

Thank you for purchasing this asset. I hope you enjoy using it.

Contact my [here](#), for bug reports, suggestions, or feedback.

Don't forget to [leave a review](#) if you enjoy using Text generation toolkit, this helps me a lot

Table of Content

[Table of Content](#)

[Fixing Errors do to updating the package](#)

[How to set up Azure Speech](#)

[Generate Audio Clip Window](#)

[Get Text From Speech Window](#)

[Log All Mic Names](#)

[How to use Text To Speech \(Core\)](#)

[How to use Speech To Text \(Core\)](#)

[How to use Audio Converter \(Core\)](#)

[How to use Microphone Utils \(Core\)](#)

[AzureSpeechEnums, and Jsons \(Core\)](#)

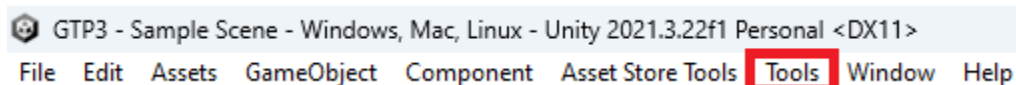
How to set up Azure Speech

1. Go to [Azure Portal](#), and sign in, or make a new free account
 - a. If you believe your project will really take off, sign up [Microsoft's Founders Hub](#), and [get thousands of dollars in free credits](#).
2. Under Azure services, click “Create a resource”.
3. search for “Speech”, and then click the one made by microsoft. Then click create.
4. Now choose a subscription, a Resource group, a Region, a Name, and a pricing tier.
5. In the bottom left corner of the page click Review + create.
6. Click create. This may take a few seconds. Then click go to resource.
7. Scroll down and copy the Key 1, or Key 2.
8. Now go back into your Unity project with this plugin in it.
9. Go to
Plugins/TextAndVoiceGenerationToolkit/AzureSpeech/Config/Editor/Azure
Speech Key.
10. Now past the key onto the subscription key variable.
11. Now open the **Azure Speech Default Values** Scriptable Object, in the same location of the **Azure Speech Key** scriptable object.
12. Now select the region of your recourse, and the default language.
13. You are now good to go!

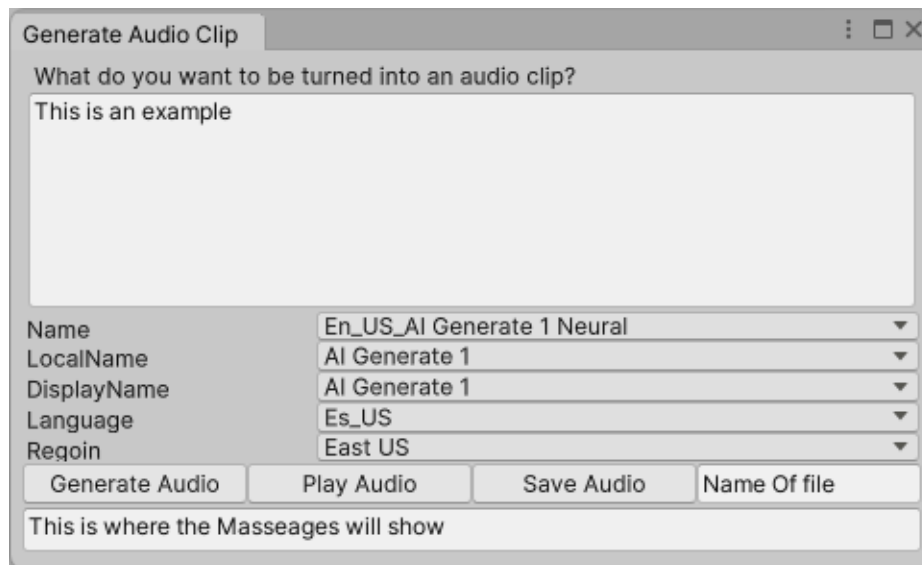
Generate Audio Clip Window

This is an editor window that allows you to Generate, Play, and Save audio clips.

