**: *Enter a unique name*.
    - **Pricing tier**: Select **F0** (*free*), or **S** (*standard*) if F is not available.
4. Select **Review + create**, then select **Create** to provision the resource.
5. Wait for deployment to complete, and then go to the deployed resource.
6. View the **Keys and Endpoint** page in the **Resource Management** section. You will need the information on this page later in the exercise.

## Prepare to develop an app in Cloud Shell

1. Leaving the **Keys and Endpoint** page open, use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal.

    > **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

2. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

    **Ensure you've switched to the classic version of the cloud shell before continuing.**

3. In the PowerShell pane, enter the following commands to clone the GitHub repo for this exercise:

    Code Copy

```
rm -r mslearn-ai-language -f
git clone https://github.com/microsoftlearning/mslearn-ai-language
```

> **Tip**: As you enter commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

4. After the repo has been cloned, navigate to the folder containing the code files:

Code Copy

```
cd mslearn-ai-language/Labfiles/08-speech-translation/Python/translator
```

5. In the command line pane, run the following command to view the code files in the **translator** folder:

Code Copy

```
ls -a -l
```

The files include a configuration file (**.env**) and a code file (**translator.py**).

6. Create a Python virtual environment and install the Azure AI Speech SDK package and other required packages by running the following command:

Code Copy

```
python -m venv labenv
./labenv/bin/Activate.ps1
pip install -r requirements.txt azure-cognitiveservices-speech==1.42.0
```

7. Enter the following command to edit the configuration file that has been provided:

Code Copy

```
code .env
```

The file is opened in a code editor.

8. Update the configuration values to include the **region** and a **key** from the Azure AI Speech resource you created (available on the **Keys and Endpoint** page for your Azure AI Translator resource in the Azure portal).

9. After you've replaced the placeholders, use the **CTRL+S** command to save your changes and then use the **CTRL+Q** command to close the code editor while keeping the cloud shell command line open.

## Add code to use the Azure AI Speech SDK

> **Tip**: As you add code, be sure to maintain the correct indentation.

1. Enter the following command to edit the code file that has been provided:

Code Copy

```
code translator.py
```

2. At the top of the code file, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Azure AI Speech SDK:

```
Code
Copy

# Import namespaces
from azure.core.credentials import AzureKeyCredential
import azure.cognitiveservices.speech as speech_sdk
```

3. In the **main** function, under the comment **Get config settings**, note that the code loads the key and region you defined in the configuration file.

4. Find the following code under the comment **Configure translation**, and add the following code to configure your connection to the Azure AI Services Speech endpoint:

```
Code
Copy

# Configure translation
translation_config = speech_sdk.translation.SpeechTranslationConfig(speech_key,
speech_region)
translation_config.speech_recognition_language = 'en-US'
translation_config.add_target_language('fr')
translation_config.add_target_language('es')
translation_config.add_target_language('hi')
print('Ready to translate from',translation_config.speech_recognition_language)
```

5. You will use the **SpeechTranslationConfig** to translate speech into text, but you will also use a **SpeechConfig** to synthesize translations into speech. Add the following code under the comment **Configure speech**:

```
Code
Copy

# Configure speech
speech_config = speech_sdk.SpeechConfig(speech_key, speech_region)
print('Ready to use speech service in:', speech_config.region)
```

6. Save your changes (*CTRL+S*), but leave the code editor open.

## Run the app

So far, the app doesn't do anything other than connect to your Azure AI Speech resource, but it's useful to run it and check that it works before adding speech functionality.

1. In the command line, enter the following command to run the translator app:

```
Code
Copy

python translator.py
```

The code should display the region of the speech service resource the application will use, a message that it is ready to translate from en-US and prompt you for a target language. A successful run indicates that the app has connected to your Azure AI Speech service. Press ENTER to end the program.

# Implement speech translation

Now that you have a **SpeechTranslationConfig** for the Azure AI Speech service, you can use the Azure AI Speech translation API to recognize and translate speech.

1. In the code file, note that the code uses the **Translate** function to translate spoken input. Then in the **Translate** function, under the comment **Translate speech**, add the following code to create a **TranslationRecognizer** client that can be used to recognize and translate speech from a file.

```
# Translate speech
current_dir = os.getcwd()
audioFile = current_dir + '/station.wav'
audio_config_in = speech_sdk.AudioConfig(filename=audioFile)
translator = speech_sdk.translation.TranslationRecognizer(translation_config, audio_config =
audio_config_in)
print("Getting speech from file...")
result = translator.recognize_once_async().get()
print('Translating "{}"'.format(result.text))
translation = result.translations[targetLanguage]
print(translation)
```

2. Save your changes (*CTRL+S*), and re-run the program:

```
python translator.py
```

3. When prompted, enter a valid language code (*fr*, *es*, or *hi*). The program should transcribe your input file and translate it to the language you specified (French, Spanish, or Hindi). Repeat this process, trying each language supported by the application.

