

Speech service documentation

Recognize speech, synthesize speech, get real-time translations, transcribe conversations, or integrate speech into your bot experiences.



OVERVIEW

[What is the Speech service?](#)



OVERVIEW

[About the Speech SDK](#)



WHAT'S NEW

[What's new](#)



REFERENCE

[Language support in the Speech service](#)

Speech to text

- [Get started with speech to text](#)
- [Improve accuracy with custom speech](#)
- [Use batch transcription](#)

[More >](#)

Text to speech

- [Get started with text to speech](#)
- [Improve synthesis with SSML](#)
- [Create a custom voice](#)
- [Make text to speech sound natural](#)

[More >](#)

Scenario deep-dives

- [Captioning with speech to text](#)
- [Call center overview](#)
- [Pronunciation assessment](#)

Tools

- [About Speech Studio](#)
- [About Speech CLI](#)

Speech translation

- [!\[\]\(467d80e979964f7f8c752fb22248b5b7_img.jpg\) About speech translation](#)
- [!\[\]\(b71552d33dbf62adf5e5199a70ee02bf_img.jpg\) Get started with speech translation](#)

Azure OpenAI

- [!\[\]\(31b03e46ee8a80a1f1467b8c03bd76e8_img.jpg\) What is the Whisper model?](#)
- [!\[\]\(7d9665ff04f9d2270c38081c6215a724_img.jpg\) What are OpenAI text to speech voices?](#)

Hosting

- [!\[\]\(815df092dd722ee9268ef8e6d0193e3a_img.jpg\) Speech service containers](#)
- [!\[\]\(c72edb9626cad660f3a9f5fb0f22a68c_img.jpg\) Speech service containers with Kubernetes](#)
- [!\[\]\(0c564128c6342bd2f601e97f4518828a_img.jpg\) Speech service containers on Azure Container Instances](#)

Migration

- [!\[\]\(c6a8736a601a632e2c96605cf66055ed_img.jpg\) Migrate to speech to text REST API version 2024-11-15](#)
- [!\[\]\(64ef2b19d70b31fbbfce0e0e2aa3d7b4_img.jpg\) Migrate to Batch synthesis REST API](#)

Software development kits (SDKs)

Get started with the Speech SDK in your favorite programming language.



C#



C++



Java



JavaScript

iOS [Objective-C and Swift](#)



Python

REST APIs

- [Speech to text](#)
- [Batch transcription](#)
- [Text to speech](#)

Resources

- [Pricing ↗](#)
- [Regions](#)
- [Learn](#)

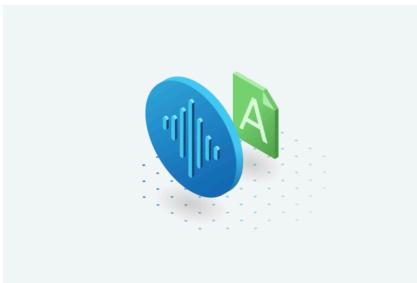
Support

- [Support and help](#)
- [GitHub issues ↗](#)

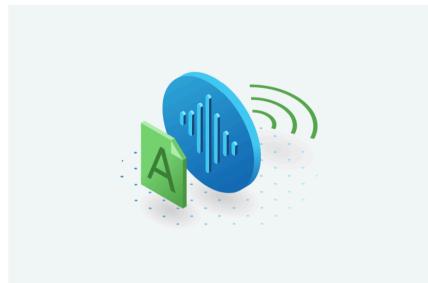
Conversation transcription

What is the Speech service?

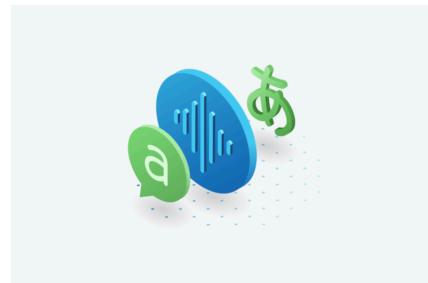
The Speech service provides speech to text and text to speech capabilities with a [Speech resource](#). You can transcribe speech to text with high accuracy, produce natural-sounding text to speech voices, translate spoken audio, and conduct live AI voice conversations.



Convert speech to text



Synthesize text to speech



Translate speech

Create custom voices, add specific words to your base vocabulary, or build your own models. Run Speech anywhere, in the cloud or at the edge in containers. It's easy to speech enable your applications, tools, and devices with the [Speech CLI](#), [Speech SDK](#), and [REST APIs](#).

Speech is available for many [languages](#), [regions](#), and [price points](#) ↗.

Speech scenarios

Common scenarios for speech include:

- [Captioning](#): Learn how to synchronize captions with your input audio, apply profanity filters, get partial results, apply customizations, and identify spoken languages for multilingual scenarios.
- [Audio Content Creation](#): You can use neural voices to make interactions with chatbots and voice agents more natural and engaging, convert digital texts such as e-books into audiobooks and enhance in-car navigation systems.
- [Call Center](#): Transcribe calls in real-time or process a batch of calls, redact personally identifying information, and extract insights such as sentiment to help with your call center use case.
- [Language learning](#): Provide pronunciation assessment feedback to language learners, support real-time transcription for remote learning conversations, and read aloud teaching materials with neural voices.
- [Voice live](#): Create natural, human like conversational interfaces for applications and experiences. The voice live feature provides fast, reliable interaction between a human and an agent implementation.

Microsoft uses Speech for many scenarios, such as captioning in Teams, dictation in Office 365, and Read Aloud in the Microsoft Edge browser.



Sales /
Support

Live captions

Subtitles/
Translation

Read aloud

Dictation

Commute

Cortana



AZURE SPEECH

Speech capabilities

These sections summarize Speech features with links for more information.

Speech to text

Use [speech to text](#) to convert audio into text - whether through [real-time transcription](#) for streaming audio, [fast transcription](#) for pre-recorded audio files, or [batch transcription](#) for processing large volumes of audio asynchronously.

The base model might not be sufficient if the audio contains ambient noise or includes numerous industry and domain-specific jargon. In these cases, you can create and train [custom speech models](#) with acoustic, language, and pronunciation data. Custom speech models are private and can offer a competitive advantage.

Text to speech

With [text to speech](#), you can convert input text into human like synthesized speech. Use neural voices, which are human like voices powered by deep neural networks. Use the [Speech Synthesis Markup Language \(SSML\)](#) to fine-tune the pitch, pronunciation, speaking rate, volume, and more.

- Standard voice: Highly natural out-of-the-box voices. Check the standard voice samples the [Voice Gallery](#) and determine the right voice for your business needs.
- Custom voice: Besides the standard voices that come out of the box, you can also create a [custom voice](#) that is recognizable and unique to your brand or product. Custom voices are private and can offer a competitive advantage. Check the custom voice samples [here](#).

Speech translation

[Speech translation](#) enables real-time, multilingual translation of speech to your applications, tools, and devices. Use this feature for speech to speech and speech to text translation.

LLM speech (preview)

[LLM speech](#) currently supports the following speech tasks:

- `transcribe`: Convert pre-recorded audio into text.
- `translate`: Convert pre-recorded audio into text in a specified target language.

LLM speech uses a large-language-model-enhanced speech model that delivers improved quality, deep contextual understanding, multilingual support, and prompt-tuning capabilities. It shares the same ultra-fast inference performance as fast transcription, making it ideal for use cases such as generating captions and subtitles from audio files, summarizing meeting notes, assisting call center agents, transcribing voicemails, and more.

Language identification

[Language identification](#) is used to identify languages spoken in audio when compared against a list of [supported languages](#). Use language identification by itself, with speech to text recognition, or with speech translation.

Pronunciation assessment

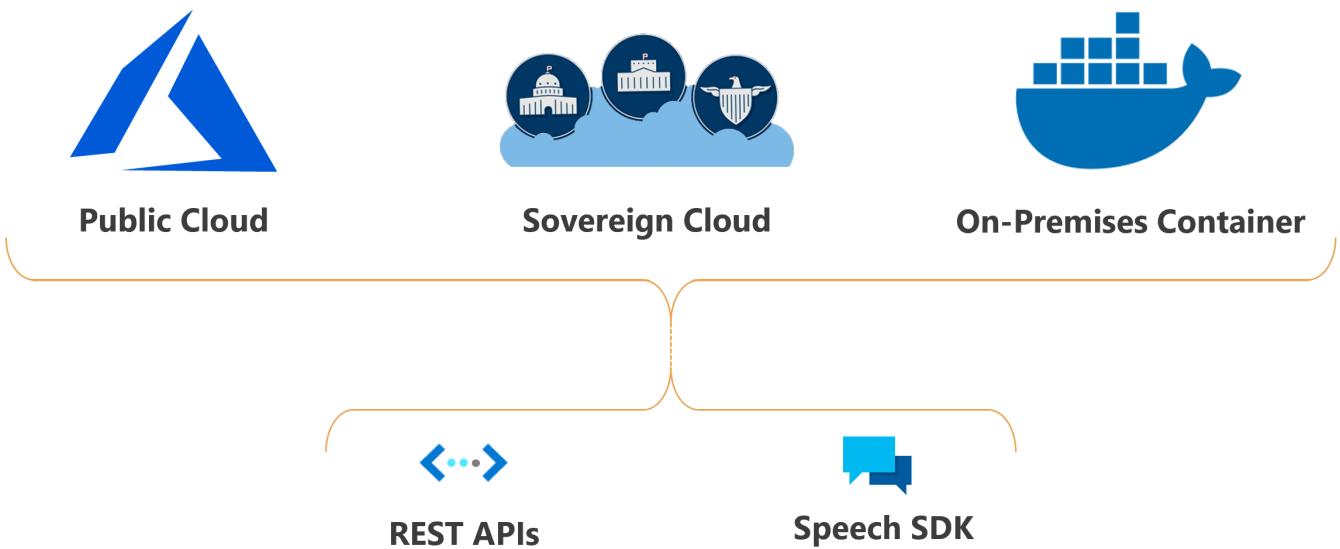
[Pronunciation assessment](#) evaluates speech pronunciation and gives speakers feedback on the accuracy and fluency of spoken audio. With pronunciation assessment, language learners can practice, get instant feedback, and improve their pronunciation so that they can speak and present with confidence.

Delivery and presence

You can deploy Azure AI Speech features in the cloud or on-premises.

With [containers](#), you can bring the service closer to your data for compliance, security, or other operational reasons.

Speech service deployment in sovereign clouds is available for some government entities and their partners. For example, the Azure Government cloud is available to US government entities and their partners. Microsoft Azure operated by 21Vianet cloud is available to organizations with a business presence in China. For more information, see [sovereign clouds](#).



Use Speech in your application

The [Speech Studio](#) is a set of UI-based tools for building and integrating features from Azure AI Speech service in your applications. You create projects in Speech Studio by using a no-code approach, and then reference those assets in your applications by using the [Speech SDK](#), the [Speech CLI](#), or the REST APIs.

The [Speech CLI](#) is a command-line tool for using Speech service without having to write any code. Most features in the Speech SDK are available in the Speech CLI, and some advanced features and customizations are simplified in the Speech CLI.

The [Speech SDK](#) exposes many of the Speech service capabilities you can use to develop speech-enabled applications. The Speech SDK is available in many programming languages and across all platforms.

In some cases, you can't or shouldn't use the [Speech SDK](#). In those cases, you can use REST APIs to access the Speech service. For example, use REST APIs for [batch transcription](#).

Get started

We offer quickstarts in many popular programming languages. Each quickstart is designed to teach you basic design patterns and have you running code in less than 10 minutes. See the following list for the quickstart for each feature:

- [Speech to text quickstart](#)
- [Text to speech quickstart](#)
- [Speech translation quickstart](#)

Code samples

Sample code for the Speech service is available on GitHub. These samples cover common scenarios like reading audio from a file or stream, continuous and single-shot recognition, and working with custom models. Use these links to view SDK and REST samples:

- [Speech to text, text to speech, and speech translation samples \(SDK\) ↗](#)
- [Batch transcription samples \(REST\) ↗](#)
- [Text to speech samples \(REST\) ↗](#)

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who are affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

Speech to text

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)
- [Integration and responsible use](#)
- [Data, privacy, and security](#)

Pronunciation Assessment

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)

Custom voice

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)
- [Limited access](#)
- [Responsible deployment of synthetic speech](#)
- [Disclosure of voice talent](#)
- [Disclosure of design guidelines](#)
- [Disclosure of design patterns](#)
- [Code of conduct](#)
- [Data, privacy, and security](#)

Next steps

- Get started with speech to text
 - Get started with text to speech
-

Last updated on 11/05/2025

What's new in Azure AI Speech?