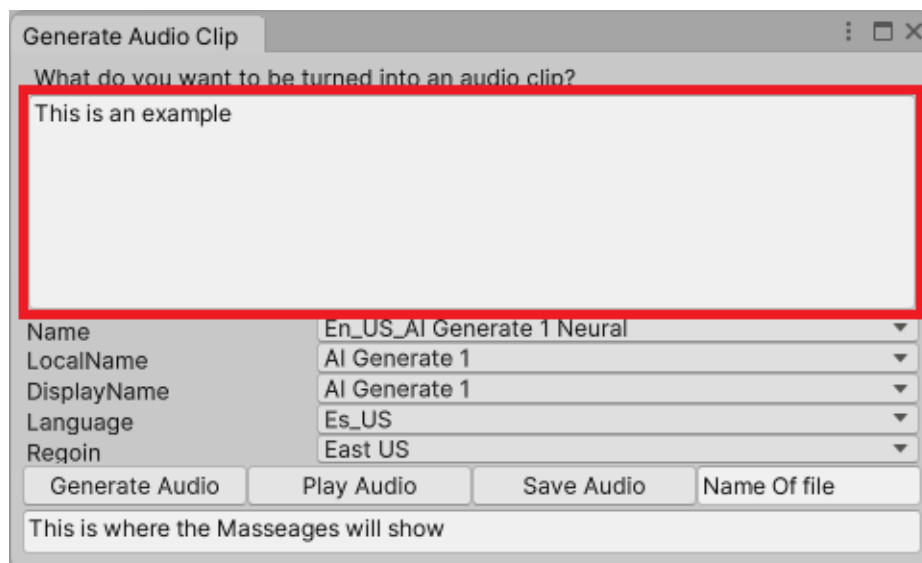
1. To open the Generate Audio Clip window, go to **Tools/Text and Voice Generation Toolkit/Azure Speech/Generate Audio Clip** in the top left corner of your screen



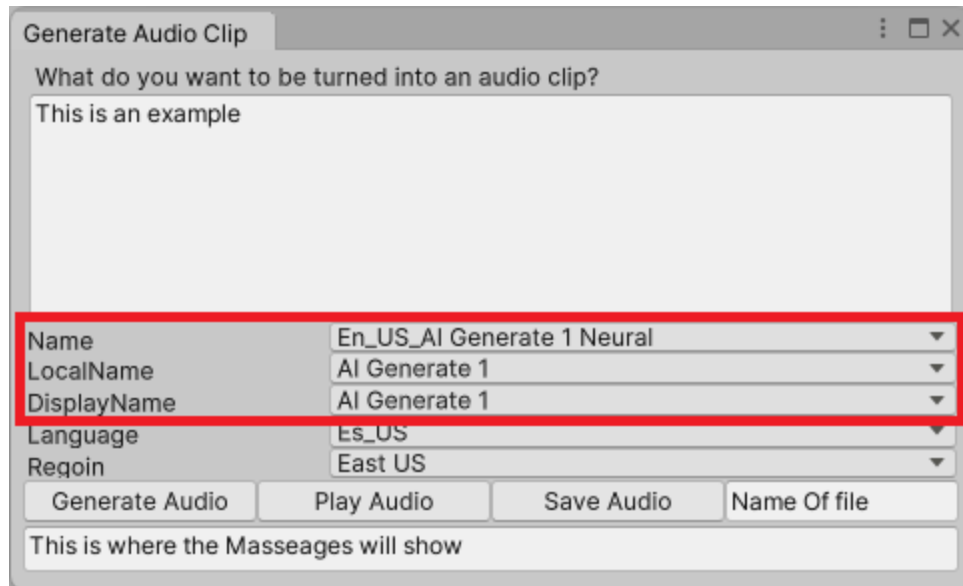
2. Now you should be presented by this window.



3. Enter what you would like to be transcribed into this box.



4. Now you have to choose your name. You have three different options: Name, Local Name, Display Name. if you change one it will change all the others to match.



Generate Audio Clip

What do you want to be turned into an audio clip?

