Overview of Azure AI Speech Services

Azure AI Speech Services provides a suite of APIs that allow developers to add speech recognition, speech synthesis, and translation capabilities to their applications. The main components include:

- **Speech-to-Text (STT)**: Converts spoken language into written text.
- **Text-to-Speech (TTS)**: Converts written text into spoken language.
- **Speech Translation**: Translates spoken language in real-time.

These services can be integrated into various applications such as virtual assistants, customer service bots, and accessibility tools.

Provision an Azure Resource for Speech

To use Azure AI Speech Services, you first need to provision an Azure resource specifically for speech. Follow these steps to set it up:

Step 1: Sign in to the Azure Portal

- Go to the Azure Portal and sign in with your Azure account credentials.

Step 2: Create a New Speech Resource

- In the Azure Portal dashboard, click on **"Create a resource"**.
- In the search bar, type **"Speech"** and select **"Speech"** from the list of services.

Step 3: Configure the Speech Resource

- **Subscription**: Select your Azure subscription.
- **Resource Group**: Choose an existing resource group or create a new one to organize your resources.
- **Region**: Select the Azure region where you want to host the service. Choose a region close to your users to minimize latency.
- **Name**: Provide a unique name for your Speech resource.
- **Pricing Tier**: Choose a pricing tier based on your expected usage. The free tier allows limited usage, while higher tiers offer more capabilities.

Step 4: Review and Create

- After configuring the settings, click **"Review + create"**.
- Review the settings and click **"Create"** to provision the resource.

Step 5: Access the Resource

- Once the resource is created, you can access it from the **"Resource groups"** section or by searching for it in **"All resources"**.
- Click on the resource to view its details.

Step 6: Obtain API Keys and Endpoints

- In the resource page, find the **"Keys and Endpoint"** section.
- Copy the **API Key** and **Endpoint URL**. These are needed to authenticate and interact with the Speech service in your applications.

3. Integrating Speech Capabilities into Your App

With the Azure Speech resource provisioned, you can now integrate its capabilities into your application:

Speech-to-Text (STT) Integration

- Use the STT API to convert spoken input from users into text.
- Ideal for voice commands, transcription services, and interactive voice response systems.

Text-to-Speech (TTS) Integration

- Use the TTS API to generate spoken output from text.
- Useful for creating voice assistants, reading content aloud, and providing auditory feedback.

Speech Translation Integration

- Combine STT and translation services to enable real-time speech translation.
- Suitable for multilingual communication tools, global customer support, and conferencing solutions.

API Example: Speech-to-Text

<u>Http:</u>

```
POST https://<your-resource-region>.api.cognitive.microsoft.com/speechtotext/v3.0/transcriptions

Ocp-Apim-Subscription-Key: <your-api-key>

Content-Type: application/json
```

```
{
  "contentUrls": ["https://your-blob-storage-url/audio-file.wav"],
  "locale": "en-US",
  "displayName": "Sample Transcription"
}
```

In this example:

- Replace **<your-resource-region>** with the region where your resource is hosted (e.g., ` **eastus** `).
- Replace **<your-api-key>** with your Speech resource API key.
- Provide the URL of the audio file you want to transcribe.

Provisioning an Azure resource for speech is the first step toward creating speech-enabled applications using Azure AI Services. Once the resource is set up, you can integrate advanced speech capabilities like speech-to-text, text-to-speech, and real-time translation into your applications, enhancing user interaction and accessibility. These services can be applied across various industries, from customer support to education, making your applications more intuitive and versatile.

Using the Azure AI Speech-to-Text API

Azure AI Speech-to-Text (STT) API enables developers to convert spoken language into written text, which can be integrated into various applications, such as transcription services, voice commands, or interactive voice response (IVR) systems. Here's a step-by-step guide to using the Azure AI Speech-to-Text API.

1. Prerequisites

Before you can use the Speech-to-Text API, ensure you have the following:

- An active Azure subscription.
- A Speech resource provisioned in Azure (as detailed in the previous instructions).
- The API key and endpoint URL for your Speech resource.

