Azure OpenAI speech to speech chat

Prerequisites

- Azure subscription - Create one for free
- Create a Microsoft Azure OpenAI Service resource in the Azure portal.
- Deploy a model in your Azure OpenAI resource. For more information about model deployment, see the Azure OpenAI resource deployment guide.
- Get the Azure OpenAI resource key and endpoint. After your Azure OpenAI resource is deployed, select **Go to resource** to view and manage keys. For more information about Azure AI services resources, see Get the keys for your resource.
- Create a Speech resource in the Azure portal.
- Get the Speech resource key and region. After your Speech resource is deployed, select **Go to resource** to view and manage keys. For more information about Azure AI services resources, see Get the keys for your resource.

Set up the environment

The Speech SDK is available as a NuGet package and implements .NET Standard 2.0. You install the Speech SDK later in this guide, but first check the SDK installation guide for any more requirements.

Set environment variables

This example requires environment variables named OPEN_AI_KEY, OPEN_AI_ENDPOINT, OPEN_AI_DEPLOYMENT_NAME, SPEECH_KEY, and SPEECH_REGION.

Your application must be authenticated to access Azure AI services resources. For production, use a secure way of storing and accessing your credentials. For example, after you get a key for your Speech resource, write it to a new environment variable on the local machine running the application.

Tip

Don't include the key directly in your code, and never post it publicly. See **Azure AI services security** for more authentication options like **Azure Key Vault**.

To set the environment variables, open a console window, and follow the instructions for your operating system and development environment.

- To set the OPEN_AI_KEY environment variable, replace your-openai-key with one of the keys for your resource.
- To set the OPEN_AI_ENDPOINT environment variable, replace your-openai-endpoint with one of the regions for your resource.
- To set the OPEN_AI_DEPLOYMENT_NAME environment variable, replace your-openai-deployment-name with one of the regions for your resource.
- To set the SPEECH_KEY environment variable, replace your-speech-key with one of the keys for your resource.
- To set the SPEECH_REGION environment variable, replace your-speech-region with one of the regions for your resource.

Windows

Console

setx OPEN_AI_KEY your-openai-key

setx OPEN_AI_ENDPOINT your-openai-endpoint

setx OPEN_AI_DEPLOYMENT_NAME your-openai-deployment-name

setx SPEECH_KEY your-speech-key

setx SPEECH_REGION your-speech-region

Note

If you only need to access the environment variable in the current running console, set the environment variable with set instead of setx.

After you add the environment variables, you might need to restart any running programs that need to read the environment variable, including the console window. For example, if Visual Studio is your editor, restart Visual Studio before running the example.

Recognize speech from a microphone

Follow these steps to create a new console application.

1. Open a command prompt window in the folder where you want the new project. Run this command to create a console application with the .NET CLI.

.NET CLI

dotnet new console

The command creates a *Program.cs* file in the project directory.

