How to synthesize speech from text

Article • 09/20/2024

Reference documentation | Package (npm) | Additional samples on GitHub | Library source code

In this how-to guide, you learn common design patterns for doing text to speech synthesis.

For more information about the following areas, see What is text to speech?

- Getting responses as in-memory streams.
- Customizing output sample rate and bit rate.
- Submitting synthesis requests by using Speech Synthesis Markup Language (SSML).
- Using neural voices.
- Subscribing to events and acting on results.

Select synthesis language and voice

The text to speech feature in the Speech service supports more than 400 voices and more than 140 languages and variants. You can get the full list or try them in the Voice Gallery .

Specify the language or voice of `SpeechConfig` to match your input text and use the specified voice:

JavaScript

```
function synthesizeSpeech() {
    const speechConfig = sdk.SpeechConfig.fromSubscription("YourSpeechKey",
"YourSpeechRegion");
    // Set either the `SpeechSynthesisVoiceName` or `SpeechSynthesisLanguage`.
    speechConfig.speechSynthesisLanguage = "en-US";
    speechConfig.speechSynthesisVoiceName = "en-US-AvaMultilingualNeural";
}

synthesizeSpeech();
```

All neural voices are multilingual and fluent in their own language and English. For example, if the input text in English is, "I'm excited to try text to speech," and you select `es-ES-ElviraNeural`, the text is spoken in English with a Spanish accent.

If the voice doesn't speak the language of the input text, the Speech service doesn't create synthesized audio. For a full list of supported neural voices, see Language and voice support for the Speech service.

> **Note**
>
> The default voice is the first voice returned per locale from the Voice List API.

The voice that speaks is determined in order of priority as follows:

- If you don't set `SpeechSynthesisVoiceName` or `SpeechSynthesisLanguage`, the default voice for `en-US` speaks.
- If you only set `SpeechSynthesisLanguage`, the default voice for the specified locale speaks.
- If both `SpeechSynthesisVoiceName` and `SpeechSynthesisLanguage` are set, the `SpeechSynthesisLanguage` setting is ignored. The voice that you specify by using `SpeechSynthesisVoiceName` speaks.
- If the voice element is set by using Speech Synthesis Markup Language (SSML), the `SpeechSynthesisVoiceName` and `SpeechSynthesisLanguage` settings are ignored.

In summary, the order of priority can be described as:

Expand table

SpeechSynthesisVoiceName	SpeechSynthesisLanguage	SSML	Outcome
✗	✗	✗	Default voice for `en-US` speaks
✗	✓	✗	Default voice for specified locale speaks.
✓	✓	✗	The voice that you specify by using `SpeechSynthesisVoiceName` speaks.
✓	✓	✓	The voice that you specify by using SSML speaks.

Synthesize text to speech

browserjs

To output synthesized speech to the current active output device such as a speaker, instantiate `AudioConfig` by using the `fromDefaultSpeakerOutput()` static function. Here's an example:

JavaScript

```
function synthesizeSpeech() {
    const speechConfig =
sdk.SpeechConfig.fromSubscription("YourSpeechKey", "YourSpeechRegion");
    const audioConfig = sdk.AudioConfig.fromDefaultSpeakerOutput();

    const speechSynthesizer = new SpeechSynthesizer(speechConfig, audio-
Config);
    speechSynthesizer.speakTextAsync(
        "I'm excited to try text to speech",
        result => {
            if (result) {
                speechSynthesizer.close();
                return result.audioData;
            }
        },
        error => {
            console.log(error);
            speechSynthesizer.close();
        });
}
```

When you run the program, synthesized audio is played from the speaker. This result is a good example of the most basic usage. Next, you can customize the output and handle the output response as an in-memory stream for working with custom scenarios.

Get a result as an in-memory stream

browserjs

You can use the resulting audio data as an in-memory stream rather than directly writing to a file. With in-memory stream, you can build custom behavior:

- Abstract the resulting byte array as a seekable stream for custom downstream services.
- Integrate the result with other APIs or services.
- Modify the audio data, write custom `.wav` headers, and do related tasks.

You can make this change to the previous example. Remove the `AudioConfig` block, because you manage the output behavior manually from this point onward for increased control. Then pass `null` for `AudioConfig` in the `SpeechSynthesizer` constructor.