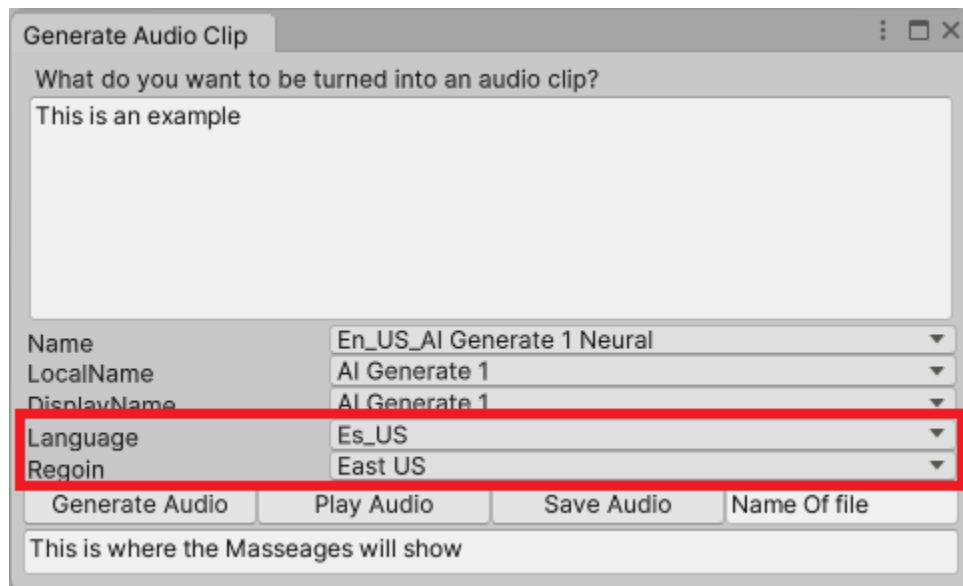
This is an example

Name	En_US_AI Generate 1 Neural
LocalName	AI Generate 1
DisplayName	AI Generate 1
Language	Es_US
Region	East US

Generate Audio Play Audio Save Audio Name Of file

This is where the Masseages will show

5. **Language** this is the language that the AI will speak. And the **Region** is the region of you recourse



Generate Audio Clip

What do you want to be turned into an audio clip?

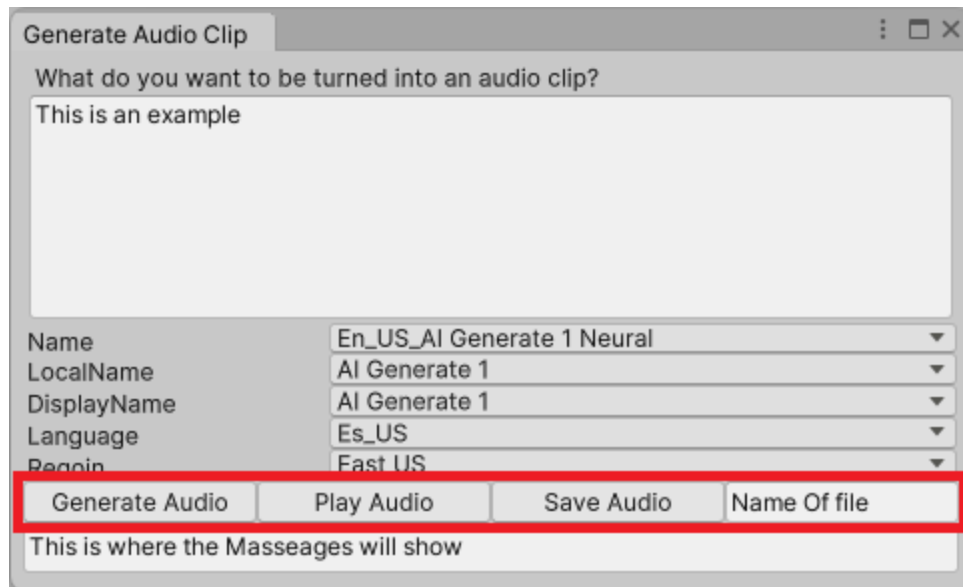
This is an example

Name	En_US_AI Generate 1 Neural
LocalName	AI Generate 1
DisplayName	AI Generate 1
Language	Es_US
Region	East US

Generate Audio Play Audio Save Audio Name Of file

This is where the Masseages will show

6. Then there are the **Generate Audio**, **Play Audio**, and **Save Audio** buttons.



- a. The **Generate Audio** button will call Azure Text-To-Speech to get the audio.