> **NOTE**: The translation to Hindi may not always be displayed correctly in the Console window due to character encoding issues.

4. When you're finished, press ENTER to end the program.

> **NOTE**: The code in your application translates the input to all three languages in a single call. Only the translation for the specific language is displayed, but you could retrieve any of the translations by specifying the target language code in the **translations** collection of the result.

# Synthesize the translation to speech

So far, your application translates spoken input to text; which might be sufficient if you need to ask someone for help while traveling. However, it would be better to have the translation spoken aloud in a suitable voice.

> **Note**: Due to the hardware limitations of the cloud shell, we'll direct the synthesized speech output to a file.

1. In the **Translate** function, find the comment **Synthesize translation**, and add the following code to use a **SpeechSynthesizer** client to synthesize the translation as speech and save it as a .wav file:

```
# Synthesize translation
output_file = "output.wav"
voices = {
        "fr": "fr-FR-HenriNeural",
        "es": "es-ES-ElviraNeural",
        "hi": "hi-IN-MadhurNeural"
}
speech_config.speech_synthesis_voice_name = voices.get(targetLanguage)
audio_config_out = speech_sdk.audio.AudioConfig(filename=output_file)
speech_synthesizer = speech_sdk.SpeechSynthesizer(speech_config, audio_config_out)
speak = speech_synthesizer.speak_text_async(translation).get()
if speak.reason != speech_sdk.ResultReason.SynthesizingAudioCompleted:
    print(speak.reason)
else:
    print("Spoken output saved in " + output_file)
```

2. Save your changes (*CTRL+S*), and re-run the program:

Code Copy

```
python translator.py
```

3. Review the output from the application, which should indicate that the spoken output translation was saved in a file. When you're finished, press **ENTER** to end the program.

4. If you have a media player capable of playing .wav audio files, download the file that was generated by entering the following command:

Code Copy

```
download ./output.wav
```

The download command creates a popup link at the bottom right of your browser, which you can select to download and open the file.

> **NOTE** *In this example, you've used a **SpeechTranslationConfig** to translate speech to text, and then used a **SpeechConfig** to synthesize the translation as speech. You can in fact use the **SpeechTranslationConfig** to synthesize the translation directly, but this only works when translating to a single language, and results in an audio stream that is typically saved as a file.*

## Clean up resources

If you're finished exploring the Azure AI Speech service, you can delete the resources you created in this exercise. Here's how:

1. Close the Azure cloud shell pane
2. In the Azure portal, browse to the Azure AI Speech resource you created in this lab.
3. On the resource page, select **Delete** and follow the instructions to delete the resource.

## What if you have a mic and speaker?

In this exercise, the Azure Cloud Shell environment we used doesn't support audio hardware, so you used audio files for the speech input and output. Let's see how the code can be modified to use audio hardware if you have it available.

### Using speech translation with a microphone

1. If you have a mic, you can use the following code to capture spoken input for speech translation:

Code Copy

```
# Translate speech
audio_config_in = speech_sdk.AudioConfig(use_default_microphone=True)
translator = speech_sdk.translation.TranslationRecognizer(translation_config, audio_config =
audio_config_in)
print("Speak now...")
result = translator.recognize_once_async().get()
print('Translating "{}"'.format(result.text))
translation = result.translations[targetLanguage]
print(translation)
```

> **Note**: The system default microphone is the default audio input, so you could also just omit the AudioConfig altogether!

### Using speech synthesis with a speaker

1. If you have a speaker, you can use the following code to synthesize speech.

Code Copy

```
# Synthesize translation
voices = {
        "fr": "fr-FR-HenriNeural",
        "es": "es-ES-ElviraNeural",
        "hi": "hi-IN-MadhurNeural"
}
speech_config.speech_synthesis_voice_name = voices.get(targetLanguage)
audio_config_out = speech_sdk.audio.AudioOutputConfig(use_default_speaker=True)
speech_synthesizer = speech_sdk.SpeechSynthesizer(speech_config, audio_config_out)
speak = speech_synthesizer.speak_text_async(translation).get()
if speak.reason != speech_sdk.ResultReason.SynthesizingAudioCompleted:
    print(speak.reason)
```

> **Note**: The system default speaker is the default audio output, so you could also just omit the AudioConfig altogether!

## More information

For more information about using the Azure AI Speech translation API, see the Speech translation documentation.