

How to do real-time synthesis for text to speech avatar

Article • 08/02/2024

In this how-to guide, you learn how to use text to speech avatar with real-time synthesis. The synthetic avatar video will be generated in almost real time after the system receives the text input.

Prerequisites


To get started, make sure you have the following prerequisites:

- **Azure subscription:** [Create one for free](#) .
- **Speech resource:** [Create a speech resource](#) in the Azure portal. Select "Standard S0" pricing tier if you want to create speech resource to access avatar.
- **Your speech resource key and region:** After your Speech resource is deployed, select **Go to resource** to view and manage keys.

Set up environment

For real-time avatar synthesis, you need to install the Speech SDK for JavaScript to use with a webpage. For the installation instructions, see [Install the Speech SDK](#).

Here's the compatibility of real-time avatar on different platforms and browsers:

 Expand table

Platform	Chrome	Microsoft Edge	Safari	Firefox	Opera
Windows	Y	Y	N/A	Y ¹	Y
Android	Y	Y	N/A	Y ¹²	N
iOS	Y	Y	Y	Y	Y
macOS	Y	Y	Y	Y ¹	Y

¹ It dosen't work with ICE server by Communication Service but works with Coturn.

² Background transparency doesn't work.

Select text to speech language and voice

The text to speech feature in the Speech service supports a broad portfolio of [languages and voices](#). 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. The following code snippet shows how this technique works:

JavaScript

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

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](#).

The default voice is the first voice returned per locale from the [voice list API](#). The order of priority for speaking is as follows:

- If you don't set `SpeechSynthesisVoiceName` or `SpeechSynthesisLanguage`, the default voice in `en-US` speaks.
- If you only set `SpeechSynthesisLanguage`, the default voice in 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.

Select avatar character and style

The supported avatar characters and styles can be found [here](#).

The following code snippet shows how to set avatar character and style:

JavaScript

```
const avatarConfig = new SpeechSDK.AvatarConfig(  
    "lisa", // Set avatar character here.  
    "casual-sitting", // Set avatar style here.  
);
```

Set up connection to real-time avatar

Real-time avatar uses WebRTC protocol to output the avatar video stream. You need to set up the connection with the avatar service through WebRTC peer connection.

First, you need to create a WebRTC peer connection object. WebRTC is a P2P protocol, which relies on ICE server for network relay. Speech service provides network relay function and exposes a REST API to issue the ICE server information. Therefore, we recommend you fetch the ICE server from the speech service. You can also choose to use your own ICE server.

Here is a sample request to fetch ICE information from the speech service endpoint:

HTTP

```
GET /cognitiveservices/avatar/relay/token/v1 HTTP/1.1  
  
Host: westus2.tts.speech.microsoft.com  
Ocp-Apim-Subscription-Key: YOUR_RESOURCE_KEY
```

The following code snippet shows how to create the WebRTC peer connection. The ICE server URL, ICE server username, and ICE server credential can all be fetched from the payload of above HTTP request.

JavaScript

```
// Create WebRTC peer connection  
peerConnection = new RTCPeerConnection({  
    iceServers: [{  
        urls: [ "Your ICE server URL" ],  
        username: "Your ICE server username",  
        credential: "Your ICE server credential"  
    }]  
})
```

📌 **Note**

The ICE server URL has two kinds: one with prefix `turn` (such as `turn:relay.communication.microsoft.com:3478`), and one with prefix `stun` (such as `stun:relay.communication.microsoft.com:3478`). In the previous example scenario, for `urls` you only need to include a URL with the `turn` prefix.

Secondly, you need to set up the video and audio player elements in the `ontrack` callback function of the peer connection. This callback is invoked twice during the connection, once for video track and once for audio track. You need to create both video and audio player elements in the callback function.

The following code snippet shows how to do so:

JavaScript

```
// Fetch WebRTC video/audio streams and mount them to HTML video/audio
player elements
peerConnection.ontrack = function (event) {
    if (event.track.kind === 'video') {
        const videoElement = document.createElement(event.track.kind)
        videoElement.id = 'videoPlayer'
        videoElement.srcObject = event.streams[0]
        videoElement.autoplay = true
    }

    if (event.track.kind === 'audio') {
        const audioElement = document.createElement(event.track.kind)
        audioElement.id = 'audioPlayer'
        audioElement.srcObject = event.streams[0]
        audioElement.autoplay = true
    }
}

// Offer to receive one video track, and one audio track
peerConnection.addTransceiver('video', { direction: 'sendrecv' })
peerConnection.addTransceiver('audio', { direction: 'sendrecv' })
```

Thirdly, you need to invoke the Speech SDK to create an avatar synthesizer and connect to the avatar service, with the peer connection as a parameter.

JavaScript

```
// Create avatar synthesizer
var avatarSynthesizer = new SpeechSDK.AvatarSynthesizer(speechConfig,
avatarConfig)

// Start avatar and establish WebRTC connection
avatarSynthesizer.startAvatarAsync(peerConnection).then(
    (r) => { console.log("Avatar started.") } )
```

```
).catch(  
  (error) => { console.log("Avatar failed to start. Error: " + error) }  
);
```

Our real-time API disconnects after 5 minutes of avatar's idle state. Even if the avatar is not idle and functioning normally, the real-time API will disconnect after a 10-minute connection. To ensure continuous operation of the real-time avatar for more than 10 minutes, you can enable auto-reconnect. For how to set up auto-reconnect, refer to this [sample code](#) (search "auto reconnect").

Synthesize talking avatar video from text input

After the above steps, you should see the avatar video being played in the web browser. The avatar is active, with eye blink and slight body movement, but not speaking yet. The avatar is waiting for text input to start speaking.

The following code snippet shows how to send text to the avatar synthesizer and let the avatar speak:

JavaScript

```
var spokenText = "I'm excited to try text to speech avatar."  
avatarSynthesizer.speakTextAsync(spokenText).then(  
  (result) => {  
    if (result.reason ===  
      SpeechSDK.ResultReason.SynthesizingAudioCompleted) {  
      console.log("Speech and avatar synthesized to video stream.")  
    } else {  
      console.log("Unable to speak. Result ID: " + result.resultId)  
      if (result.reason === SpeechSDK.ResultReason.Canceled) {  
        let cancellationDetails =  
          SpeechSDK.CancellationDetails.fromResult(result)  
        console.log(cancellationDetails.reason)  
        if (cancellationDetails.reason ===  
          SpeechSDK.CancellationReason.Error) {  
          console.log(cancellationDetails.errorDetails)  
        }  
      }  
    }  
  }).catch((error) => {  
    console.log(error)  
    avatarSynthesizer.close()  
  });
```

You can find end-to-end working samples on [GitHub](#) .

Edit background

The avatar real-time synthesis API currently doesn't support setting a background image/video and only supports setting a solid-color background, without transparent background support. However, there's an alternative way to implement background customization on the client side, following these guidelines:

- Set the background color to green (for ease of matting), which the avatar real-time synthesis API supports.
- Create a canvas element with the same size as the avatar video.
- Capture each frame of the avatar video and apply a pixel-by-pixel calculation to set the green pixel to transparent, and draw the recalculated frame to the canvas.
- Hide the original video.

With this approach, you can get an animated canvas that plays like a video, which has a transparent background. Here's the [sample code](#) to demonstrate such an approach.

After you have a transparent-background avatar, you can set the background to any image or video by placing the image or video behind the canvas.

Next steps

- [What is text to speech avatar](#)
- [Install the Speech SDK](#)

Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)