2. Install the Speech SDK in your new project with the .NET CLI.

.NET CLI

dotnet add package Microsoft.CognitiveServices.Speech

3. Install the Azure OpenAI SDK (prerelease) in your new project with the .NET CLI.

.NET CLI

dotnet add package Azure.AI.OpenAI --prerelease

4. Replace the contents of Program.cs with the following code.

C#

```
using System.Text;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Azure;
using Azure.AI.OpenAI;

// This example requires environment variables named "OPEN_AI_KEY",
"OPEN_AI_ENDPOINT" and "OPEN_AI_DEPLOYMENT_NAME"
// Your endpoint should look like the following
https://YOUR_OPEN_AI_RESOURCE_NAME.openai.azure.com/
string openAIKey = Environment.GetEnvironmentVariable("OPEN_AI_KEY") ??
                   throw new ArgumentException("Missing OPEN_AI_KEY");
string openAIEndpoint = Environment.GetEnvironmentVariable("OPEN_AI_ENDPOINT")
??
                        throw new ArgumentException("Missing
OPEN_AI_ENDPOINT");

// Enter the deployment name you chose when you deployed the model.
string engine = Environment.GetEnvironmentVariable("OPEN_AI_DEPLOYMENT_NAME")
??
                throw new ArgumentException("Missing
OPEN_AI_DEPLOYMENT_NAME");

// This example requires environment variables named "SPEECH_KEY" and
"SPEECH_REGION"
string speechKey = Environment.GetEnvironmentVariable("SPEECH_KEY") ??
                   throw new ArgumentException("Missing SPEECH_KEY");
string speechRegion = Environment.GetEnvironmentVariable("SPEECH_REGION") ??
                      throw new ArgumentException("Missing SPEECH_REGION");

// Sentence end symbols for splitting the response into sentences.
List<string> sentenceSaperators = new() { ".", "!", "?", ";", "。", "！", "？
", "；", "\n" };

try
{
```

```
    await ChatWithOpenAI();
}
catch (Exception ex)
{
    Console.WriteLine(ex);
}

// Prompts Azure OpenAI with a request and synthesizes the response.
async Task AskOpenAI(string prompt)
{
    object consoleLock = new();
    var speechConfig = SpeechConfig.FromSubscription(speechKey, speechRegion);

    // The language of the voice that speaks.
    speechConfig.SpeechSynthesisVoiceName = "en-US-JennyMultilingualNeural";
    var audioOutputConfig = AudioConfig.FromDefaultSpeakerOutput();
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig,
audioOutputConfig);
    speechSynthesizer.Synthesizing += (sender, args) =>
    {
        lock (consoleLock)
        {
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.Write($"[Audio]");
            Console.ResetColor();
        }
    };

    // Ask Azure OpenAI
    OpenAIClient client = new(new Uri(openAIEndpoint), new
AzureKeyCredential(openAIKey));
    var completionsOptions = new ChatCompletionsOptions()
    {
        DeploymentName = engine,
        Messages = { new ChatRequestUserMessage(prompt) },
        MaxTokens = 100,
    };
    var responseStream = await
client.GetChatCompletionsStreamingAsync(completionsOptions);

    StringBuilder gptBuffer = new();
    await foreach (var completionUpdate in responseStream)
    {
        var message = completionUpdate.ContentUpdate;
        if (string.IsNullOrEmpty(message))
        {
            continue;
        }

        lock (consoleLock)
        {
            Console.ForegroundColor = ConsoleColor.DarkBlue;
            Console.Write($"{message}");
            Console.ResetColor();
        }
```

```
        gptBuffer.Append(message);

        if (sentenceSaperators.Any(message.Contains))
        {
            var sentence = gptBuffer.ToString().Trim();
            if (!string.IsNullOrEmpty(sentence))
            {
                await speechSynthesizer.SpeakTextAsync(sentence);
                gptBuffer.Clear();
            }
        }
    }
}

// Continuously listens for speech input to recognize and send as text to
Azure OpenAI
async Task ChatWithOpenAI()
{
    // Should be the locale for the speaker's language.
    var speechConfig = SpeechConfig.FromSubscription(speechKey, speechRegion);
    speechConfig.SpeechRecognitionLanguage = "en-US";

    using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
    using var speechRecognizer = new SpeechRecognizer(speechConfig,
audioConfig);
    var conversationEnded = false;

    while (!conversationEnded)
    {
        Console.WriteLine("Azure OpenAI is listening. Say 'Stop' or press
Ctrl-Z to end the conversation.");

        // Get audio from the microphone and then send it to the TTS service.
        var speechRecognitionResult = await
speechRecognizer.RecognizeOnceAsync();

        switch (speechRecognitionResult.Reason)
        {
            case ResultReason.RecognizedSpeech:
                if (speechRecognitionResult.Text == "Stop.")
                {
                    Console.WriteLine("Conversation ended.");
                    conversationEnded = true;
                }
                else
                {
                    Console.WriteLine($"Recognized speech:
{speechRecognitionResult.Text}");
                    await AskOpenAI(speechRecognitionResult.Text);
                }

                break;
            case ResultReason.NoMatch:
                Console.WriteLine($"No speech could be recognized: ");
                break;
            case ResultReason.Canceled:
```

```
                var cancellationDetails =
CancellationDetails.FromResult(speechRecognitionResult);
                Console.WriteLine($"Speech Recognition canceled:
{cancellationDetails.Reason}");
                if (cancellationDetails.Reason == CancellationReason.Error)
                {
                    Console.WriteLine($"Error
details={cancellationDetails.ErrorDetails}");
                }

                break;
        }
    }
}
```

5. To increase or decrease the number of tokens returned by Azure OpenAI, change the MaxTokens property in the ChatCompletionsOptions class instance.
6. Run your new console application to start speech recognition from a microphone:

Console

dotnet run

Important

Make sure that you set the OPEN_AI_KEY, OPEN_AI_ENDPOINT, OPEN_AI_DEPLOYMENT_NAME, SPEECH_KEY and SPEECH_REGION **environment variables** as described. If you don't set these variables, the sample will fail with an error message.

Speak into your microphone when prompted. The console output includes the prompt for you to begin speaking, then your request as text, and then the response from Azure OpenAI as text. The response from Azure OpenAI should be converted from text to speech and then output to the default speaker.

Remarks

Here are some more considerations:

- To change the speech recognition language, replace en-US with another supported language. For example, es-ES for Spanish (Spain). The default language is en-US.
- To change the voice that you hear, replace en-US-JennyMultilingualNeural with another supported voice. If the voice doesn't speak the language of the text returned from Azure OpenAI, the Speech service doesn't output synthesized audio.

- To use a different model, replace gpt-35-turbo-instruct with the ID of another deployment. The deployment ID isn't necessarily the same as the model name. You named your deployment when you created it in Azure OpenAI Studio.
- Azure OpenAI also performs content moderation on the prompt inputs and generated outputs. The prompts or responses might be filtered if harmful content is detected.