Azure AI Speech is updated on an ongoing basis. To stay up-to-date with recent developments, this article provides you with information about new releases and features.

Recent highlights

- LLM speech API is public preview now. It uses a large-language-model-enhanced speech model that delivers improved quality, deep contextual understanding, multilingual support, and prompt-tuning capabilities. It currently supports the following speech tasks:
 - `transcribe`: Convert pre-recorded audio into text.
 - `translate`: Convert pre-recorded audio into text in a specified target language.

For more information, see [LLM speech](#).

- Voice live API is generally available now. Transform conversations into seamless experiences with Voice live API—the all-in-one solution that combines speech recognition, generative AI, and text-to-speech into a single low-latency interface for building intelligent voice agents. For more information, see [Voice live](#).
- To transcribe multi-lingual contents continuously and accurately in an audio file, now you can use the latest multi-lingual model without specifying the locale codes via fast transcription API. For more information, see [multi-lingual transcription in fast transcription](#).
- Fast transcription is now generally available. It can transcribe audio much faster than the actual audio duration. For more information, see the [fast transcription API guide](#).
- Azure AI Speech Toolkit extension is now available for Visual Studio Code users. It contains a list of speech quick-starts and scenario samples that can be easily built and run with simple clicks. For more information, see [Azure AI Speech Toolkit in Visual Studio Code Marketplace](#).
- Azure AI speech high definition (HD) voices are available in public preview. The HD voices can understand the content, automatically detect emotions in the input text, and adjust the speaking tone in real-time to match the sentiment. For more information, see [What are Azure AI Speech high definition \(HD\) voices?](#).
- Video translation is now available in the Azure AI Speech service. For more information, see [What is video translation?](#).
- Speech services are now available in three new regions: *Canada East*, *Italy North* and *UK West*. For more information, see [Speech service supported regions](#).

Release notes

Choose a service or resource

SDK

Important

Content assessment (preview) via the Speech SDK is being retired in July 2025.

Instead, you can use Azure OpenAI models to get content assessment results as described in the [content assessment documentation](#).

Speech SDK 1.46: 2025-September release

New features:

- Added support for speech start event sensitivity with `Speech_StartEventSensitivity` property.
- Deprecated `SpeechServiceConnection_EndSilenceTimeoutMs` property.
- Retired the content assessment functionality in pronunciation assessment.
- Updated Android OpenSSL to 3.0.17.
- Added a size limit to the telemetry queue in order to prevent growth in memory usage.
- Added a timeout guard for cache reading in TTS in order to prevents potential IO hangs.
- Added configurable properties to control URL redirection cache behavior.
- [C#] Added support for **EventSource** based logging.
- [Python] Added support for *AzureKeyCredential* authentication.

Bug fixes

- Fixed a ja-JP pronunciation issue in embedded text-to-speech.
- Fixed a notable increase in memory usage over a long period of time in embedded speech-to-text.
- Fixed crash caused by race condition during timeout stopping recognition.
- [JavaScript] Fixed an issue where `fromHost` did not work with Docker container service.

Samples

- Updated samples to demonstrate the usage of `AzureKeyCredential` and `AAD token credential` authentications.
- [JavaScript, Python] Updated samples to use `fromEndpoint`.

Speech SDK 1.45: 2025-July release

New features:

- Added support for setting the phrase list grammar weight. (Currently only effects embedded scenarios)
- Added more specific file opening error codes.
- Updated Unicode path support so that SDK Windows DLLs can be located under non-ASCII paths.
- Updated descriptions of segmentation strategy properties to align with the service logic.
- [C#, Java] Added support for authentication using `ApiKeyCredential`.

Bug fixes

- Fixed the Microsoft Audio Stack (MAS) initialization error about microphone geometry in certain regions.
- Fixed profanity settings not working in speech translation (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2856>).
- Fixed a crash in intent recognition pattern matching with Japanese language.
- Fixed custom domain resolution not working with Node.js v22 or newer.

Samples

- [Java] Added sample code to demonstrate AAD token credential authentication.

Speech SDK 1.44.1: Patch release

SDK version 1.44.1 is being released for JavaScript only with 4 bug fixes:

Bug fixes

- Fixed an out of range exception when only one segmentation control parameter was provided.

- `enableDictation` was not correctly passed to the Speech Service.
- `ConversationTranscriber` did not use the correct URL path when created using the `fromEndpoint` method.
- Fixed error when data is pushed to an input stream after it is detached.

Speech SDK 1.44: 2025-May release

Important

Support for target platforms is changing:

- The minimum supported Android version is now Android 8.0 (API level 26).
- The publishing of Speech SDK Unity packages is suspended after this release.

New features:

- Added support for Android 16 KB memory page sizes.
- Reduced the latency of `SpeechStartDetected` events in embedded speech recognition.
- [C++, Python] Added a method to get the available size of `AudioDataStream`.
- [C++, Python] Added support for custom lexicon URLs and preferred locales in speech synthesis requests.
- [Java, Python] Added support for Microsoft Entra token-based authentication with automatic token refresh.
- [Go] Added support for Conversation Transcription.

Bug fixes

- Fixed translation speech synthesis not working when source language detection was used.
- Fixed file paths with non-ASCII characters not working for embedded speech models, KWS models, or log files (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2288>).
- Fixed a `NoMatch` loop in embedded speech recognition in certain conditions.
- Fixed the destructor of native objects being blocked due to recognition not marked as stopped when events are disconnected.
- Fixed `IntentRecognizer` pattern matching not working correctly with multi-byte characters in certain conditions.
- Calling `Close()` on a `Connection` object wasn't synchronous.

- Fixed a race condition in connection deallocation that could lead to a crash.
- [macOS] Fixed "Info:" messages appearing on the console (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2610>).

Samples

- [Python] Added sample code for `recognizer` using Microsoft Entra token credentials.

Speech SDK for JavaScript

New features:

- Updated development dependency: TypeScript 3.5.3 → 4.5
- Updated TranslationRecognizer to use V2 endpoints by default.
- Updated SpeechRecongizer to use V2 endpoints.
 - This results in no longer receiving NoMatch results.
- Added support for Microsoft Entra token-based authentication for Speech Recognition and Translation.
- Updated FromEndpoint API to be the recommended method for constructing a SpeechConfig for most scenarios.
 - Applies to using:
 - SpeechRecognizer
 - TranslationRecognizer (via SpeechTranslationConfig)
 - ConversationTranscriber
 - SpeechSynthesizer
 - You can now use the endpoint from the Azure portal for Speech and Azure AI Foundry resources to construct a SpeechConfig object.
 - All other methods to construct a SpeechConfig continue to function and are supported.

Bug fixes

- Fixed an infinite connection retry loop on unsupported connection closing codes (<https://github.com/microsoft/cognitive-services-speech-sdk-js/issues/896>).

Speech SDK 1.43: 2025-March release

! Note

Ubuntu 20.04 "standard security maintenance" expires in April 2025 [↗](#) and will no longer be available as ADO Build agents. Future Speech SDK releases will require Ubuntu 22.04 LTS (instead of Ubuntu 20.04) as the minimum supported version.

New features:

- Updated FromEndpoint API to be the recommended method for constructing a SpeechConfig for most scenarios.
 - Applies to using:
 - SpeechRecognizer
 - TranslationRecognizer (via SpeechTranslationConfig)
 - ConversationTranscriber
 - SpeechSynthesizer In all programming languages except JavaScript.
 - You can now use the Endpoint from the Azure Portal for Speech and Cognitive Services resources to construct a SpeechConfig object.
 - All other methods to construct a SpeechConfig continue to function and are supported.
- Updated TranslationRecognizer to use V2 endpoints by default.
 - This moves control parameters from the URL to in-channel messages when using a V2 endpoint.
 - Behavior change: The default language returned for "zh" is now "zh-CN" instead of "zh-hans"
- Added property ids for SpeechSynthesis_FrameTimeoutInterval and SpeechSynthesis_RtfTimeoutThreshold.
- Optimized the number of times the SDK reconnects for long running recognitions.
- [C++, Python] Added support for specifying the style and temperature in text streaming requests.
- [C#] Added support for automatic AAD token refresh when using FromEndpoint to construct a config object.
 - This adds a dependency from the Speech SDK to the Azure.Core nuget package.
 - The Speech SDK can now accept TokenCredential derived objects for authentication when using:
 - SpeechRecognizer
 - TranslationRecognizer
 - ConversationTranscriber
- [Objective-C] Updated SPXTranslationRecognizer to support source language auto detection from open range.
- [Objective-C , Python] Added diagnostics APIs EventLogger, FileLogger, and MemoryLogger.
- [Go]: Added TranslationRecognizer support

Bug fixes

- Fixed OpenSSL 3 support on Linux arm32 (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2736>).
- Fixed the missing status field in the speech synthesis voice list (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2771>).
- Fixed IntentRecognizer pattern matching Japanese language parser not correctly identifying integer characters.
- Fixed a potential issue with duplicate results from embedded speech recognition.
- [Java] Fixed empty participants in ConversationParticipantsChangedEventArgs on Android 12 and newer (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2687>).

Samples

- [C++] Added a sample for standalone intent recognition using pattern matching.
 - With the retirement of the LUIS service in October 2025 the Speech SDK will also be retiring the IntentRecognizer object family.
 - Before that, we wanted to share the implementation for pattern matching.
- [C++, C#, Java, Python] Updated most samples to use FromEndpoint API instead of FromSubscription.
- [C#] Added a scenario sample for a multi-tier speech recognition application.
 - Demonstrates a methodology for audio replay and reconnection from an edge device to a middle tier service that then forwards audio to the Speech Service via the Speech SDK
- [C#] Updated samples to use automatic AAD token refresh.
- [Python] Added samples for new diagnostics APIs.
- [Unity] Added instructions for installing the new Azure.Core dependency.

Speech SDK 1.42.0: 2024-December release

New features

- Java: Added Diagnostics logging APIs using classes of FileLogger, MemoryLogger, EventLogger and SpxTrace.
- Support sending JSON property "details" of meeting participant to service
- Go: Added public property id SpeechServiceConnection_ProxyHostBypass to specify hosts for which proxy is not used.
- JavaScript, Go: Added public property id Speech_SegmentationStrategy to determine when a spoken phrase has ended and a final recognized result should be

generated(including semantic segmentation)

- JavaScript, Go: Added public property id Speech_SegmentationMaximumTimeMs determine the end of a spoken phrase based on time in Java, Python, C#, C++

Bug fixes

- Fixed embedded TTS voice (re)loaded for every synthesis if the voice name is not set.
- Fixed offset calculation problems when using MeetingTranscriber in some scenarios.
- Fixed potential deadlock when registering multiple Diagnostic event listeners in parallel.
- (JavaScript) Fixed possible lost NoMatch results when at the end of audio. This fix also aligns the behavior at the end of speech with the other SDK languages and may result in some empty events no longer being raised.
- (JavaScript) Fixup offsets in result JSON to align with the offset on result objects. Previously only the result object's offset property was fixed up to account for service reconnections.
- Go language: Fixed a compilation error <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2639>
- Fixed result offsets in meeting transcription when a reconnection to the service occurs.
- Fixed a deadlock in logging.

Samples

- Updated C# samples to use .NET 8.0.
- Java sample use Diagnostics logging API showing usage of the new Diagnostics Logging classes.