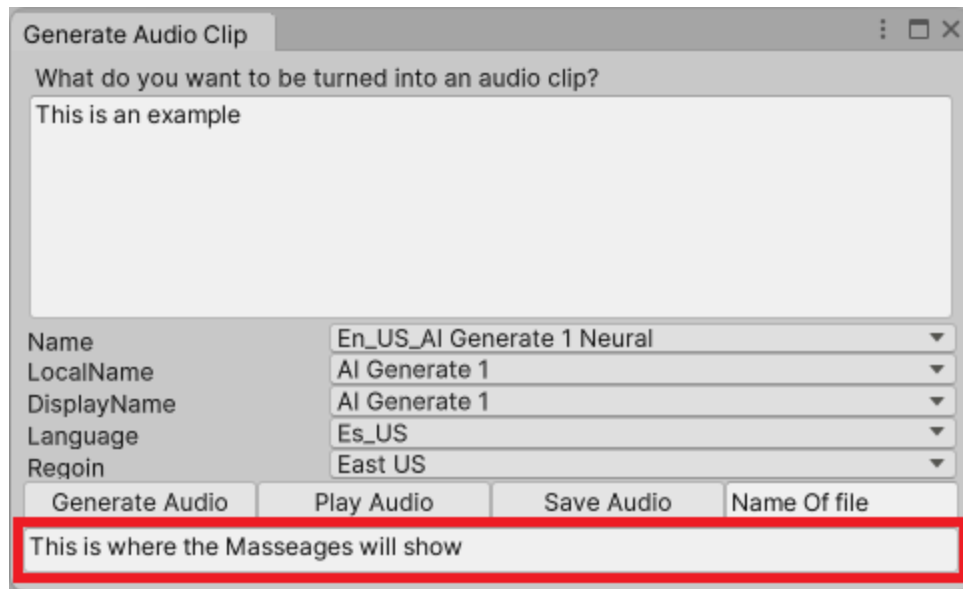
- b. The **Play Audio** button will play the audio that is generated.

NOTE: It will not automatically be called after you generate the audio.

- c. The **Save Audio** button will save the audio as a .wav file. It will be at <YOUR UNITY PROJECT LOCATION>\Assets\Audio.

IMPORTANT: unity will not immediately generate the mete file. To force unity to generate it, open a new window, then close it.

7. Then there is a makeshift console. It will tell you when the audio file is finished being generated, and when the audio file has finished being saved.



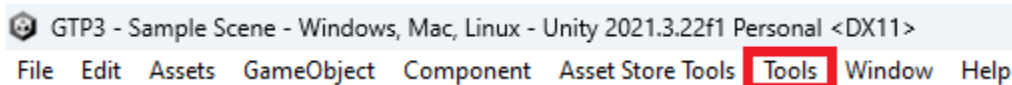
The screenshot shows a window titled "Generate Audio Clip" with standard window controls (minimize, maximize, close) in the top right corner. Inside the window, there is a text input area with the placeholder text "What do you want to be turned into an audio clip?" and an example text "This is an example". Below the input area, there are five dropdown menus for configuration: "Name" (set to "En_US_AI Generate 1 Neural"), "LocalName" (set to "AI Generate 1"), "DisplayName" (set to "AI Generate 1"), "Language" (set to "Es_US"), and "Region" (set to "East US"). At the bottom of the window, there are four buttons: "Generate Audio", "Play Audio", "Save Audio", and "Name Of file". Below these buttons is a console area with a red border containing the text "This is where the Masseages will show".

8. If you have any more questions please let me know [here](#).

Get Text From Speech Window

This window is designed to allow you to talk into your mic, and get a response from it.

1. To open the Get Text From Speech Window, go to **Tools/Text and Voice Generation Toolkit/Azure Speech/Get Text From Speech** in the top left corner of your screen



2. You should be presented with a window similar to this.

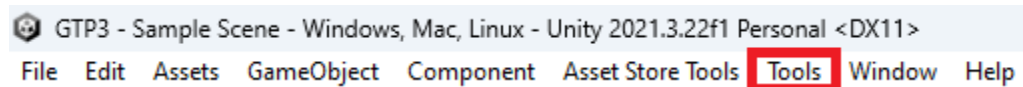


3. The Microphone button will start and stop recording from your microphone.
4. The Play Audio Button will play the recorded Audio.
NOTE: There might be some part of the audio that is cut off, because I have it record the audio for 1 second then add the audio to another audio clip. If you know of a better way please contact me [here](#).
5. The Get Text Button will transcribe the audio.
6. Microphone Names has a list of all the microphones unity recognizes.