> **Note**
>
> Passing `null` for `AudioConfig`, rather than omitting it as you did in the previous speaker output example, doesn't play the audio by default on the current active output device.

Save the result to a SpeechSynthesisResult variable. The `SpeechSynthesisResult.audioData` property returns an `ArrayBuffer` value of the output data, the default browser stream type. For server-side code, convert `ArrayBuffer` to a buffer stream.

The following code works for the client side:

JavaScript

```
function synthesizeSpeech() {
    const speechConfig =
sdk.SpeechConfig.fromSubscription("YourSpeechKey", "YourSpeechRegion");
    const speechSynthesizer = new sdk.SpeechSynthesizer(speechConfig);

    speechSynthesizer.speakTextAsync(
        "I'm excited to try text to speech",
        result => {
            speechSynthesizer.close();
            return result.audioData;
        },
        error => {
            console.log(error);
            speechSynthesizer.close();
        });
}
```

You can implement any custom behavior by using the resulting `ArrayBuffer` object. `ArrayBuffer` is a common type to receive in a browser and play from this format.

For any server-based code, if you need to work with the data as a stream, you need to convert the `ArrayBuffer` object into a stream:

JavaScript

```
function synthesizeSpeech() {
    const speechConfig =
sdk.SpeechConfig.fromSubscription("YourSpeechKey", "YourSpeechRegion");
    const speechSynthesizer = new sdk.SpeechSynthesizer(speechConfig);

    speechSynthesizer.speakTextAsync(
        "I'm excited to try text to speech",
        result => {
            const { audioData } = result;

            speechSynthesizer.close();

            // convert arrayBuffer to stream
            // return stream
            const bufferStream = new PassThrough();
            bufferStream.end(Buffer.from(audioData));
            return bufferStream;
        },
        error => {
            console.log(error);
            speechSynthesizer.close();
        });
}
```

Customize audio format

You can customize audio output attributes, including:

- Audio file type
- Sample rate
- Bit depth

To change the audio format, use the `speechSynthesisOutputFormat` property on the `SpeechConfig` object. This property expects an `enum` instance of type

SpeechSynthesisOutputFormat. Use the `enum` to select the output format. For available formats, see the list of audio formats.

There are various options for different file types, depending on your requirements. By definition, raw formats like `Raw24Khz16BitMonoPcm` don't include audio headers. Use raw formats only in one of these situations:

- You know that your downstream implementation can decode a raw bitstream.
- You plan to manually build headers based on factors like bit depth, sample rate, and number of channels.

This example specifies the high-fidelity RIFF format `Riff24Khz16BitMonoPcm` by setting `speechSynthesisOutputFormat` on the `SpeechConfig` object. Similar to the example in the previous section, get the audio `ArrayBuffer` data and interact with it.

JavaScript

```
function synthesizeSpeech() {
    const speechConfig = SpeechConfig.fromSubscription("YourSpeechKey",
"YourSpeechRegion");

    // Set the output format
    speechConfig.speechSynthesisOutputFormat =
sdk.SpeechSynthesisOutputFormat.Riff24Khz16BitMonoPcm;

    const speechSynthesizer = new sdk.SpeechSynthesizer(speechConfig, null);
    speechSynthesizer.speakTextAsync(
        "I'm excited to try text to speech",
        result => {
            // Interact with the audio ArrayBuffer data
            const audioData = result.audioData;
            console.log(`Audio data byte size: ${audioData.byteLength}.`)

            speechSynthesizer.close();
        },
        error => {
            console.log(error);
            speechSynthesizer.close();
        });
}
```

Use SSML to customize speech characteristics

You can use SSML to fine-tune the pitch, pronunciation, speaking rate, volume, and other aspects in the text to speech output by submitting your requests from an XML schema. This section shows an example of changing the voice. For more information, see Speech Synthesis Markup Language overview.

To start using SSML for customization, you make a minor change that switches the voice.

1. Create a new XML file for the SSML configuration in your root project directory.

   XML

   ```xml
   <speak version="1.0" xmlns="https://www.w3.org/2001/10/synthesis"
   xml:lang="en-US">
     <voice name="en-US-AvaMultilingualNeural">
       When you're on the freeway, it's a good idea to use a GPS.
     </voice>
   </speak>
   ```

   In this example, it's *ssml.xml*. The root element is always `<speak>`. Wrapping the text in a `<voice>` element allows you to change the voice by using the `name` parameter. For the full list of supported neural voices, see Supported languages.