2024-November release

Azure AI Speech Toolkit extension for Visual Studio Code

Azure AI Speech Toolkit extension is now available for Visual Studio Code users. It contains a list of speech quick-starts and scenario samples that can be easily built and run with simple clicks. For more information, see [Azure AI Speech Toolkit in Visual Studio Code Marketplace](#).

Text to speech avatar code samples

We added text to speech avatar code samples for [Android](#) and [iOS](#). These samples demonstrate how to use real-time text to speech avatars in your mobile applications.

Speech SDK 1.41.1: 2024-October release

New Features

- Added support for Amazon Linux 2023 and Azure Linux 3.0.
- Added public property id SpeechServiceConnection_ProxyHostBypass to specify hosts for which proxy is not used.
- Added properties to control new phrase segmentation strategies.

Bug Fixes

- Fixed incomplete support for keyword recognition Advanced models produced after August 2024.
 - <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2564>
 - <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2571>
 - <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2590>
 - Note that with Swift on iOS your project must use either MicrosoftCognitiveServicesSpeech-EmbeddedXCFramework-1.41.1.zip (from <https://aka.ms/csspeech/iosbinaryembedded>) or the MicrosoftCognitiveServicesSpeechEmbedded-iOS pod that include the Advanced model support.
- Fixed a memory leak in C# related to string usage.
- Fixed not being able to get SPXAutoDetectSourceLanguageResult from SPXConversationTranscriptionResult in Objective-C and Swift.
- Fixed an occasional crash when using the Microsoft Audio Stack in recognition.
- Fixed type hints in Python. <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2539>
- Fixed not being able to fetch the list of TTS voices when using a custom endpoint.
- Fixed embedded TTS re-initializing for every speak request when the voice is specified by a short name.
- Fixed the API reference documentation for the max duration of RecognizeOnce audio.
- Fixed error handling arbitrary sampling rates in JavaScript
 - Thanks to [rseanhall](#) for this contribution.
- Fixed error calculating the audio offset in JavaScript
 - Thanks to [motamed](#) for this contribution.

Breaking Changes

- Keyword recognition support on Windows ARM 32-bit has been removed due to the required ONNX runtime not available for this platform.

Speech SDK 1.40: 2024-August release

ⓘ Note

Speech SDK version 1.39.0 was an internal release and isn't missing.

New features

- Added support for streaming of `G.722` compressed audio in speech recognition.
- Added support for pitch, rate, and volume setting in input text streaming in speech synthesis.
- Added support for personal voice input text streaming by introducing `PersonalVoiceSynthesisRequest` in speech synthesis. This API is in preview and subject to change in future versions.
- Added support for diarization of intermediate results when `ConversationTranscriber` is used.
- Removed CentOS/RHEL 7 support due to [CentOS 7 EOL ↗](#) and [the end of RHEL 7 Maintenance Support 2 ↗](#).
- Use of embedded speech models now requires a model license instead of a model key. If you're an existing embedded speech customer and want to upgrade, please contact your support person at Microsoft for details on model updates.

Bug fixes

- Built Speech SDK binaries for Windows with the `_DISABLE_CONSTEXPR_MUTEX_CONSTRUCTOR` flag as mitigation for the Visual C++ runtime issue [Access violation with std::mutex::lock after upgrading to VS 2022 version 17.10.0 - Developer Community \(visualstudio.com\) ↗](#). Windows C++ applications using the Speech SDK might need to apply the same build configuration flag if their code uses `std::mutex` (see details in the linked issue).
- Fixed OpenSSL 3.x detection not working on Linux arm64 ([https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2420 ↗](https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2420)).
- Fixed the issue that when deploying a UWP app, libraries, and model from MAS NuGet package wouldn't get copied to the deployment location.

- Fixed a content provider conflict in Android packages (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2463>).
- Fixed postprocessing options not applying to intermediate speech recognition results.
- Fixed .NET 8 warning about distribution specific runtime identifiers (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2244>).

Samples

- Updated embedded speech samples to use a model license instead of a key.

Speech SDK 1.38.0: 2024-June release

New features

- Upgrade Speech SDK Linux platform requirements:
 - The new minimum baseline is Ubuntu 20.04 LTS or compatible with `glibc` 2.31 or newer.
 - Binaries for Linux x86 are removed in accordance with Ubuntu 20.04 platform support.
 - **Note that RHEL/CentOS 7** remain supported until June 30 (the end of CentOS 7 and the end of RHEL 7 Maintenance Support 2). Binaries for them will be removed in the Speech SDK 1.39.0 release.
- Add support for OpenSSL 3 on Linux.
- Add support for g722-16khz-64kbps audio output format with speech synthesizer.
- Add support for sending messages through a connection object with speech synthesizer.
- Add Start/StopKeywordRecognition APIs in Objective-C and Swift.
- Add API for selecting a custom translation model category.
- Update GStreamer usage with speech synthesizer.

Bug fixes

- Fix "Websocket message size can't exceed 65,536 bytes" error during Start/StopKeywordRecognition.
- Fix a Python segmentation fault during speech synthesis.

Samples

- Update C# samples to use .NET 6.0 by default.

Speech SDK 1.37.0: 2024-April release

New features

- Add support for input text streaming in speech synthesis.
- Change the default speech synthesis voice to en-US-AvaMultilingualNeural.
- Update Android builds to use OpenSSL 3.x.

Bug fixes

- Fix occasional JVM crashes during SpeechRecognizer dispose when using MAS. (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2125>)
- Improve detection of default audio devices on Linux. (<https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2292>)

Samples

- Updated for new features.

Speech SDK 1.36.0: 2024-March release

New features

- Add support for language identification in multi-lingual translation on v2 endpoints using AutoDetectSourceLanguageConfig::FromOpenRange().

Bug fixes

- Fix SynthesisCanceled event not fired if stop is called during SynthesisStarted event.
- Fix a noise issue in embedded speech synthesis.
- Fix a crash in embedded speech recognition when running multiple recognizers in parallel.
- Fix the phrase detection mode setting on v1/v2 endpoints.
- Fixes to various issues with Microsoft Audio Stack.

Samples

- Updates for new features.

Speech SDK 1.35.0: February 2024 release

New features

- Change the default text to speech voice from en-US-JennyMultilingualNeural to en-US-AvaNeural.
- Support word-level detail in embedded speech translation results using the detailed output format.

Bug fixes

- Fix the AudioDataStream position getter API in Python.
- Fix speech translation using v2 endpoints without language detection.
- Fix a random crash and duplicate word boundary events in embedded text to speech.
- Return a correct cancellation error code for an internal server error on WebSocket connections.
- Fix the failure to load FPIEProcessor.dll library when MAS is used with C#.

Samples

- Minor formatting updates for Embedded recognition samples.

Speech SDK 1.34.1: January 2024 release

Breaking changes

- Bug fixes only

New features

- Bug fixes only

Bug fixes

- Fix regression introduced in 1.34.0 where service endpoint url was constructed with bad locale info for users in several China regions.

Speech SDK 1.34.0: November 2023 release

Breaking changes

- `SpeechRecognizer` is updated to use a new endpoint by default (that is, when not explicitly specifying a URL) which no longer supports query string parameters for most of the properties. Instead of setting query string parameters directly with `ServicePropertyChannel.UriQueryParameter`, please use the corresponding API functions.

New features

- Compatibility with .NET 8 (Fix for <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2170> except for warning about centos7-x64)
- Support for embedded speech performance metrics which can be used to evaluate the capability of a device to run embedded speech.
- Support for source language identification in embedded multi-lingual translation.
- Support for embedded speech-to-text, text to speech and translation for iOS and Swift/Objective-C released in preview.
- Embedded support is provided in MicrosoftCognitiveServicesSpeechEmbedded-iOS Cocoapod.

Bug fixes

- Fix for iOS SDK x2 times binary size growth · Issue #2113 · Azure-Samples/cognitive-services-speech-sdk (github.com)
- Fix for Unable to get word level time stamps from Azure speech to text API · Issue #2156 · Azure-Samples/cognitive-services-speech-sdk (github.com)
- Fix for DialogServiceConnector destruction phase to disconnect events correctly. This was causing crashes occasionally.
- Fix for exception during creation of a recognizer when MAS is used.
- FPIEProcessor.dll from Microsoft.CognitiveServices.Speech.Extension.MAS NuGet package for Windows UWP x64 and Arm64 had dependency on VC runtime libraries for native C++. The issue has been rectified by updating the dependency to correct VC runtime libraries (for UWP).
- Fix for [MAS] Recurrent calls to recognizeOnceAsync lead to SPXERR_ALREADY_INITIALIZED when using MAS · Issue #2124 · Azure-Samples/cognitive-services-speech-sdk (github.com)
- Fix for embedded speech recognition crash when phrase lists are used.

Samples

- Embedded iOS samples for speech-to-text, text to speech and translation.

Speech CLI 1.34.0: November 2023 release

New features

- Support word boundary events output when synthesizing speech.

Bug fixes

- Updated JMESPath dependency to the latest release, improves string evaluations

Speech SDK 1.33.0: October 2023 release

Breaking change notice

- The new NuGet package added for Microsoft Audio Stack (MAS) is now required to be included by applications that are using MAS in their package configuration files.

New features

- Added the new NuGet package Microsoft.CognitiveServices.Speech.Extension.MAS.nupkg, which provides improved echo cancellation performance when using Microsoft Audio Stack
- Pronunciation Assessment: added support for prosody and content evaluation, which can assess the spoken speech in terms of prosody, vocabulary, grammar, and topic.

Bug fixes

- Fixed keyword recognition result offsets so that they correctly match the input audio stream since the beginning. The fix applies to both stand-alone keyword recognition and keyword-triggered speech recognition.
- Fixed Synthesizer stopSpeaking doesn't return immediately [SPXSpeechSynthesizer stopSpeaking\(\) method can't return immediately on iOS 17 - Issue #2081](#)
- Fixed Mac catalyst import issue on Swift module Support for mac catalyst with apple silicon. [Issue #1948](#)
- JS: AudioWorkletNode module loads [now uses a trusted URL](#), with fallback for CDN browser includes.

- JS: Packed lib files now target ES6 JS, with support for ES5 JS removed.
- JS: [intermediate events for translation scenario targeting v2 endpoint are correctly handled](#)
- JS: [The language property for TranslationRecognitionEventArgs is now set for translation.hypothesis events.](#)
- Speech Synthesis: SynthesisCompleted event is guaranteed to be emitted after all metadata events, so it could be used to indicate to the end of events. [How to detect when visemes are received completely? Issue #2093 Azure-Samples/cognitive-services-speech-sdk](#)

Samples

- Added sample to demonstrate [MULAW streaming using Python](#))
- Fix for [speech-to-text NAudio sample](#)

Speech CLI 1.33.0: October 2023 release

New features

- Support word boundary events output when synthesizing speech.

Bug fixes

- none

Speech SDK 1.32.1: September 2023 release

Bug fixes

- Android packages updates with latest security fixes from OpenSSL1.1.1v
- JS – WebWorkerLoadType property added to allow bypass of data URL load for timeout worker
- JS – Fix Conversation Translation disconnect after 10 minutes
- JS – Conversation Translation auth token from Conversation now propagates to Translation service connection

Samples

- [Conversation transcription with Swift APIs](#)

Speech SDK 1.31.0: August 2023 release

New Features

- Support for [real-time diarization](#) is available in public preview with the Speech SDK 1.31.0. This feature is available in the following SDKs: C#, C++, Java, JavaScript, Python, and Objective-C/Swift.
- Synchronized speech synthesis word boundary and viseme events with audio playback

Breaking changes

- The former "conversation transcription" scenario is renamed to "meeting transcription". For example, use `MeetingTranscriber` instead of `ConversationTranscriber`, and use `CreateMeetingAsync` instead of `CreateConversationAsync`. Although the names of SDK objects and methods have changed, the renaming doesn't change the feature itself. Use meeting transcription objects for transcription of meetings with user profiles and voice signatures. The "conversation translation" objects and methods aren't affected by these changes. You can still use the `ConversationTranslator` object and its methods for meeting translation scenarios.
- For real-time diarization, a new `ConversationTranscriber` object is introduced. The new "conversation transcription" object model and call patterns are similar to continuous recognition with the `SpeechRecognizer` object. A key difference is that the `ConversationTranscriber` object is designed to be used in a conversation scenario where you want to differentiate multiple speakers (diarization). User profiles and voice signatures aren't applicable. See the [real-time diarization quickstart](#) for more information.