Log All Mic Names

This will log all of the names of the mics that unity recognizes, and the index of it.

1. To Log all the mic names, go to **Tools/Text and Voice Generation Toolkit/Azure Speech/Log all Mic names** in the top left corner of your screen



Azure Text To Speech Component

NOTE: This can not be used in editor, because of how the unity audio clip works. If you would like to use it in editor then you can use the [Generate Audio Clip](#)

2. Attach the “**Azure Text To Speech**” component to a gameobject.
3. Now drag and drop the “**Azure Speech Default Values**” scriptable object onto the Default Values variable.
4. Change the input to whatever you would like it to be.
5. Change the language to one your preferred language.
6. Lastly, Change the name to whatever you would like it to be.

Now I will show you how to call it though code!

7. Add this line of code to the top of your script

```
using TheAshBot.Assets.TextAndVoiceGenerationToolkit.AzureSpeech.MonoBehaviours;
```

8. In the code get a reference to the script, and call the “**CallAzure**” function.

```
AzureTextToSpeech azureTextToSpeech = GetComponent<AzureTextToSpeech>();  
AudioClip audioClip = await azureTextToSpeech.CallAzure("Hello World!");
```

Azure Speech To Text Component

NOTE: because of how audio clips work in unity you will need to use the [Get Text From Speech](#) to transcribe what you speak into your mic without using an external program to record the audio.

1. Attach the “**Azure Speech To Text**” component to a gameobject.
2. Drag and drop the audio clip that you would like to be transcribed into the Audio Clip variable.
3. Then drag and drop the “**Azure Speech Default Values**” scriptable object onto the Default Values variable.
4. Lastly, click the Get Text button.
5. After a little bit you should see a new text box that has the audio transcribed.

Now I will show you how to call it though code!

6. Add this line of code to the top of your script

```
using TheAshBot.Assets.TextAndVoiceGenerationToolkit.AzureSpeech.MonoBehaviours;
```

7. Add this code to your script where you would like to call it.

```
AzureSpeechToText azureSpeechToText = GetComponent<AzureSpeechToText>();  
string text = await azureSpeechToText.CallAzure(audioClip);
```

How to use Text To Speech (Core)

1. Make a new C# script for testing.

2. Put this line of code at the top.

```
using TheAshBot.Assets.TextAndVoiceGenerationToolkit.AzureSpeech.Core;
```

3. For testing I would recommend putting your code in the Start function.

```
private void Start()
{
}
}
```

4. Now you need to make a new TextToSpeech variable.

```
TextToSpeech textToSpeech = new TextToSpeech();
```

5. Now you need to pass in the “**languageOfSpeech**”, “**region**”, and “**subscriptionKey**”.

a. “**languageOfSpeech**” is an enum value.

```
AzureSpeechEnums.languages.en_US
```

b. “**Region**” is an enum value.

```
AzureSpeechEnums.Regions.EastUS
```

c. “**subscriptionKey**” is a string value. You can get it by following [How to set up Azure Speech](#)

```
"<YOU API KEY>"
```

6. Now you have to call this function. That returns a audio clip

```
AudioClip audioClip = await textToSpeech.CallAzure();
```

7. Now you have to pass in the “**text**”, and “**nameOfSpeaker**” variables

- a. “**Text**” is a string value. It is the text that the AI will say.

```
"Hello there traveler!"
```

- b. “**nameOfSpeaker**” is an enum value. And there are multiple different enums you can pass in such as “**AzureSpeechEnums.ShortNames**”, “**AzureSpeechEnums.LocalNames**”, and “**AzureSpeechEnums.DisplayNames**”

- i. “**AzureSpeechEnums.ShortNames**”

```
AzureSpeechEnums.ShortNames.en_US_AIGenerate1Neural
```