2. Setting Up Your Environment

To interact with the Azure Speech-to-Text API, you can use tools like`curl `, Postman, or a programming language such as Python, C#, or JavaScript. Below are examples of how to use the API with `curl `and Python.

3. Basic API Call with curl

Here’s how you can make a basic API call using curl to transcribe an audio file:

Bash:

```
curl -X POST "https://<your-resource-region>.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?language=en-US" \
-H "Ocp-Apim-Subscription-Key: <your-api-key>" \
-H "Content-Type: audio/wav" \
--data-binary "@path/to/your/audio-file.wav"
```

- Replace `**<your-resource-region>**` with the Azure region where your Speech resource is located (e.g., **eastus**).
- Replace `**<your-api-key>**` with your Speech API key.
- Replace `**@path/to/your/audio-file.wav**` with the path to your audio file.

Response: The API will return a JSON object with the transcription of the audio file.

4. Using the API with Python

To use the Speech-to-Text API with Python, you’ll need the azure-cognitiveservices-speech library.

Step 1: Install the Azure Speech SDK

Bash:

```
pip install azure-cognitiveservices-speech
```

Step 2: Sample Python Script Here’s a Python script that uses the Speech SDK to transcribe speech from an audio file:

Python:

```
import os

import azure.cognitiveservices.speech as speechsdk


# Set up your subscription info

speech_key = "Your-API-Key"

service_region = "Your-Region"


# Create an instance of a speech config with the specified subscription key and service
region

speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)


# Specify the path to the audio file

audio_filename = "path/to/your/audio-file.wav"


# Create an audio configuration using the audio file

audio_input = speechsdk.AudioConfig(filename=audio_filename)


# Create a speech recognizer with the given settings

speech_recognizer = speechsdk.SpeechRecognizer(speech_config=speech_config,
audio_config=audio_input)


# Start the recognition and wait for a result

result = speech_recognizer.recognize_once()


# Check the result

if result.reason == speechsdk.ResultReason.RecognizedSpeech:

    print("Recognized: {}".format(result.text))

elif result.reason == speechsdk.ResultReason.NoMatch:
    print("No speech could be recognized")
else:
    print("Speech Recognition canceled: {}".format(result.cancellation_details.reason))
```

Explanation:

- Replace "Your-API-Key" with your Azure Speech API key.
- Replace "Your-Region" with the Azure region where your Speech resource is hosted.
- Replace path/to/your/audio-file.wav with the path to your audio file.

Running the Script:

- When you run the script, it will print the recognized text from the audio file to the console.

5. Advanced Features and Customization

The Speech-to-Text API offers several advanced features, including:

- **Continuous Recognition**: Use the API to transcribe longer audio streams in real-time.
- **Custom Models**: Integrate custom speech models trained with domain-specific vocabulary to improve accuracy.
- **Punctuation**: Enable automatic punctuation in the transcribed text.
- **Language Support**: Specify different languages and dialects for the transcription.

Example: Continuous Recognition in Python

Python:

```
def recognized(evt):
    print('RECOGNIZED: {}'.format(evt.result.text))


speech_recognizer.recognized.connect(recognized)


# Start continuous recognition
speech_recognizer.start_continuous_recognition()
```

```
# Keep the recognition running

import time

while True:

    time.sleep(0.5)
```

This script sets up continuous recognition and prints the transcriptions in real-time.

Using the Azure AI Text-to-Speech API

The Azure AI Text-to-Speech (TTS) API allows you to convert written text into natural-sounding speech. This can be integrated into applications to provide auditory feedback, voice-based interactions, or accessibility features. Below is a step-by-step guide on how to use the Text-to-Speech API.

1. Prerequisites

Before using the Text-to-Speech API, ensure you have:

- An active Azure subscription.
- A Speech resource provisioned in Azure (as described in the previous instructions).
- The API key and endpoint URL for your Speech resource.