This table shows the previous and new object names for real-time diarization and meeting transcription. The scenario name is in the first column, the previous object names are in the second column, and the new object names are in the third column.

[+] Expand table

Scenario name	Previous object names	New object names
Real-time diarization	N/A	<code>ConversationTranscriber</code>

Scenario name	Previous object names	New object names
Meeting transcription	ConversationTranscriber ConversationTranscriptionEventArgs ConversationTranscriptionCanceledEventArgs ConversationTranscriptionResult RemoteConversationTranscriptionResult RemoteConversationTranscriptionClient RemoteConversationTranscriptionResult Participant ¹ ParticipantChangedReason ¹ User ¹	MeetingTranscriber MeetingTranscriptionEventArgs MeetingTranscriptionCanceledEventArgs MeetingTranscriptionResult RemoteMeetingTranscriptionResult RemoteMeetingTranscriptionClient RemoteMeetingTranscriptionResult Participant ParticipantChangedReason User Meeting ²

¹ The `Participant`, `ParticipantChangedReason`, and `User` objects are applicable to both meeting transcription and meeting translation scenarios.

² The `Meeting` object is new and is used with the `MeetingTranscriber` object.

Bug fixes

- Fixed macOS minimum supported version <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/2017> ↗
- Fixed Pronunciation Assessment bug:
 - Addressed phoneme accuracy scores issue, ensuring they now accurately reflect only the specific mispronounced phoneme. <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/1917> ↗
 - Resolved an issue where the Pronunciation Assessment feature was inaccurately identifying entirely correct pronunciations as erroneous, particularly in situations where words could have multiple valid pronunciations. <https://github.com/Azure-Samples/cognitive-services-speech-sdk/issues/1530> ↗

Samples

- CSharp
 - [New C# conversation transcription quickstart](#) ↗
 - [New C# meeting transcription quickstart](#) ↗
- JavaScript
 - [New JavaScript conversation transcription quickstart](#) ↗

- [New JavaScript meeting transcription quickstart ↗](#)
- [New Node.js conversation transcription quickstart ↗](#)
- [New Node.js meeting transcription quickstart ↗](#)

Speech SDK 1.30.0: July 2023 release

New Features

- **C++, C#, Java** - Added support for `DisplayWords` in Embedded Speech Recognition's detailed result.
- **Objective-C/Swift** - Added support for `ConnectionMessageReceived` event in Objective-C/Swift.
- **Objective-C/Swift** - Improved keyword-spotting models for iOS. This change has increased the size of certain packages, which contain iOS binaries (like NuGet, XCFramework). We're working to reduce the size for future releases.

Bug fixes

- Fixed a memory leak when using speech recognizer with `PhraseListGrammar`, as reported by a customer ([GitHub issue ↗](#)).
- Fixed a deadlock in text to speech open connection API.

More notes

- **Java** - Some internally used, `public` Java API methods were changed to package `internal`, `protected` or `private`. This change shouldn't have an effect on developers, as we don't expect applications to be using those. Noted here for transparency.

Samples

- New Pronunciation Assessment samples on how to specify a learning language in your own application
 - **C#**: See [sample code ↗](#).
 - **C++**: See [sample code ↗](#).
 - **JavaScript**: See [sample code ↗](#).
 - **Objective-C**: See [sample code ↗](#).
 - **Python**: See [sample code ↗](#).
 - **Swift**: See [sample code ↗](#).

Speech SDK 1.29.0: June 2023 release

New Features

- C++, C#, Java - Preview of Embedded Speech Translation APIs. Now you can do speech translation without cloud connection!
- JavaScript - Continuous Language Identification (LID) now enabled for speech translation.
- JavaScript - Community contribution for adding `LocaleName` property to `VoiceInfo` class. Thank you GitHub user [shivsarthak](#) for the pull request.
- C++, C#, Java - Added support for resampling Embedded text to speech output from 16 kHz to 48 kHz sample rate.
- Added support for `hi-IN` locale in Intent Recognizer with Simple Pattern Matching.

Bug fixes

- Fixed a crash caused by a race condition in Speech Recognizer during object destruction, as seen in some of our Android tests
- Fixed possible deadlocks in Intent Recognizer with Simple Pattern Matcher

Samples

- New Embedded Speech Translation samples

Speech SDK 1.28.0: May 2023 release

Breaking change

- **JavaScript SDK:** Online Certificate Status Protocol (OCSP) was removed. This allows clients to better conform to browser and Node standards for certificate handling. Version 1.28 and onward will no longer include our custom OCSP module.

New Features

- **Embedded Speech Recognition** now returns `NoMatchReason::EndSilenceTimeout` when a silence timeout occurs at the end of an utterance. This matches the behavior when doing recognition using the real-time speech service.
- **JavaScript SDK:** Set properties on `SpeechTranslationConfig` using `PropertyId` enum values.

Bug fixes

- **C# on Windows** - Fix potential race condition/deadlock in Windows audio extension. In scenarios that both dispose of the audio renderer quickly and also use the Synthesizer method to stop speaking, the underlying event wasn't reset by stop, and could cause the renderer object to never be disposed, all while it could be holding a global lock for disposal, freezing the dotnet GC thread.

Samples

- Added an embedded speech sample for MAUI.
- Updated the embedded speech sample for Android Java to include text to speech.

Speech SDK 1.27.0: April 2023 release

Notification about upcoming changes

- We plan to remove Online Certificate Status Protocol (OCSP) in the next JavaScript SDK release. This allows clients to better conform to browser and Node standards for certificate handling. Version 1.27 is the last release that includes our custom OCSP module.

New Features

- **JavaScript** – Added support for microphone input from the browser with Speaker Identification and Verification.
- **Embedded Speech Recognition** - Update support for `PropertyId::Speech_SegmentationSilenceTimeoutMs` setting.

Bug fixes

- **General** - Reliability updates in service reconnection logic (all programming languages except JavaScript).
- **General** - Fix string conversions leaking memory on Windows (all relevant programming languages except JavaScript).
- **Embedded Speech Recognition** - Fix crash in French Speech Recognition when using certain grammar list entries.
- **Source code documentation** - Corrections to SDK reference documentation comments related to audio logging on the service.
- **Intent recognition** - Fix Pattern Matcher priorities related to list entities.

Samples

- Properly handle authentication failure in C# Conversation Transcription (CTS) sample.
- Added example of streaming pronunciation assessment for Python, JavaScript, Objective-C and Swift.

Speech SDK 1.26.0: March 2023 release

Breaking changes

- Bitcode has been disabled in all iOS targets in the following packages: Cocoapod with xcframework, NuGet (for Xamarin and MAUI) and Unity. The change is due to Apple's deprecation of bitcode support from Xcode 14 and onwards. This change also means if you're using Xcode 13 version or you have explicitly enabled the bitcode on your application using the Speech SDK, you might encounter an error saying "framework doesn't contain bitcode and you must rebuild it". To resolve this issue, make sure your targets have bitcode disabled.
- Minimum iOS deployment target is upgraded to 11.0 in this release, which means armv7 HW is no longer supported.

New features

- Embedded (on-device) Speech Recognition now supports both 8 and 16-kHz sampling rate input audio (16-bit per sample, mono PCM).
- Speech Synthesis now reports connection, network, and service latencies in the result to help end-to-end latency optimization.
- New tie breaking rules for [Intent Recognition with simple pattern matching](#). The more character bytes that are matched, will win over pattern matches with lower character byte count. Example: Pattern "Select {something} in the top right" will win over "Select {something}"

Bug fixes

- Speech Synthesis: fix a bug where the emoji isn't correct in word boundary events.
- [Intent Recognition with Conversational Language Understanding \(CLU\)](#):
 - Intents from the CLU Orchestrator Workflow now appear correctly.
 - The JSON result is now available via the property `ID`
`LanguageUnderstandingServiceResponse.JsonResult`.
- Speech recognition with keyword activation: Fix for missing ~150 ms audio after a keyword recognition.

- Fix for Speech SDK NuGet iOS MAUI Release build, reported by customer ([GitHub issue ↗](#))

Samples

- Fix for Swift iOS sample, reported by customer ([GitHub issue ↗](#))

Speech SDK 1.25.0: January 2023 release

Breaking changes

- Language Identification (preview) APIs have been simplified. If you update to Speech SDK 1.25 and see a build break, please visit the [Language Identification](#) page to learn about the new property `SpeechServiceConnection_LanguageIdMode`. This single property replaces the two previous ones `SpeechServiceConnection_SingleLanguageIdPriority` and `SpeechServiceConnection_ContinuousLanguageIdPriority`. Prioritizing between low latency and high accuracy is no longer necessary following recent model improvements. Now, you only need to select whether to run at-start or continuous Language Identification when doing continuous speech recognition or translation.

New features

- **C#/C++/Java:** Embedded Speech SDK is now released under gated public preview. See [Embedded Speech \(preview\)](#) documentation. You can now do on-device speech to text and text to speech when cloud connectivity is intermittent or unavailable. Supported on Android, Linux, macOS, and Windows platforms
- **C# MAUI:** Support added for iOS and Mac Catalyst targets in Speech SDK NuGet ([Customer issue ↗](#))
- **Unity:** Android x86_64 architecture added to Unity package ([Customer issue ↗](#))
- **Go:**
 - ALAW/MULAW direct streaming support added for speech recognition ([Customer issue ↗](#))
 - Added support for PhraseListGrammar. Thank you GitHub user [czkoko ↗](#) for the community contribution!
- **C#/C++:** Intent Recognizer now supports Conversational Language Understanding models in C++ and C# with orchestration on the Microsoft service

Bug fixes

- Fix an occasional hang in **KeywordRecognizer** when trying to stop it
- **Python:**
 - Fix for getting Pronunciation Assessment results when `PronunciationAssessmentGranularity.FullText` is set ([Customer issue ↗](#))
 - Fix for gender property for Male voices not being retrieved, when getting speech synthesis voices
- **JavaScript**
 - Fix for parsing some WAV files that were recorded on iOS devices ([Customer issue ↗](#))
 - JS SDK now builds without using npm-force-resolutions ([Customer issue ↗](#))
 - Conversation Translator now correctly sets service endpoint when using a `speechConfig` instance created using `SpeechConfig.fromEndpoint()`

Samples

- Added samples showing how to use Embedded Speech
- Added Speech to text sample for MAUI

See [Speech SDK samples repository ↗](#).

Speech SDK 1.24.2: November 2022 release

New features

- No new features, just an embedded engine fix to support new model files.

Bug fixes

- All programming languages
 - Fixed an issue with encryption of embedded speech recognition models.

Speech SDK 1.24.1: November 2022 release

New features

- Published packages for the Embedded Speech preview. See [https://aka.ms/embedded-speech ↗](https://aka.ms/embedded-speech) for more information.

Bug fixes

- All programming languages
 - Fix embedded TTS crash when voice font isn't supported
 - Fix stopSpeaking() can't stop playback on Linux ([#1686](#))
- JavaScript SDK
 - Fixed regression in how conversation transcriber gated audio.
- Java
 - Temporarily Published updated POM and Javadocs files to Maven Central to enable the docs pipeline to update online reference docs.
- Python
 - Fix regression where Python speak_text(ssml) returns void.

Speech SDK 1.24.0: October 2022 release

New features

- All programming languages: AMR-WB (16khz) added to the supported list of Text to speech audio output formats
- Python: Package added for Linux Arm64 for supported Linux distributions.
- C#/C++/Java/Python: Support added for ALAW & MULAW direct streaming to the speech service (in addition to existing PCM stream) using `AudioStreamWaveFormat`.
- C# MAUI: NuGet package updated to support Android targets for [.NET MAUI](#) developers ([Customer issue](#))
- Mac: Added separate XCframework for Mac, which doesn't contain any iOS binaries. This offers an option for developers who need only Mac binaries using a smaller XCframework package.
- Microsoft Audio Stack (MAS):
 - When beam-forming angles are specified, sound originating outside of specified range will be suppressed better.
 - Approximately 70% reduction in the size of `libMicrosoft.CognitiveServices.Speech.extension.mas.so` for Linux ARM32 and Linux Arm64.
- Intent Recognition using pattern matching:
 - Add orthography support for the languages `fr`, `de`, `es`, `jp`
 - Added prebuilt integer support for language `es`.