2. Change the speech synthesis request to reference your XML file. The request is mostly the same, but instead of using the `speakTextAsync()` function, you use `speakSsmlAsync()`. This function expects an XML string. Create a function to load an XML file and return it as a string:

   JavaScript

   ```javascript
   function xmlToString(filePath) {
       const xml = readFileSync(filePath, "utf8");
       return xml;
   }
   ```

   For more information on `readFileSync`, see Node.js file system .

   The result object is exactly the same as previous examples:

   JavaScript

   ```javascript
   function synthesizeSpeech() {
       const speechConfig =
   sdk.SpeechConfig.fromSubscription("YourSpeechKey", "YourSpeechRegion");
   ```

```
    const speechSynthesizer = new sdk.SpeechSynthesizer(speechConfig,
null);

    const ssml = xmlToString("ssml.xml");
    speechSynthesizer.speakSsmlAsync(
        ssml,
        result => {
            if (result.errorDetails) {
                console.error(result.errorDetails);
            } else {
                console.log(JSON.stringify(result));
            }

            speechSynthesizer.close();
        },
        error => {
            console.log(error);
            speechSynthesizer.close();
        });
}
```

> **Note**
>
> To change the voice without using SSML, you can set the property on `SpeechConfig` by using `SpeechConfig.speechSynthesisVoiceName = "en-US-AvaMultilingualNeural";`.

Subscribe to synthesizer events

You might want more insights about the text to speech processing and results. For example, you might want to know when the synthesizer starts and stops, or you might want to know about other events encountered during synthesis.

While using the SpeechSynthesizer for text to speech, you can subscribe to the events in this table:

Expand table

Event	Description	Use case
`BookmarkReached`	Signals that a bookmark was reached. To trigger a bookmark reached event, a `bookmark` element is required in the SSML. This event reports the output	You can use the `bookmark` element to insert custom markers in SSML to get the offset of each marker in the audio stream. The `bookmark` element

Event	Description	Use case
	audio's elapsed time between the beginning of synthesis and the `bookmark` element. The event's `Text` property is the string value that you set in the bookmark's `mark` attribute. The `bookmark` elements aren't spoken.	can be used to reference a specific location in the text or tag sequence.
`SynthesisCanceled`	Signals that the speech synthesis was canceled.	You can confirm when synthesis is canceled.
`SynthesisCompleted`	Signals that speech synthesis is complete.	You can confirm when synthesis is complete.
`SynthesisStarted`	Signals that speech synthesis started.	You can confirm when synthesis started.
`Synthesizing`	Signals that speech synthesis is ongoing. This event fires each time the SDK receives an audio chunk from the Speech service.	You can confirm when synthesis is in progress.
`VisemeReceived`	Signals that a viseme event was received.	Visemes are often used to represent the key poses in observed speech. Key poses include the position of the lips, jaw, and tongue in producing a particular phoneme. You can use visemes to animate the face of a character as speech audio plays.
`WordBoundary`	Signals that a word boundary was received. This event is raised at the beginning of each new spoken word, punctuation, and sentence. The event reports the current word's time offset, in ticks, from the beginning of the output audio. This event also reports the character position in the input text or SSML immediately before the word that's about to be spoken.	This event is commonly used to get relative positions of the text and corresponding audio. You might want to know about a new word, and then take action based on the timing. For example, you can get information that can help you decide when and for how long to highlight words as they're spoken.

Note

Events are raised as the output audio data becomes available, which is faster than playback to an output device. The caller must appropriately synchronize streaming and real-time.

Here's an example that shows how to subscribe to events for speech synthesis.

> **Important**
>
> If you use an API key, store it securely somewhere else, such as in Azure Key Vault. Don't include the API key directly in your code, and never post it publicly.
>
> For more information about AI services security, see Authenticate requests to Azure AI services.

You can follow the instructions in the quickstart, but replace the contents of that *SpeechSynthesis.js* file with the following JavaScript code.