2. Basic API Call with` curl `

Here’s how you can make a basic API call using` **curl** ` to convert text to speech:

Bash:

```
curl -X POST "https://<your-resource-region>.tts.speech.microsoft.com/cognitiveservices/v1" \

-H "Ocp-Apim-Subscription-Key: <your-api-key>" \

-H "Content-Type: application/ssml+xml" \

-H "X-Microsoft-OutputFormat: riff-24khz-16bit-mono-pcm" \

-H "User-Agent: curl" \
-d "<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-US'><voice name='en-US-JennyNeural'>Hello, this is a text-to-speech conversion.</voice></speak>" \

--output output.wav
```

- **Replace `<your-resource-region>`** with the Azure region where your Speech resource is located (e.g.,` **eastus** `).
- **Replace `<your-api-key>`** with your Speech API key.
- **The `-d` flag** specifies the SSML (Speech Synthesis Markup Language) input, which includes the text to be spoken and the voice selection.
- **The `--output` flag** specifies the output file where the synthesized speech will be saved (in this case,` **output.wav** `).

Explanation:

- **X-Microsoft-OutputFormat**: Specifies the audio format (e.g., PCM, MP3, etc.).
- **voice name**: Selects the specific voice to use (in this case,` **en-US-JennyNeural** `).

Response: The API returns an audio file (`**output.wav** `) containing the spoken text.

3. Using the API with Python

You can also use the Azure Text-to-Speech API with Python by utilizing the **azure-cognitiveservices-speech** library.

Step 1: Install the Azure Speech SDK

Bash:

```
pip install azure-cognitiveservices-speech
```

Step 2: Sample Python Script

Here’s a Python script that uses the Speech SDK to convert text to speech:

Python:

```
import os

import azure.cognitiveservices.speech as speechsdk

# Set up your subscription info

speech_key = "Your-API-Key"

service_region = "Your-Region"


# Create an instance of a speech config with the specified subscription key and service
region

speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)


# Set the voice (optional)

speech_config.speech_synthesis_voice_name = "en-US-JennyNeural"


# Create a synthesizer using the default speaker as audio output

synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
audio_config=None)


# Text to be synthesized

text = "Hello, this is a text-to-speech conversion."


# Synthesize the text to speech

result = synthesizer.speak_text_async(text).get()


# Check the result

if result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:

    print("Speech synthesized successfully.")

elif result.reason == speechsdk.ResultReason.Canceled:

    cancellation_details = result.cancellation_details

    print("Speech synthesis canceled: {}".format(cancellation_details.reason))
```

```
  if cancellation_details.error_details:
    print("Error details: {}".format(cancellation_details.error_details))
```

Explanation:

- Replace **"Your-API-Key"** with your Azure Speech API key.
- Replace **"Your-Region"** with the Azure region where your Speech resource is hosted.
- The ` **speech_synthesis_voice_name** `specifies the voice to use (in this case**,`** **en-US-JennyNeural** `).

Running the Script:

- When you run the script, it will synthesize the text and play the speech through your device's default speaker.

4. Advanced Features and Customization

The Azure Text-to-Speech API offers several advanced features, including:

- **SSML Support**: Use SSML to control various aspects of speech synthesis, such as pitch, rate, volume, and pauses.
- **Custom Voices**: Create custom voices to reflect your brand's unique tone or style.
- **Audio Formats**: Choose from different audio output formats, such as PCM, MP3, and Ogg Vorbis.
- **Language and Voice Selection**: Select from a wide range of languages and voices to suit your application's needs.