Bug fixes

- iOS: fix speech synthesis error on iOS 16 caused by compressed audio decoding failure ([Customer Issue](#)).
- JavaScript:

- Fix authentication token not working when getting speech synthesis voice list ([Customer issue ↗](#)).
- Use data URL for worker loading ([Customer issue ↗](#)).
- Create audio processor worklet only when AudioWorklet is supported in browser ([Customer issue ↗](#)). This was a community contribution by [William Wong ↗](#). Thank you William!
- Fix recognized callback when LUIS response `connectionMessage` is empty ([Customer issue ↗](#)).
- Properly set speech segmentation timeout.
- **Intent Recognition using pattern matching:**
 - Non-json characters inside models now load properly.
 - Fix hanging issue when `recognizeOnceAsync(text)` was called during continuous recognition.

Speech SDK 1.23.0: July 2022 release

New features

- **C#, C++, Java:** Added support for languages `zh-cn` and `zh-hk` in Intent Recognition with Pattern Matching.
- **C#:** Added support for `AnyCPU` .NET Framework builds

Bug fixes

- **Android:** Fixed OpenSSL vulnerability CVE-2022-2068 by updating OpenSSL to 1.1.1q
- **Python:** Fix crash when using PushAudioInputStream
- **iOS:** Fix "EXC_BAD_ACCESS: Attempted to dereference null pointer" as reported on iOS ([GitHub issue ↗](#))

Speech SDK 1.22.0: June 2022 release

New features

- **Java:** IntentRecognitionResult API for `getEntities()`, `applyLanguageModels()`, and `recognizeOnceAsync(text)` added to support the "simple pattern matching" engine.
- **Unity:** Added support for Mac M1 (Apple Silicon) for Unity package ([GitHub issue ↗](#))
- **C#:** Added support for x86_64 for Xamarin Android ([GitHub issue ↗](#))
- **C#:** .NET framework minimum version updated to v4.6.2 for SDK C# package as v4.6.1 has retired (see [Microsoft .NET Framework Component Lifecycle Policy](#))

- **Linux:** Added support for Debian 11 and Ubuntu 22.04 LTS. Ubuntu 22.04 LTS requires manual installation of libssl1.1 either as a binary package from [here](#) (for example, libssl1.1_1.1.1l-1ubuntu1.3_amd64.deb or newer for x64), or by compiling from sources.

Bug fixes

- **UWP:** OpenSSL dependency removed from UWP libraries and replaced with WinRT websocket and HTTP APIs to meet security compliance and smaller binary footprint.
- **Mac:** Fixed "MicrosoftCognitiveServicesSpeech Module Not Found" issue when using Swift projects targeting macOS platform
- **Windows, Mac:** Fixed a platform-specific issue where audio sources that were configured via properties to stream at a real-time rate sometimes fell behind and eventually exceeded capacity

Samples ([GitHub](#))

- **C#:** .NET framework samples updated to use v4.6.2
- **Unity:** Virtual-assistant sample fixed for Android and UWP
- **Unity:** Unity samples updated for Unity 2020 LTS version

Speech SDK 1.21.0: April 2022 release

New features

- **Java & JavaScript:** Added support for Continuous Language Identification when using the SpeechRecognizer object
- **JavaScript:** Added Diagnostics APIs to enable console logging level and (Node only) file logging, to help Microsoft troubleshoot customer-reported issues
- **Python:** Added support for Conversation Transcription
- **Go:** Added support for Speaker Recognition
- **C++ & C#:** Added support for a required group of words in the Intent Recognizer (simple pattern matching). For example: "(set|start|begin) a timer" where either "set", "start" or "begin" must be present for the intent to be recognized.
- **All programming languages, Speech Synthesis:** Added duration property in word boundary events. Added support for punctuation boundary and sentence boundary
- **Objective-C/Swift/Java:** Added word-level results on the Pronunciation Assessment result object (similar to C#). The application no longer needs to parse a JSON result string to get word-level information ([GitHub issue](#))
- **iOS platform:** Added experimental support for ARMv7 architecture

Bug fixes

- **iOS platform:** Fix to allow building for the target "Any iOS Device", when using CocoaPod ([GitHub issue ↗](#))
- **Android platform:** OpenSSL version has been updated to 1.1.1n to fix security vulnerability [CVE-2022-0778 ↗](#)
- **JavaScript:** Fix issue where wav header wasn't updated with file size ([GitHub issue ↗](#))
- **JavaScript:** Fix request ID desync issue breaking translation scenarios ([GitHub issue ↗](#))
- **JavaScript:** Fix issue when instantiating SpeakerAudioDestination with no stream ([GitHub issue ↗](#))
- **C++:** Fix C++ headers to remove a warning when compiling for C++17 or newer

Samples [GitHub ↗](#)

- New **Java** samples for Speech Recognition with Language Identification
- New **Python** and **Java** samples for Conversation Transcription
- New **Go** sample for Speaker Recognition
- New **C++ and C#** tool for Windows that enumerates all audio capture and render devices, for finding their Device ID. This ID is needed by the Speech SDK if you plan to [capture audio from, or render audio to, a nondefault device](#).

Speech SDK 1.20.0: January 2022 release

New features

- **Objective-C, Swift, and Python:** Added support for DialogServiceConnector, used for voice assistant scenarios.
- **Python:** Support for Python 3.10 was added. Support for Python 3.6 was removed, per Python's [end-of-life for 3.6 ↗](#).
- **Unity:** Speech SDK is now supported for Unity applications on Linux.
- **C++, C#:** IntentRecognizer using pattern matching is now supported in C#. In addition, scenarios with custom entities, optional groups, and entity roles are now supported in C++ and C#.
- **C++, C#:** Improved diagnostics trace logging using new classes FileLogger, MemoryLogger, and EventLogger. SDK logs are an important tool for Microsoft to diagnose customer-reported issues. These new classes make it easier for customers to integrate Speech SDK logs into their own logging system.
- **All programming languages:** PronunciationAssessmentConfig now has properties to set the desired phoneme alphabet (IPA or SAPI) and N-Best Phoneme Count

(avoiding the need to author a configuration JSON as per [GitHub issue 1284](#)). Also, syllable level output is now supported.

- **Android, iOS, and macOS (all programming languages):** GStreamer is no longer needed to support limited-bandwidth networks. SpeechSynthesizer now uses the operating system's audio decoding capabilities to decode compressed audio streamed from the text to speech service.
- **All programming languages:** SpeechSynthesizer now supports three new raw output Opus formats (without container), which are widely used in live streaming scenarios.
- **JavaScript:** Added `getVoicesAsync()` API to SpeechSynthesizer to retrieve the list of supported synthesis voices ([GitHub issue 1350](#))
- **JavaScript:** Added `getWaveFormat()` API to `AudioStreamFormat` to support non-PCM wave formats ([GitHub issue 452](#))
- **JavaScript:** Added volume getter/setter and `mute()/unmute()` APIs to `SpeakerAudioDestination` ([GitHub issue 463](#))

Bug fixes

- **C++, C#, Java, JavaScript, Objective-C, and Swift:** Fix to remove a 10-second delay while stopping a speech recognizer that uses a `PushAudioInputStream`. This is for the case where no new audio is pushed in after `StopContinuousRecognition` is called ([GitHub issues 1318](#), [331](#))
- **Unity on Android and UWP:** Unity meta files were fixed for UWP, Android Arm64, and Windows Subsystem for Android (WSA) Arm64 ([GitHub issue 1360](#))
- **iOS:** Compiling your Speech SDK application on any iOS Device when using CocoaPods is now fixed ([GitHub issue 1320](#))
- **iOS:** When `SpeechSynthesizer` is configured to output audio directly to a speaker, playback stopped at the beginning in rare conditions. This was fixed.
- **JavaScript:** Use script processor fallback for microphone input if no audio worklet is found ([GitHub issue 455](#))
- **JavaScript:** Add protocol to agent to mitigate bug found with Sentry integration ([GitHub issue 465](#))

Samples [GitHub](#)

- **C++**, **C#**, **Python**, and **Java** samples showing how to get detailed recognition results. The details include alternative recognition results, confidence score, Lexical form, Normalized form, Masked Normalized form, with word-level timing for each.
- **iOS sample** added using AVFoundation as external audio source.

- [Java sample](#) added to show how to get SRT (SubRip Text) format using WordBoundary event.
- [Android samples](#) for Pronunciation Assessment.
- [C++](#), [C#](#) showing usage of the new Diagnostics Logging classes.

Speech SDK 1.19.0: 2021-Nov release

Highlights

- Speaker Recognition service is generally available (GA) now. Speech SDK APIs are available on C++, C#, Java, and JavaScript. With Speaker Recognition, you can accurately verify and identify speakers by their unique voice characteristics. For more information about this topic, see the [documentation](#).
- We've dropped support for Ubuntu 16.04 in conjunction with Azure DevOps and GitHub. Ubuntu 16.04 reached end of life back in April of 2021. Migrate your Ubuntu 16.04 workflows to Ubuntu 18.04 or newer.
- OpenSSL linking in Linux binaries changed to dynamic. Linux binary size has been reduced by about 50%.
- Mac M1 ARM-based silicon support added.

New features

- **C++/C#/Java:** New APIs added to enable audio processing support for speech input with Microsoft Audio Stack. Documentation [here](#).
- **C++:** New APIs for intent recognition to facilitate more advanced pattern matching. This includes List and Prebuilt Integer entities as well as support for grouping intents and entities as models (Documentation, updates, and samples are under development and will be published in the near future).
- **Mac:** Support for Arm64 (M1) based silicon for CocoaPod, Python, Java, and NuGet packages related to [GitHub issue 1244](#).
- **iOS/Mac:** iOS and macOS binaries are now packaged into xcframework related to [GitHub issue 919](#).
- **iOS/Mac:** Support for Mac catalyst related to [GitHub issue 1171](#).
- **Linux:** New tar package added for CentOS7 [About the Speech SDK](#). The Linux .tar package now contains specific libraries for RHEL/CentOS 7 in `lib/centos7-x64`.

Speech SDK libraries in lib/x64 are still applicable for all the other supported Linux x64 distributions (including RHEL/CentOS 8) and won't work on RHEL/CentOS 7.

- **JavaScript:** VoiceProfile & SpeakerRecognizer APIs made async/awaitable.
- **JavaScript:** Support added for US government Azure regions.
- **Windows:** Support added for playback on Universal Windows Platform (UWP).

Bug fixes

- **Android:** OpenSSL security update (updated to version 1.1.1l) for Android packages.
- **Python:** Resolved bug where selecting speaker device on Python fails.
- **Core:** Automatically reconnect when a connection attempt fails.
- **iOS:** Audio compression disabled on iOS packages due instability and bitcode build problems when using GStreamer. Details are available via [GitHub issue 1209 ↗](#).

Samples [GitHub ↗](#)

- **Mac/iOS:** Updated samples and quickstarts to use xcframework package.
- **.NET:** Samples updated to use .NET core 3.1 version.
- **JavaScript:** Added sample for Voice Assistants.

Speech SDK 1.18.0: 2021-July release

Note: Get started with the Speech SDK [here](#).

Highlights summary

- Ubuntu 16.04 reached end of life in April of 2021. With Azure DevOps and GitHub, we'll drop support for 16.04 in September 2021. Migrate ubuntu-16.04 workflows to ubuntu-18.04 or newer before then.

New features

- **C++:** Simple Language Pattern matching with the Intent Recognizer now makes it easier to [implement simple intent recognition scenarios](#).