JavaScript

```
(function() {

    "use strict";

    var sdk = require("microsoft-cognitiveservices-speech-sdk");

    var audioFile = "YourAudioFile.wav";
    // This example requires environment variables named "SPEECH_KEY" and
"SPEECH_REGION"
    const speechConfig =
sdk.SpeechConfig.fromSubscription(process.env.SPEECH_KEY,
process.env.SPEECH_REGION);
    const audioConfig = sdk.AudioConfig.fromAudioFileOutput(audioFile);

    var speechSynthesisVoiceName  = "en-US-AvaMultilingualNeural";
    var ssml = `<speak version='1.0' xml:lang='en-US'
xmlns='http://www.w3.org/2001/10/synthesis'
xmlns:mstts='http://www.w3.org/2001/mstts'> \r\n \
        <voice name='${speechSynthesisVoiceName}'> \r\n \
            <mstts:viseme type='redlips_front'/> \r\n \
            The rainbow has seven colors: <bookmark
mark='colors_list_begin'/>Red, orange, yellow, green, blue, indigo, and violet.
<bookmark mark='colors_list_end'/>. \r\n \
        </voice> \r\n \
    </speak>`;
```

```
    // Required for WordBoundary event sentences.

speechConfig.setProperty(sdk.PropertyId.SpeechServiceResponse_RequestSentenceBo
undary, "true");

    // Create the speech speechSynthesizer.
    var speechSynthesizer = new sdk.SpeechSynthesizer(speechConfig, audioCon-
fig);

    speechSynthesizer.bookmarkReached = function (s, e) {
        var str = `BookmarkReached event: \
            \r\n\tAudioOffset: ${(e.audioOffset + 5000) / 10000}ms \
            \r\n\tText: \"${e.text}\".`;
        console.log(str);
    };

    speechSynthesizer.synthesisCanceled = function (s, e) {
        console.log("SynthesisCanceled event");
    };

    speechSynthesizer.synthesisCompleted = function (s, e) {
        var str = `SynthesisCompleted event: \
                    \r\n\tAudioData: ${e.result.audioData.byteLength} bytes \
                    \r\n\tAudioDuration: ${e.result.audioDuration}`;
        console.log(str);
    };

    speechSynthesizer.synthesisStarted = function (s, e) {
        console.log("SynthesisStarted event");
    };

    speechSynthesizer.synthesizing = function (s, e) {
        var str = `Synthesizing event: \
            \r\n\tAudioData: ${e.result.audioData.byteLength} bytes`;
        console.log(str);
    };

    speechSynthesizer.visemeReceived = function(s, e) {
        var str = `VisemeReceived event: \
            \r\n\tAudioOffset: ${(e.audioOffset + 5000) / 10000}ms \
            \r\n\tVisemeId: ${e.visemeId}`;
        console.log(str);
    };

    speechSynthesizer.wordBoundary = function (s, e) {
        // Word, Punctuation, or Sentence
        var str = `WordBoundary event: \
            \r\n\tBoundaryType: ${e.boundaryType} \
            \r\n\tAudioOffset: ${(e.audioOffset + 5000) / 10000}ms \
            \r\n\tDuration: ${e.duration} \
            \r\n\tText: \"${e.text}\" \
```

```
            \r\n\tTextOffset: ${e.textOffset} \
            \r\n\tWordLength: ${e.wordLength}`;
        console.log(str);
    };

    // Synthesize the SSML
    console.log(`SSML to synthesize: \r\n ${ssml}`)
    console.log(`Synthesize to: ${audioFile}`);
    speechSynthesizer.speakSsmlAsync(ssml,
        function (result) {
      if (result.reason === sdk.ResultReason.SynthesizingAudioCompleted) {
        console.log("SynthesizingAudioCompleted result");
      } else {
        console.error("Speech synthesis canceled, " + result.errorDetails +
            "\nDid you set the speech resource key and region values?");
      }
      speechSynthesizer.close();
      speechSynthesizer = null;
    },
        function (err) {
      console.trace("err - " + err);
      speechSynthesizer.close();
      speechSynthesizer = null;
    });
}());
```

You can find more text to speech samples at GitHub .

Run and use a container

Speech containers provide websocket-based query endpoint APIs that are accessed through the Speech SDK and Speech CLI. By default, the Speech SDK and Speech CLI use the public Speech service. To use the container, you need to change the initialization method. Use a container host URL instead of key and region.

For more information about containers, see Install and run Speech containers with Docker.

Next steps

- Try the text to speech quickstart
- Get started with custom neural voice
- Improve synthesis with SSML

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A