


Quickstart: Convert text-to-speech using Python

12/08/2019 • 4 minutes to read •  +5

In this article

Prerequisites

[Create a project and import required modules](#)

[Set the subscription key and create a prompt for TTS](#)

[Get an access token](#)

[Make a request and save the response](#)

[Put it all together](#)

[Run the sample app](#)

[Clean up resources](#)

[Next steps](#)

[See also](#)

In this quickstart, you'll learn how to convert text-to-speech using Python and the text-to-speech REST API. The request body in this guide is structured as [Speech Synthesis Markup Language \(SSML\)](#), which allows you to choose the voice and language of the response.

This quickstart requires an [Azure Cognitive Services account](#) with a Speech service resource. If you don't have an account, you can use the [free trial](#) to get a subscription key.

Prerequisites

This quickstart requires:

- Python 2.7.x or 3.x
- [Visual Studio](#), [Visual Studio Code](#), or your favorite text editor
- An Azure subscription key for the Speech service

Create a project and import required modules

Create a new Python project using your favorite IDE or editor. Then copy this code snippet into your project in a file named `tts.py`.

Python

 Copy

```
import os
import requests
import time
from xml.etree import ElementTree
```

ⓘ Note

If you haven't used these modules you'll need to install them before running your program. To install these packages, run: `pip install requests`.

These modules are used to write the speech response to a file with a timestamp, construct the HTTP request, and call the text-to-speech API.

Set the subscription key and create a prompt for TTS

In the next few sections you'll create methods to handle authorization, call the text-to-speech API, and validate the response. Let's start by adding some code that makes sure this sample will work with Python 2.7.x and 3.x.

Python

 Copy

```
try:
    input = raw_input
except NameError:
    pass
```

Next, let's create a class. This is where we'll put our methods for token exchange, and calling the text-to-speech API.

Python

 Copy

```
class TextToSpeech(object):
    def __init__(self, subscription_key):
        self.subscription_key = subscription_key
        self.tts = input("What would you like to convert to speech: ")
        self.timestr = time.strftime("%Y%m%d-%H%M")
        self.access_token = None
```

The `subscription_key` is your unique key from the Azure portal. `tts` prompts the user to enter text that will be converted to speech. This input is a string literal, so characters don't need to be escaped. Finally, `timestr` gets the current time, which we'll use to name your file.

Get an access token

The text-to-speech REST API requires an access token for authentication. To get an access token, an exchange is required. This sample exchanges your Speech service subscription key for an access token using the `issueToken` endpoint.

This sample assumes that your Speech service subscription is in the West US region. If you're using a different region, update the value for `fetch_token_url`. For a full list, see [Regions](#).

Copy this code into the `TextToSpeech` class:

Python

 Copy

```
def get_token(self):
    fetch_token_url =
    "https://westus.api.cognitive.microsoft.com/sts/v1.0/issueToken"
    headers = {
        'Ocp-Apim-Subscription-Key': self.subscription_key
    }
    response = requests.post(fetch_token_url, headers=headers)
    self.access_token = str(response.text)
```

ⓘ Note

For more information on authentication, see [Authenticate with an access token](#).

Make a request and save the response

Here you're going to build the request and save the speech response. First, you need to set the `base_url` and `path`. This sample assumes you're using the West US endpoint. If your resource is registered to a different region, make sure you update the `base_url`. For more information, see [Speech service regions](#).

Next, you need to add required headers for the request. Make sure that you update `User-Agent` with the name of your resource (located in the Azure portal), and set `X-Microsoft-OutputFormat` to your preferred audio output. For a full list of output formats, see [Audio outputs](#).

Then construct the request body using Speech Synthesis Markup Language (SSML). This sample defines the structure, and uses the `tts` input you created earlier.

ⓘ Note

This sample uses the `Guy24KRUS` voice font. For a complete list of Microsoft provided voices/languages, see [Language support](#). If you're interested in creating a unique, recognizable voice for your brand, see [Creating custom voice fonts](#).

Finally, you'll make a request to the service. If the request is successful, and a 200 status code is returned, the speech response is written to a timestamped file.

Copy this code into the `TextToSpeech` class:

Python

 Copy

```
def save_audio(self):
    base_url = 'https://westus.tts.speech.microsoft.com/'
    path = 'cognitiveservices/v1'
    constructed_url = base_url + path
    headers = {
        'Authorization': 'Bearer ' + self.access_token,
        'Content-Type': 'application/ssml+xml',
        'X-Microsoft-OutputFormat': 'riff-24khz-16bit-mono-pcm',
        'User-Agent': 'YOUR_RESOURCE_NAME'
    }
    xml_body = ElementTree.Element('speak', version='1.0')
    xml_body.set('{http://www.w3.org/XML/1998/namespace}lang', 'en-us')
    voice = ElementTree.SubElement(xml_body, 'voice')
    voice.set('{http://www.w3.org/XML/1998/namespace}lang', 'en-US')
    voice.set(
        'name', 'Microsoft Server Speech Text to Speech Voice (en-US,
Guy24KRUS)')
    voice.text = self.tts
    body = ElementTree.tostring(xml_body)

    response = requests.post(constructed_url, headers=headers, data=body)
    if response.status_code == 200:
        with open('sample-' + self.timestr + '.wav', 'wb') as audio:
            audio.write(response.content)
```

```
print("\nStatus code: " + str(response.status_code) +
      "\nYour TTS is ready for playback.\n")
else:
    print("\nStatus code: " + str(response.status_code) +
          "\nSomething went wrong. Check your subscription key and
headers.\n")
```

Put it all together

You're almost done. The last step is to instantiate your class and call your functions.

Python

 Copy

```
if __name__ == "__main__":
    subscription_key = "YOUR_KEY_HERE"
    app = TextToSpeech(subscription_key)
    app.get_token()
    app.save_audio()
```

Run the sample app

That's it, you're ready to run your text-to-speech sample app. From the command line (or terminal session), navigate to your project directory and run:

console

 Copy

```
python tts.py
```

When prompted, type in whatever you'd like to convert from text-to-speech. If successful, the speech file is located in your project folder. Play it using your favorite media player.

Clean up resources

Make sure to remove any confidential information from your sample app's source code, like subscription keys.

Next steps

[Explore Python samples on GitHub](#)

See also

- [Text-to-speech API reference](#)
- [Using Python and Speech SDK to convert text-to-speech](#)
- [Creating custom voice fonts](#)
- [Record voice samples to create a custom voice](#)

Is this page helpful?

 Yes  No