- **C++/C#/Java:** We added a new API, `GetActivationPhrasesAsync()` to the `VoiceProfileClient` class for receiving a list of valid activation phrases in Speaker Recognition enrollment phase for independent recognition scenarios.
 - **Important:** The Speaker Recognition feature is in Preview. All voice profiles created in Preview will be discontinued 90 days after the Speaker Recognition feature is moved out of Preview into General Availability. At that point the Preview voice profiles will stop functioning.
- **Python:** Added support for continuous Language Identification (LID) on the existing `SpeechRecognizer` and `TranslationRecognizer` objects.
- **Python:** Added a new Python object named `SourceLanguageRecognizer` to do one-time or continuous LID (without recognition or translation).
- **JavaScript:** `getActivationPhrasesAsync` API added to `VoiceProfileClient` class for receiving a list of valid activation phrases in Speaker Recognition enrollment phase for independent recognition scenarios.
- **JavaScript** `VoiceProfileClient`'s `enrollProfileAsync` API is now async awaitable. See [this independent identification code ↗](#), for example, usage.

Improvements

- **Java:** `AutoCloseable` support added to many Java objects. Now the try-with-resources model is supported to release resources. See [this sample that uses try-with-resources ↗](#). Also see the Oracle Java documentation tutorial for [The try-with-resources Statement ↗](#) to learn about this pattern.
- **Disk footprint** has been significantly reduced for many platforms and architectures. Examples for the `Microsoft.CognitiveServices.Speech.core` binary: x64 Linux is 475KB smaller (8.0% reduction); Arm64 Windows UWP is 464KB smaller (11.5% reduction); x86 Windows is 343KB smaller (17.5% reduction); and x64 Windows is 451KB smaller (19.4% reduction).

Bug fixes

- **Java:** Fixed synthesis error when the synthesis text contains surrogate characters. Details [here ↗](#).
- **JavaScript:** Browser microphone audio processing now uses `AudioWorkletNode` instead of deprecated `ScriptProcessorNode`. Details [here ↗](#).
- **JavaScript:** Correctly keep conversations alive during long running conversation translation scenarios. Details [here ↗](#).
- **JavaScript:** Fixed issue with recognizer reconnecting to a mediastream in continuous recognition. Details [here ↗](#).

- **JavaScript:** Fixed issue with recognizer reconnecting to a pushStream in continuous recognition. Details [here ↗](#).
- **JavaScript:** Corrected word level offset calculation in detailed recognition results. Details [here ↗](#).

Samples

- Java quickstart samples updated [here ↗](#).
- JavaScript Speaker Recognition samples updated to show new usage of `enrollProfileAsync()`. See samples [here ↗](#).

Speech SDK 1.17.0: 2021-May release

⚠ Note

Get started with the Speech SDK [here](#).

Highlights summary

- Smaller footprint - we continue to decrease the memory and disk footprint of the Speech SDK and its components.
- A new stand-alone Language Identification API allows you to recognize what language is being spoken.
- Develop speech enabled mixed reality and gaming applications using Unity on macOS.
- You can now use Text to speech in addition to speech recognition from the Go programming language.
- Several Bug fixes to address issues YOU, our valued customers, have flagged on GitHub! THANK YOU! Keep the feedback coming!

New features

- **C++/C#:** New stand-alone At-Start and Continuous Language Detection via the `SourceLanguageRecognizer` API. If you only want to detect the language(s) spoken in audio content, this is the API to do that. See details for [C++](#) and [C#](#).
- **C++/C#:** Speech Recognition and Translation Recognition now support both at-start and continuous Language Identification so you can programmatically determine which language(s) are being spoken before they're transcribed or translated. See documentation [here for Speech Recognition](#) and [here for Speech Translation](#).

- **C#**: Added support Unity support to macOS (x64). This unlocks speech recognition and speech synthesis use cases in mixed reality and gaming!
- **Go**: We added support for speech synthesis text to speech to the Go programming language to make speech synthesis available in even more use cases. See our [quickstart](#) or our [reference documentation ↗](#).
- **C++/C#/Java/Python/Objective-C/Go**: The speech synthesizer now supports the `connection` object. This helps you manage and monitor the connection to the Speech service, and is especially helpful to pre-connect to reduce latency. See documentation [here](#).
- **C++/C#/Java/Python/Objective-C/Go**: We now expose the latency and underrun time in `SpeechSynthesisResult` to help you monitor and diagnose speech synthesis latency issues. See details for [C++](#), [C#](#), [Java](#), [Python](#), [Objective-C](#) and [Go ↗](#).
- **C++/C#/Java/Python/Objective-C**: Text to speech [now uses neural voices](#) by default when you don't specify a voice to be used. This gives you higher fidelity output by default, but also [increases the default price ↗](#).
- **C++/C#/Java/Python/Objective-C/Go**: We added a Gender property to the synthesis voice info to make it easier to select voices based on gender. This addresses [GitHub issue #1055 ↗](#).
- **C++, C#, Java, JavaScript**: We now support `retrieveEnrollmentResultAsync`, `getAuthorizationPhrasesAsync`, and `getAllProfilesAsync()` in Speaker Recognition to ease user management of all voice profiles for a given account. See documentation for [C++](#), [C#](#), [Java](#), [JavaScript](#). This addresses [GitHub issue #338 ↗](#).
- **JavaScript**: We added retry for connection failures that will make your JavaScript-based speech applications more robust.

Improvements

- Linux and Android Speech SDK binaries have been updated to use the latest version of OpenSSL (1.1.1k)
- Code Size improvements:
 - Language Understanding is now split into a separate "lu" library.
 - Windows x64 core binary size decreased by 14.4%.
 - Android Arm64 core binary size decreased by 13.7%.
 - other components also decreased in size.

Bug fixes

- **All**: Fixed [GitHub issue #842 ↗](#) for ServiceTimeout. You can now transcribe long audio files using the Speech SDK without the connection to the service terminating with this error. However, we still recommend you use [batch transcription](#) for long files.

- **C#**: Fixed [GitHub issue #947](#) where no speech input could leave your app in a bad state.
- **Java**: Fixed [GitHub Issue #997](#) where the Speech SDK for Java 1.16 crashes when using DialogServiceConnector without a network connection or an invalid subscription key.
- Fixed a crash when abruptly stopping speech recognition (for example, using CTRL+C on console app).
- **Java**: Added a fix to delete temporary files on Windows when using Speech SDK for Java.
- **Java**: Fixed [GitHub issue #994](#) where calling `DialogServiceConnector.stopListeningAsync` could result in an error.
- **Java**: Fixed a customer issue in the [virtual assistant quickstart](#).
- **JavaScript**: Fixed [GitHub issue #366](#) where `ConversationTranslator` threw an error 'this.cancelSpeech isn't a function'.
- **JavaScript**: Fixed [GitHub issue #298](#) where 'Get result as an in-memory stream' sample played sound out loud.
- **JavaScript**: Fixed [GitHub issue #350](#) where calling `AudioConfig` could result in a 'ReferenceError: MediaStream isn't defined'.
- **JavaScript**: Fixed an UnhandledPromiseRejection warning in Node.js for long-running sessions.

Samples

- Updated Unity samples documentation for macOS [here](#).
- A React Native sample for the Azure AI Speech recognition service is now available [here](#).

Speech SDK 1.16.0: 2021-March release

Note

The Speech SDK on Windows depends on the shared Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019.

New features

- **C++/C#/Java/Python**: Moved to the latest version of GStreamer (1.18.3) to add support for transcribing any media format on Windows, Linux, and Android. See documentation [here](#).

- **C++/C#/Java/Objective-C/Python:** Added support for decoding compressed TTS/synthesized audio to the SDK. If you set output audio format to PCM and GStreamer is available on your system, the SDK will automatically request compressed audio from the service to save bandwidth and decode the audio on the client. You can set `SpeechServiceConnection_SynthEnableCompressedAudioTransmission` to `false` to disable this feature. Details for [C++](#), [C#](#), [Java](#), [Objective-C](#), [Python](#).
- **JavaScript:** Node.js users can now use the [AudioConfig.fromWavFileInput API](#). This addresses [GitHub issue #252](#).
- **C++/C#/Java/Objective-C/Python:** Added `GetVoicesAsync()` method for TTS to return all available synthesis voices. Details for [C++](#), [C#](#), [Java](#), [Objective-C](#), and [Python](#).
- **C++/C#/Java/JavaScript/Objective-C/Python:** Added `VisemeReceived` event for TTS/speech synthesis to return synchronous viseme animation. See documentation [here](#).
- **C++/C#/Java/JavaScript/Objective-C/Python:** Added `BookmarkReached` event for TTS. You can set bookmarks in the input SSML and get the audio offsets for each bookmark. See documentation [here](#).
- **Java:** Added support for Speaker Recognition APIs. Details [here](#).
- **C++/C#/Java/JavaScript/Objective-C/Python:** Added two new output audio formats with WebM container for TTS (Webm16Khz16BitMonoOpus and Webm24Khz16BitMonoOpus). These are better formats for streaming audio with the Opus codec. Details for [C++](#), [C#](#), [Java](#), [JavaScript](#), [Objective-C](#), [Python](#).
- **C++/C#/Java:** Added support for retrieving voice profile for Speaker Recognition scenario. Details for [C++](#), [C#](#), and [Java](#).
- **C++/C#/Java/Objective-C/Python:** Added support for separate shared library for audio microphone and speaker control. This allows the developer to use the SDK in environments that don't have required audio library dependencies.
- **Objective-C/Swift:** Added support for module framework with umbrella header. This allows the developer to import Speech SDK as a module in iOS/Mac Objective-C/Swift apps. This addresses [GitHub issue #452](#).
- **Python:** Added support for [Python 3.9](#) and dropped support for Python 3.5 per Python's [end-of-life for 3.5](#).

Known issues

- **C++/C#/Java:** `DialogServiceConnector` can't use a `CustomCommandsConfig` to access a Custom Commands application and will instead encounter a connection error. This can be worked around by manually adding your application ID to the request with `config.SetServiceProperty("X-CommandsAppId", "your-application-id", ServicePropertyChannel.UriQueryParameter)`. The expected behavior of `CustomCommandsConfig` will be restored in the next release.

Improvements

- As part of our multi-release effort to reduce the Speech SDK's memory usage and disk footprint, Android binaries are now 3% to 5% smaller.
- Improved accuracy, readability, and see-also sections of our C# reference documentation [here](#).

Bug fixes

- **JavaScript:** Large WAV file headers are now parsed correctly (increases header slice to 512 bytes). This addresses [GitHub issue #962](#).
- **JavaScript:** Corrected microphone timing issue if mic stream ends before stop recognition, addressing an issue with Speech Recognition not working in Firefox.
- **JavaScript:** We now correctly handle initialization promise when the browser forces mic off before turnOn completes.
- **JavaScript:** We replaced URL dependency with url-parse. This addresses [GitHub issue #264](#).
- **Android:** Fixed callbacks not working when `minifyEnabled` is set to true.
- **C++/C#/Java/Objective-C/Python:** `TCP_NODELAY` will be correctly set to underlying socket IO for TTS to reduce latency.
- **C++/C#/Java/Python/Objective-C/Go:** Fixed an occasional crash when the recognizer was destroyed just after starting a recognition.
- **C++/C#/Java:** Fixed an occasional crash in the destruction of speaker recognizer.

Samples

- **JavaScript:** [Browser samples](#) no longer require separate JavaScript library file download.

Speech SDK 1.15.0: 2021-January release

ⓘ Note

The Speech SDK on Windows depends on the shared Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019.

Highlights summary

- Smaller memory and disk footprint making the SDK more efficient.

- Higher fidelity output formats available for custom-neural voice private preview.
- Intent Recognizer can now get return more than the top intent, giving you the ability to make a separate assessment about your customer's intent.
- Voice assistants and bots are now easier to set up, and you can make it stop listening immediately, and exercise greater control over how it responds to errors.
- Improved on device performance through making compression optional.
- Use the Speech SDK on Windows ARM/Arm64.
- Improved low-level debugging.
- Pronunciation Assessment feature is now more widely available.
- Several Bug fixes to address issues YOU, our valued customers, have flagged on GitHub! THANK YOU! Keep the feedback coming!