- ii. “**AzureSpeechEnums.LocalNames**”

```
AzureSpeechEnums.LocalNames.AIGenerate1
```

- iii. “**AzureSpeechEnums.DisplayNames**”

```
AzureSpeechEnums.DisplayNames.AIGenerate1
```

```
private async void Start()
{
    TextToSpeech textToSpeech = new TextToSpeech(AzureSpeechEnums.languages.en_US,
        AzureSpeechEnums.Regions.EastUS, "<YOU API KEY>");

    AudioClip audioClip = await textToSpeech.CallAzure("Hello there traveler!",
        AzureSpeechEnums.ShortNames.en_US_AIGenerate1Neural);
}
```

How to use Speech To Text (Core)

1. Follow the first 3 steps of [How to use the Text To Speech](#).
2. Now you will need a way to get the audio clip. You can do so by putting this code into your script

```
[SerializeField] private AudioClip audioClip;
```

3. Now back in your start function you need to make a new Speech to text variable.

```
SpeechToText speechToText = new SpeechToText();
```

4. Now you need to pass in the “**languageOfSpeech**” “**region**”, and “**subscriptionKey**”.

- a. “**languageOfSpeech**” is an enum value.

```
AzureSpeechEnums.languages.en_US
```

- b. “**Region**” is an enum value.

```
AzureSpeechEnums.Regions.EastUS
```

- c. “**subscriptionKey**” is a string value. You can get it by following [How to set up Azure Speech](#)

```
"<YOU API KEY>"
```

5. Now you actually have to call the function to make the web call. So just put his line of code in your start function.

```
await speechToText.GetTextFromSpeech();
```

- a. But to use “**await**” you need to make the start function an async by making this simple change to it.

```
private async void Start()
```

6. Now you have to **get the returned value**, and **pass in the audio clip**.
- Getting the returned value.** To get the returned value you need to put code before you call the function, but on the same line as the function.

```
var response = await speechToText.GetTextFromSpeech();
```

- pass in the audio clip.** Now you have to pass in the audio clip, because how else would the code know which audio you want it to transcribe. To pass the audio clip into the function you just need to change the function call to this.

```
var response = await speechToText.GetTextFromSpeech(audioClip);
```

7. Now you may or may not have realized that it does not return a string. It returns a class called **SpeechToTextResponse**. To get the transcription you need to put this line of code.

```
string transcription = response.DisplayText;
```

8. And now you are good to go!

```
[SerializeField] private AudioClip audioClip;

private async void Start()
{
    SpeechToText speechToText = new SpeechToText(AzureSpeechEnums.languages.en_US,
        AzureSpeechEnums.Regions.EastUS, "<YOU API KEY>");

    var response = await speechToText.GetTextFromSpeech(audioClip);

    string transcription = response.DisplayText;
}
```

How to use Audio Converter (Core)

1. **Add Two Audio Clips** is self explanatory. It takes in two values.

NOTE: I would not recommend saving this audio clip then playing it later because there is noise in the time between the two audio clips.

- a. The first value it takes in the audio clip that is going to be at the start of the new audio clip.
- b. The first value it takes in the audio clip that is going to be at the end of the new audio clip.
- c. And it returns the two audio clips merged into one.

```
thirdAudioClip = AudioConverter.AddTwoAudioClips(firstAudioClip, secondAudioClip);
```

2. **ConvertAudioClipToByteArray** is self explanatory. It takes in a value, and returns a value.

- a. The value that it takes is the audio clip that is going to be converted to a byte array.
- b. The value that it returns is a byte array which looks like this “byte[]”

```
byte[] audioDataAsByteArray = AudioConverter.ConvertAudioClipToByteArray(audioClip);
```

3. **ConvertAudioClipToWavFile** is self explanatory. It only takes in two values.

- a. The first value it takes in is the path that you want it to go to.

NOTE: You also need to pass in the name of the .wav file

- b. The second value it takes in is an Audio Clip. This is the audio clip that is saved.

```
AudioConverter.ConvertAudioClipToWavFile("c:\\<NAME OF AUDIO CLIP>", audioClip);
```

4. **ConvertByteArrayToAudioClip** this is also self explanatory. It takes in a value and it returns a value.