Example: Using SSML for Advanced Control

Here's an example SSML input that controls pitch and speed:

XML:

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
 <voice name="en-US-JennyNeural">
  <prosody pitch="+10%" rate="-20%">This is a customized text-to-speech conversion.</prosody>
 </voice>
</speak>
```

- **<prosody>**: Controls the pitch and rate of the speech.

Configure audio format and voices

1. Configuring Audio Format

Azure AI Text-to-Speech supports various audio output formats. You can specify the desired format in the request headers when making an API call.

Supported Audio Formats

- **PCM**: Uncompressed audio, suitable for high-quality output.
  - `riff-24khz-16bit-mono-pcm`
  - `riff-16khz-16bit-mono-pcm`
  - `riff-8khz-16bit-mono-pcm`
- **MP3**: Compressed audio, ideal for saving bandwidth.
  - `audio-16khz-32kbitrate-mono-mp3`
  - `audio-16khz-128kbitrate-mono-mp3`
  - `audio-24khz-96kbitrate-mono-mp3`
- **OGG Vorbis**: Another compressed format, offering efficient streaming.
  - `audio-24khz-48kbitrate-mono-ogg`
  - `audio-48khz-96kbitrate-mono-ogg`

Example with curl:

Bash:

```bash
curl -X POST "https://<your-resource-
region>.tts.speech.microsoft.com/cognitiveservices/v1" \

-H "Ocp-Apim-Subscription-Key: <your-api-key>" \

-H "Content-Type: application/ssml+xml" \

-H "X-Microsoft-OutputFormat: audio-24khz-96kbitrate-mono-mp3" \

-H "User-Agent: curl" \

-d "<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-
US'><voice name='en-US-JennyNeural'>This is a custom audio format
example.</voice></speak>" \

--output output.mp3
```

2. Configuring Voices

Azure offers a wide range of voices, including standard voices and neural voices with more natural intonations. You can select a voice by specifying its name in the SSML or the Speech SDK configuration.

Voice Selection in SSML:

XML:

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-GuyNeural">This is an example using the Guy neural voice.</voice>
</speak>
```

Neural Voices: Use more advanced machine learning models for a more natural-sounding voice.

- Example: `**en-US-JennyNeural**`, `**en-US-GuyNeural**`

Standard Voices: Traditional synthesized voices.

- Example: `**en-US-ZiraRUS**`

3. Configuring in Python with the Speech SDK

Here's an example of how to configure both the audio format and the voice using Python:

Python:

```python
import azure.cognitiveservices.speech as speechsdk
# Set up your subscription info
speech_key = "Your-API-Key"
service_region = "Your-Region"

# Create a speech configuration with the desired voice and format
speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)
speech_config.speech_synthesis_voice_name = "en-US-GuyNeural"
speech_config.set_speech_synthesis_output_format(speechsdk.SpeechSynthesisOutputFor
mat.Audio24Khz96KBitRateMonoMp3)

# Create a synthesizer with audio output to a file

audio_config = speechsdk.audio.AudioOutputConfig(filename="output.mp3")

synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
audio_config=audio_config)


# Synthesize the text to speech

text = "This is a text-to-speech conversion with a custom voice and audio format."

result = synthesizer.speak_text_async(text).get()


# Check the result

if result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:

    print("Speech synthesized successfully.")

elif result.reason == speechsdk.ResultReason.Canceled:

    print("Speech synthesis canceled: {}".format(result.cancellation_details.reason))
```

By configuring the audio format and selecting the appropriate voice, you can tailor the Azure AI Text-to-Speech output to match your application's requirements. Whether you need high-quality uncompressed audio or efficient streaming formats, Azure offers flexibility to meet those needs. Similarly, with a wide range of voices, including neural options, you can ensure that the generated speech aligns with your brand’s tone and style.

Use Speech Synthesis Markup Language

Speech Synthesis Markup Language (SSML) is a powerful tool for customizing how text is converted to speech using the Azure AI Text-to-Speech API. SSML allows you to control aspects such as pitch, rate, volume, pauses, and even select different voices within a single speech output.