Improvements

- The Speech SDK is now more efficient and lightweight. We've started a multi-release effort to reduce the Speech SDK's memory usage and disk footprint. As a first step we made significant file size reductions in shared libraries on most platforms. Compared to the 1.14 release:
 - 64-bit UWP-compatible Windows libraries are about 30% smaller.
 - 32-bit Windows libraries aren't yet seeing a size improvement.
 - Linux libraries are 20-25% smaller.
 - Android libraries are 3-5% smaller.

New features

- **All:** New 48 KHz output formats available for the private preview of custom-neural voice through the TTS speech synthesis API: Audio48Khz192KBitRateMonoMp3, audio-48khz-192kbitrate-mono-mp3, Audio48Khz96KBitRateMonoMp3, audio-48khz-96kbitrate-mono-mp3, Raw48Khz16BitMonoPcm, raw-48khz-16bit-mono-pcm, Riff48Khz16BitMonoPcm, riff-48khz-16bit-mono-pcm.
- **All:** Custom voice is also easier to use. Added support for setting custom voice via `EndpointId` ([C++](#), [C#](#), [Java](#), [JavaScript](#), [Objective-C](#), [Python](#)). Before this change, custom voice users needed to set the endpoint URL via the `FromEndpoint` method. Now customers can use the `FromSubscription` method just like standard voices, and then provide the deployment ID by setting `EndpointId`. This simplifies setting up custom voices.
- **C++/C#/Java/Objective-C/Python:** Get more than the top intent from `IntentRecognizer`. It now supports configuring the JSON result containing all intents and not only the top scoring intent via `LanguageUnderstandingModel`

`FromEndpoint` method by using `verbose=true` `uri` parameter. This addresses [GitHub issue #880](#). See updated documentation [here](#).

- **C++/C#/Java:** Make your voice assistant or bot stop listening immediately. `DialogServiceConnector` ([C++](#), [C#](#), [Java](#)) now has a `StopListeningAsync()` method to accompany `ListenOnceAsync()`. This will immediately stop audio capture and gracefully wait for a result, making it perfect for use with "stop now" button-press scenarios.
- **C++/C#/JavaScript:** Make your voice assistant or bot react better to underlying system errors. `DialogServiceConnector` ([C++](#), [C#](#), [Java](#), [JavaScript](#)) now has a new `TurnStatusReceived` event handler. These optional events correspond to every [ITurnContext](#) resolution on the Bot and will report turn execution failures when they happen, for example, as a result of an unhandled exception, timeout, or network drop between Direct Line Speech and the bot. `TurnStatusReceived` makes it easier to respond to failure conditions. For example, if a bot takes too long on a backend database query (for example, looking up a product), `TurnStatusReceived` allows the client to know to reprompt with "sorry, I didn't quite get that, could you please try again" or something similar.
- **C++/C#:** Use the Speech SDK on more platforms. The [Speech SDK NuGet package](#) now supports Windows ARM/Arm64 desktop native binaries (UWP was already supported) to make the Speech SDK more useful on more machine types.
- **Java:** `DialogServiceConnector` now has a `setSpeechActivityTemplate()` method that was unintentionally excluded from the language previously. This is equivalent to setting the `Conversation_Speech_Activity_Template` property and will request that all future Bot Framework activities originated by the Direct Line Speech service merge the provided content into their JSON payloads.
- **Java:** Improved low-level debugging. The `Connection` class now has a `MessageReceived` event, similar to other programming languages (C++, C#). This event provides low-level access to incoming data from the service and can be useful for diagnostics and debugging.
- **JavaScript:** Easier setup for Voice Assistants and bots through `BotFrameworkConfig`, which now has `fromHost()` and `fromEndpoint()` factory methods that simplify the use of custom service locations versus manually setting properties. We also standardized optional specification of `botId` to use a non-default bot across the configuration factories.
- **JavaScript:** Improved on device performance through added string control property for websocket compression. For performance reasons, we disabled websocket compression by default. This can be reenabled for low-bandwidth scenarios. More details [here](#). This addresses [GitHub issue #242](#).
- **JavaScript:** Added support for IPronunciation Assessment to enable evaluation of speech pronunciation. See the quickstart [here](#).

Bug fixes

- **All** (except JavaScript): Fixed a regression in version 1.14, in which too much memory was allocated by the recognizer.
- **C++**: Fixed a garbage collection issue with `DialogServiceConnector`, addressing [GitHub issue #794 ↗](#).
- **C#**: Fixed an issue with thread shutdown that caused objects to block for about a second when disposed.
- **C++/C#/Java**: Fixed an exception preventing an application from setting speech authorization token or activity template more than once on a `DialogServiceConnector`.
- **C++/C#/Java**: Fixed a recognizer crash due to a race condition in teardown.
- **JavaScript**: `DialogServiceConnector` didn't previously honor the optional `botId` parameter specified in `BotFrameworkConfig`'s factories. This made it necessary to set the `botId` query string parameter manually to use a non-default bot. The bug has been corrected and `botId` values provided to `BotFrameworkConfig`'s factories will be honored and used, including the new `fromHost()` and `fromEndpoint()` additions. This also applies to the `applicationId` parameter for `CustomCommandsConfig`.
- **JavaScript**: Fixed [GitHub issue #881 ↗](#), allowing recognizer object reusage.
- **JavaScript**: Fixed an issue where the SKD was sending `speech.config` multiple times in one TTS session, wasting bandwidth.
- **JavaScript**: Simplified error handling on microphone authorization, allowing more descriptive message to bubble up when user hasn't allowed microphone input on their browser.
- **JavaScript**: Fixed [GitHub issue #249 ↗](#) where type errors in `ConversationTranslator` and `ConversationTranscriber` caused a compilation error for TypeScript users.
- **Objective-C**: Fixed an issue where GStreamer build failed for iOS on Xcode 11.4, addressing [GitHub issue #911 ↗](#).
- **Python**: Fixed [GitHub issue #870 ↗](#), removing "DeprecationWarning: the imp module is deprecated in favor of importlib".

Samples

- [From-file sample for JavaScript browser ↗](#) now uses files for speech recognition. This addresses [GitHub issue #884 ↗](#).

Speech SDK 1.14.0: 2020-October release

Note

The Speech SDK on Windows depends on the shared Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019.

New features

- **Linux:** Added support for Debian 10 and Ubuntu 20.04 LTS.
- **Python/Objective-C:** Added support for the `KeywordRecognizer` API. Documentation will be [here](#).
- **C++/Java/C#:** Added support to set any `HTTPHeader` key/value via `ServicePropertyChannel::HTTPHeader`.
- **JavaScript:** Added support for the `ConversationTranscriber` API. Read documentation [here](#).
- **C++/C#:** Added new `AudioDataStream FromWavFileInput` method (to read .WAV files) [here \(C++\)](#) and [here \(C#\)](#).
- **C++/C#/Java/Python/Objective-C/Swift:** Added a `stopSpeakingAsync()` method to stop text to speech synthesis. Read the Reference documentation [here \(C++\)](#), [here \(C#\)](#), [here \(Java\)](#), [here \(Python\)](#), and [here \(Objective-C/Swift\)](#).
- **C#, C++, Java:** Added a `FromDialogServiceConnector()` function to the `Connection` class that can be used to monitor connection and disconnection events for `DialogServiceConnector`. Read the Reference documentation [here \(C#\)](#), [here \(C++\)](#), and [here \(Java\)](#).
- **C++/C#/Java/Python/Objective-C/Swift:** Added support for Pronunciation Assessment, which evaluates speech pronunciation and gives speakers feedback on the accuracy and fluency of spoken audio. Read the documentation [here](#).

Breaking change

- **JavaScript:** `PullAudioOutputStream.read()` has a return type change from an internal Promise to a Native JavaScript Promise.

Bug fixes

- **All:** Fixed 1.13 regression in `SetServiceProperty` where values with certain special characters were ignored.
- **C#:** Fixed Windows console samples on Visual Studio 2019 failing to find native DLLs.
- **C#:** Fixed crash with memory management if stream is used as `KeywordRecognizer` input.
- **ObjectiveC/Swift:** Fixed crash with memory management if stream is used as recognizer input.

- **Windows:** Fixed coexistence issue with BT HFP/A2DP on UWP.
- **JavaScript:** Fixed mapping of session IDs to improve logging and aid in internal debug/service correlations.
- **JavaScript:** Added fix for `DialogServiceConnector` disabling `ListenOnce` calls after the first call is made.
- **JavaScript:** Fixed issue where result output would only ever be "simple".
- **JavaScript:** Fixed continuous recognition issue in Safari on macOS.
- **JavaScript:** CPU load mitigation for high request throughput scenario.
- **JavaScript:** Allow access to details of Voice Profile Enrollment result.
- **JavaScript:** Added fix for continuous recognition in `IntentRecognizer`.
- **C++/C#/Java/Python/Swift/ObjectiveC:** Fixed incorrect url for australiaeast and brazilsouth in `IntentRecognizer`.
- **C++/C#:** Added `VoiceProfileType` as an argument when creating a `VoiceProfile` object.
- **C++/C#/Java/Python/Swift/ObjectiveC:** Fixed potential `SPX_INVALID_ARG` when trying to read `AudioDataStream` from a given position.
- **IOS:** Fixed crash with speech recognition on Unity

Samples

- **ObjectiveC:** Added sample for keyword recognition [here ↗](#).
- **C#/JavaScript:** Added quickstart for conversation transcription [here \(C#\) ↗](#) and [here \(JavaScript\) ↗](#).
- **C++/C#/Java/Python/Swift/ObjectiveC:** Added sample for Pronunciation Assessment [here ↗](#)

Known Issue

- DigiCert Global Root G2 certificate isn't supported by default in HoloLens 2 and Android 4.4 (KitKat) and needs to be added to the system to make the Speech SDK functional. The certificate will be added to HoloLens 2 OS images in the near future. Android 4.4 customers need to add the updated the certificate to the system.

COVID-19 abridged testing

Due to working remotely over the last few weeks, we couldn't do as much manual verification testing as we normally do. We haven't made any changes we think could have broken anything, and our automated tests all passed. In the unlikely event that we missed something, please let us know on [GitHub ↗](#).

Stay healthy!

Speech SDK 1.13.0: 2020-July release

⚠ Note

The Speech SDK on Windows depends on the shared Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019.

New features

- **C#**: Added support for asynchronous conversation transcription. See documentation [here](#).
- **JavaScript**: Added Speaker Recognition support for both [browser](#) and [Node.js](#).
- **JavaScript**: Added support for Language Identification/language ID. See documentation [here](#).
- **Objective-C**: Added support for [multi-device conversation](#) and conversation transcription.
- **Python**: Added compressed audio support for Python on Windows and Linux. See documentation [here](#).

Bug fixes

- **All**: Fixed an issue that caused the KeywordRecognizer to not move forward the streams after a recognition.
- **All**: Fixed an issue that caused the stream obtained from a KeywordRecognitionResult to not contain the keyword.
- **All**: Fixed an issue that the SendMessageAsync doesn't really send the message over the wire after the users finish waiting for it.
- **All**: Fixed a crash in Speaker Recognition APIs when users call VoiceProfileClient::SpeakerRecEnrollProfileAsync method multiple times and didn't wait for the calls to finish.
- **All**: Fixed enable file logging in VoiceProfileClient and SpeakerRecognizer classes.
- **JavaScript**: Fixed an [issue](#) with throttling when browser is minimized.
- **JavaScript**: Fixed an [issue](#) with a memory leak on streams.
- **JavaScript**: Added caching for OCSP responses from NodeJS.
- **Java**: Fixed an issue that was causing BigInteger fields to always return 0.
- **iOS**: Fixed an [issue](#) with publishing Speech SDK-based apps in the iOS App Store.

Samples

- C++: Added sample code for Speaker Recognition [here](#).