IMPORTANT: Not all audio data will properly convert, and might just be converted into noise.

- a. The value that it takes in is the byte array. That byte array will be converted to a unity audio clip.
- b. The value that it returns is the byte array as a unity audio clip.

```
audioClip = AudioConverter.ConvertByteArrayToAudioClip(audioDataAsByteArray);
```

5. **ConvertByteArrayToWavFile** is self explanatory. It only takes in two values.

- a. The first value it takes in is the path that you want it to go to.

NOTE: You also need to pass in the name of the .wav file

- b. The second value it takes in is an Audio Clip. This is the audio clip that is saved.

```
AudioConverter.ConvertByteArrayToWavFile(  
    "c:\\<NAME OF AUDIO CLIP>", audioDataAsByteArray);
```


How to use Microphone Utils (Core)

1. **GetAudioClip** uses the unity microphone class to get an audio clip. This function has two overrides, which mean you can pass in two different value combinations.

- a. All overrides return an audio clip, but you will have to await the response, which looks like this.

```
AudioClip audioClip = await MicrophoneUtils.GetAudioClip();
```

- b. The first override only takes in one value, for the time that you want it to record for in seconds.

```
AudioClip audioClip = await MicrophoneUtils.GetAudioClip(recordingTime);
```

- c. The second override takes in the same as the first but it also makes in the mic index. **NOTE: to find which mic is what then you can use the LogAllMicrophoneNames function.**

```
AudioClip audioClip = await MicrophoneUtils.GetAudioClip(recordingTime, micIndex);
```

2. **LogAllMicrophoneNames** This function will log all of the mic names, and the index that it is at.

```
MicrophoneUtils.LogAllMicrophoneNames();
```

3. **LogMicrophoneName** this takes in an int for the index and it will debug that one mic name.

```
MicrophoneUtils.LogMicrophoneName(micIndex);
```

4. To access these next Functions, and variables you will need to make a new instance of the “**MicrophoneUtils**” class. This class has two overrides

```
MicrophoneUtils microphoneUtils = new MicrophoneUtils();
```

- a. The first override takes in an **int** for the mic Index.
 - b. The second override takes in a **string** for the mic Name
5. **StartRecording** does not return any value. To get the value you will need to use the **StopRecording**.

NOTE: I would not recommend saving this audio clip then playing it later because there will be noise at some parts of the audio clip.

```
microphoneUtils.StartRecording();
```

6. **StopRecording** returns an Audio Clip.

NOTE: you will need an audio clip to stop so before you call this you should use the **StartRecording**.

```
AudioClip audioClip = microphoneUtils.StopRecording();
```

7. **recordedAudio** is the last saved audio clip recorded using this class.

```
microphoneUtils.recordedAudio
```

AzureSpeechEnums, and Jsons (Core)

AzureSpeechEnums is a static class that just has all the enum values. The enum values are.

NOTE: because it is impossible to make a enum with a “ ”, and the “-” characters for the values of the enum, I have switched the “ ” character to a “0”, and the “-” character to a “_”

1. **Regions** has all the different regions that your resources can be in.
2. **Languages** has the abbreviation of all the supported languages for the Text-To-Speech.
3. **DisplayNames** has all the display names of all the voices for azure speech.
NOTE: if this plugin is missing something or has something misspelled please let me know [here](#). I am a solo game developer so after I made the enums I did not go through them all and make sure everything is correct.
4. **LocalNames** has all the names in the native language of the speaker.
5. **ShortNames** has the shorthand version of all the names. It has the “abbreviation of the Native language”, then the “Display Name”, and last “Neural”.

AzureSpeechJsons is a static class that contains more classes that are used to hold the value of the Jsons.

1. **SpeechToTextResponse** is currently the only class. It holds the Json value of the [Speech to text](#) json.
 - a. **RecognitionStatus** has the status of the request. If this is not "Success" then please contact me [here](#) about the problem.

- b. **DisplayText** contains the words that azure thinks the user said.
- c. **Offset** is the time, in 100-nanosecond units, at which the recognized speech begins in the audio stream.
- d. **Duration** is the duration, in 100-nanosecond units, of the recognized speech in the audio stream.