1. Basic SSML Structure

An SSML document is an XML-based structure that defines how text should be spoken. Here's a basic example:

XML:

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    Hello, this is a text-to-speech conversion using SSML.
  </voice>
</speak>
```

2. Advanced SSML Features

You can use SSML to control various aspects of the speech, such as:

a. Controlling Prosody (Pitch, Rate, Volume)

XML:

```xml
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    <prosody pitch="+10%" rate="slow" volume="loud">This is a customized speech with
SSML.</prosody>
  </voice>
</speak>
```

pitch: Controls the pitch of the speech (e.g., +10%, -10%).

rate: Controls the speed of the speech (e.g., slow, fast, medium).
volume: Controls the volume of the speech (e.g., soft, loud).

b. Adding Pauses

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    Welcome to the Azure AI Text-to-Speech service.
    <break time="500ms"/>
    This service converts text to speech.
  </voice>
</speak>
```

`**<break>**`: Inserts a pause. The `**time**` attribute specifies the duration of the pause.

c. Emphasizing Words

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    <emphasis level="strong">This</emphasis> is important.
  </voice>
</speak>
```

<emphasis>: Adds emphasis to the word or phrase. The `**level**` attribute can be` **strong**`, `**moderate**`, or `**reduced**`.

d. Selecting Different Voices for Different Sections

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    Hello, I am Jenny.
  </voice>
  <voice name="en-US-GuyNeural">
    And I am Guy.
  </voice>
</speak>
```

<voice>: Switches the voice mid-speech to create a conversation or dialogue.

e. Specifying Pronunciation with `<phoneme>`

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-JennyNeural">
    The word <phoneme alphabet="ipa" ph="ˈæz.jʊər">Azure</phoneme> is pronounced as "ˈæz.jʊər".
  </voice>
</speak>
```

<phoneme>: Provides a phonetic pronunciation for the word. The `**alphabet**` attribute specifies the phonetic alphabet used, such as IPA (International Phonetic Alphabet).

3. Using SSML with the Azure Text-to-Speech API

Using curl

You can include your SSML markup directly in the ` **curl** `command:

Bash:

```
curl -X POST "https://<your-resource-region>.tts.speech.microsoft.com/cognitiveservices/v1" \
-H "Ocp-Apim-Subscription-Key: <your-api-key>" \
-H "Content-Type: application/ssml+xml" \
-H "X-Microsoft-OutputFormat: riff-24khz-16bit-mono-pcm" \
-H "User-Agent: curl" \
-d "<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-US'><voice name='en-US-JennyNeural'><prosody pitch='+10%' rate='slow' volume='loud'>This is a customized speech with SSML.</prosody></voice></speak>" \
--output output.wav
```

Using SSML in Python with Azure SDK

You can also use SSML in Python with the Azure SDK:

Python:

```
import azure.cognitiveservices.speech as speechsdk

# Set up your subscription info

speech_key = "Your-API-Key"

service_region = "Your-Region"


# Create a speech configuration

speech_config = speechsdk.SpeechConfig(subscription=speech_key, region=service_region)


# Create a synthesizer with audio output to a file

audio_config = speechsdk.audio.AudioOutputConfig(filename="output.wav")

synthesizer = speechsdk.SpeechSynthesizer(speech_config=speech_config,
audio_config=audio_config)


# SSML input text

ssml_text = """

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">

  <voice name="en-US-JennyNeural">

    <prosody pitch="+10%" rate="slow" volume="loud">This is a customized speech with
SSML.</prosody>

  </voice>

</speak>

"""


# Synthesize the SSML text to speech

result = synthesizer.speak_ssml_async(ssml_text).get()


# Check the result

if result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:

    print("Speech synthesized successfully.")
```

```
elif result.reason == speechsdk.ResultReason.Canceled:
    print("Speech synthesis canceled: {}".format(result.cancellation_details.reason))
```

SSML offers fine-grained control over how text is synthesized into speech. By using SSML, you can adjust the pitch, rate, volume, insert pauses, emphasize words, switch voices, and even provide custom pronunciations. This allows you to create highly customized and dynamic speech outputs that can be tailored to the specific needs of your application or audience.

For better understanding use below link for reference

https://learn.microsoft.com/en-us/training/modules/create-speech-enabled-apps/