COVID-19 abridged testing

Due to working remotely over the last few weeks, we couldn't do as much manual verification testing as we normally do. We haven't made any changes we think could have broken anything, and our automated tests all passed. In the unlikely event that we missed something, please let us know on [GitHub](#).

Stay healthy!

Speech SDK 1.12.1: 2020-June release

New features

- C#, C++: Speaker Recognition Preview: This feature enables speaker identification (who is speaking?) and speaker verification (is the speaker who they claim to be?). See the [overview documentation](#).

Bug fixes

- C#, C++: Fixed microphone recording wasn't working in 1.12 in Speaker Recognition.
- JavaScript: Fixes for Text to speech in Firefox, and Safari on macOS and iOS.
- Fix for Windows application verifier access violation crash on conversation transcription when using eight-channel stream.
- Fix for Windows application verifier access violation crash on multi-device conversation translation.

Samples

- C#: [Code sample](#) for Speaker Recognition.
- C++: [Code sample](#) for Speaker Recognition.
- Java: [Code sample](#) for intent recognition on Android.

COVID-19 abridged testing

Due to working remotely over the last few weeks, we couldn't do as much manual verification testing as we normally do. We haven't made any changes we think could have broken anything, and our automated tests all passed. In the unlikely event that we missed something, please let us know on [GitHub](#).

Stay healthy!

Speech SDK 1.12.0: 2020-May release

New features

- **Go:** New Go language support for [Speech Recognition](#) and custom voice assistant. Set up your dev environment [here](#). For sample code, see the Samples section below.
- **JavaScript:** Added Browser support for text to speech. See documentation [here](#).
- **C++, C#, Java:** New `KeywordRecognizer` object and APIs supported on Windows, Android, Linux & iOS platforms. Read the documentation [here](#). For sample code, see the Samples section below.
- **Java:** Added multi-device conversation with translation support. See the reference doc [here](#).

Improvements & Optimizations

- **JavaScript:** Optimized browser microphone implementation improving speech recognition accuracy.
- **Java:** Refactored bindings using direct JNI implementation without SWIG. This change reduces by 10x the bindings size for all Java packages used for Windows, Android, Linux, and Mac and eases further development of the Speech SDK Java implementation.
- **Linux:** Updated support [documentation](#) with the latest RHEL 7 specific notes.
- Improved connection logic to attempt connecting multiple times when service and network errors occur.
- Updated the [portal.azure.com](#) Speech Quickstart page to help developers take the next step in the Azure AI Speech journey.

Bug fixes

- **C#, Java:** Fixed an [issue](#) with loading SDK libraries on Linux ARM (both 32 bit and 64 bit).
- **C#:** Fixed explicit disposal of native handles for TranslationRecognizer, IntentRecognizer, and Connection objects.
- **C#:** Fixed audio input lifetime management for ConversationTranscriber object.
- Fixed an issue where `IntentRecognizer` result reason wasn't set properly when recognizing intents from simple phrases.
- Fixed an issue where `SpeechRecognitionEventArgs` result offset wasn't set correctly.
- Fixed a race condition where SDK was trying to send a network message before opening the websocket connection. Was reproducible for `TranslationRecognizer` while adding participants.
- Fixed memory leaks in the keyword recognizer engine.

Samples

- **Go:** Added quickstarts for [speech recognition](#) and custom voice assistant. Find sample code [here ↗](#).
- **JavaScript:** Added quickstarts for [Text to speech](#), [Translation](#), and [Intent Recognition](#).
- Keyword recognition samples for [C# ↗](#) and [Java ↗](#) (Android).

COVID-19 abridged testing

Due to working remotely over the last few weeks, we couldn't do as much manual verification testing as we normally do. We haven't made any changes we think could have broken anything, and our automated tests all passed. If we missed something, please let us know on [GitHub ↗](#).

Stay healthy!

Speech SDK 1.11.0: 2020-March release

New features

- Linux: Added support for Red Hat Enterprise Linux (RHEL)/CentOS 7 x64.
- Linux: Added support for .NET Core C# on Linux ARM32 and Arm64. Read more [here](#).
- C#, C++: Added `UtteranceId` in `ConversationTranscriptionResult`, a consistent ID across all the intermediates and final speech recognition result. Details for [C#](#), [C++](#).
- Python: Added support for `Language ID`. See `speech_sample.py` in [GitHub repo ↗](#).
- Windows: Added compressed audio input format support on Windows platform for all the win32 console applications. Details [here](#).
- JavaScript: Support speech synthesis (text to speech) in NodeJS. Learn more [here ↗](#).
- JavaScript: Add new APIs to enable inspection of all send and received messages. Learn more [here ↗](#).

Bug fixes

- C#, C++: Fixed an issue so `SendMessageAsync` now sends binary message as binary type. Details for [C#](#), [C++](#).
- C#, C++: Fixed an issue where using `Connection MessageReceived` event may cause crash if `Recognizer` is disposed before `Connection` object. Details for [C#](#), [C++](#).
- Android: Audio buffer size from microphone decreased from 800 ms to 100 ms to improve latency.
- Android: Fixed an [issue ↗](#) with x86 Android emulator in Android Studio.

- JavaScript: Added support for Regions in China with the `fromSubscription` API. Details [here](#).
- JavaScript: Add more error information for connection failures from NodeJS.

Samples

- Unity: Intent recognition public sample is fixed, where LUIS json import was failing. Details [here](#).
- Python: Sample added for `Language ID`. Details [here](#).

Covid19 abridged testing: Due to working remotely over the last few weeks, we couldn't do as much manual device verification testing as we normally do. For example, we couldn't test microphone input and speaker output on Linux, iOS, and macOS. We haven't made any changes we think could have broken anything on these platforms, and our automated tests all passed. In the unlikely event that we missed something, let us know on [GitHub](#). Thank you for your continued support. As always, please post questions or feedback on [GitHub](#) or [Stack Overflow](#).

Stay healthy!

Speech SDK 1.10.0: 2020-February release

New features

- Added Python packages to support the new 3.8 release of Python.
- Red Hat Enterprise Linux (RHEL)/CentOS 8 x64 support (C++, C#, Java, Python).

 **Note**

Customers must configure OpenSSL according to [these instructions](#).

- Linux ARM32 support for Debian and Ubuntu.
- DialogServiceConnector now supports an optional "bot ID" parameter on BotFrameworkConfig. This parameter allows the use of multiple Direct Line Speech bots with a single Speech resource. Without the parameter specified, the default bot (as determined by the Direct Line Speech channel configuration page) will be used.
- DialogServiceConnector now has a SpeechActivityTemplate property. The contents of this JSON string will be used by Direct Line Speech to prepopulate a wide variety of supported fields in all activities that reach a Direct Line Speech bot, including activities automatically generated in response to events like speech recognition.
- TTS now uses subscription key for authentication, reducing the first byte latency of the first synthesis result after creating a synthesizer.

- Updated speech recognition models for 19 locales for an average word error rate reduction of 18.6% (es-ES, es-MX, fr-CA, fr-FR, it-IT, ja-JP, ko-KR, pt-BR, zh-CN, zh-HK, nb-NO, fi-FL, ru-RU, pl-PL, ca-ES, zh-TW, th-TH, pt-PT, tr-TR). The new models bring significant improvements across multiple domains including Dictation, Call-Center Transcription, and Video Indexing scenarios.

Bug fixes

- Fixed bug where Conversation Transcriber didn't await properly in JAVA APIs.
- Add missing (Get|Set)Property methods to AudioConfig.
- Fix a TTS bug where the audioDataStream couldn't be stopped when connection fails.
- Using an endpoint without a region would cause USP failures for conversation translator.
- ID generation in Universal Windows Applications now uses an appropriately unique GUID algorithm; it previously and unintentionally defaulted to a stubbed implementation that often produced collisions over large sets of interactions.

Samples

- Unity sample for using Speech SDK with [Unity microphone and push mode streaming](#)

Other changes

- [OpenSSL configuration documentation updated for Linux](#)

Speech SDK 1.9.0: 2020-January release

New features

- Multi-device conversation: connect multiple devices to the same speech or text-based conversation, and optionally translate messages sent between them. Learn more in [this article](#).
- Keyword recognition support added for Android .aar package and added support for x86 and x64 flavors.
- Objective-C: `SendMessage` and `SetMessageProperty` methods added to `Connection` object. See documentation [here](#).
- TTS C++ api now supports `std::wstring` as synthesis text input, removing the need to convert a wstring to string before passing it to the SDK. See details [here](#).
- C#: `Language ID` and `source language config` are now available.

- JavaScript: Added a feature to `Connection` object to pass through custom messages from the Speech service as callback `receivedServiceMessage`.
- JavaScript: Added support for `FromHost API` to ease use with on-premises containers and sovereign clouds. See documentation [here](#).
- JavaScript: We now honor `NODE_TLS_REJECT_UNAUTHORIZED` thanks to a contribution from [orgads](#). See details [here](#).

Breaking changes

- `OpenSSL` has been updated to version 1.1.1b and is statically linked to the Speech SDK core library for Linux. This may cause a break if your inbox `OpenSSL` hasn't been installed to the `/usr/lib/ssl` directory in the system. Check [our documentation](#) under Speech SDK docs to work around the issue.
- We've changed the data type returned for C# `WordLevelTimingResult.Offset` from `int` to `long` to allow for access to `WordLevelTimingResults` when speech data is longer than 2 minutes.
- `PushAudioInputStream` and `PullAudioInputStream` now send wav header information to the Speech service based on `AudioStreamFormat`, optionally specified when they were created. Customers must now use the [supported audio input format](#). Any other formats will get suboptimal recognition results or may cause other issues.

Bug fixes

- See the `OpenSSL` update under Breaking changes above. We fixed both an intermittent crash and a performance issue (lock contention under high load) in Linux and Java.
- Java: Made improvements to object closure in high concurrency scenarios.
- Restructured our NuGet package. We removed the three copies of `Microsoft.CognitiveServices.Speech.core.dll` and `Microsoft.CognitiveServices.Speech.extension.kws.dll` under lib folders, making the NuGet package smaller and faster to download, and we added headers needed to compile some C++ native apps.
- Fixed quickstart samples [here](#). These were exiting without displaying "microphone not found" exception on Linux, macOS, Windows.
- Fixed SDK crash with long speech recognition results on certain code paths like [this sample](#).
- Fixed SDK deployment error in Azure Web App environment to address [this customer issue](#).
- Fixed a TTS error while using multi `<voice>` tag or `<audio>` tag to address [this customer issue](#).

- Fixed a TTS 401 error when the SDK is recovered from suspended.
- JavaScript: Fixed a circular import of audio data thanks to a contribution from [euirim](#).
- JavaScript: added support for setting service properties, as added in 1.7.
- JavaScript: fixed an issue where a connection error could result in continuous, unsuccessful websocket reconnect attempts.

Samples

- Added keyword recognition sample for Android [here](#).
- Added TTS sample for the server scenario [here](#).
- Added Multi-device conversation quickstarts for C# and C++ [here](#).

Other changes

- Optimized SDK core library size on Android.
- SDK in 1.9.0 and onwards supports both `int` and `string` types in the voice signature version field for Conversation Transcriber.

Speech SDK 1.8.0: 2019-November release

New features

- Added a `FromHost()` API, to ease use with on-premises containers and sovereign clouds.
- Added Source Language Identification for Speech Recognition (in Java and C++)
- Added `SourceLanguageConfig` object for Speech Recognition, used to specify expected source languages (in Java and C++)
- Added `KeywordRecognizer` support on Windows (UWP), Android and iOS through the NuGet and Unity packages
- Added Remote Conversation Java API to do Conversation Transcription in asynchronous batches.

Breaking changes

- Conversation Transcriber functionalities moved under namespace `Microsoft.CognitiveServices.Speech.Transcription`.
- Parts of the Conversation Transcriber methods are moved to new `Conversation` class.
- Dropped support for 32-bit (ARMv7 and x86) iOS

Bug fixes

- Fix for crash if local `KeywordRecognizer` is used without a valid Speech service subscription key

Samples

- Xamarin sample for `KeywordRecognizer`
- Unity sample for `KeywordRecognizer`
- C++ and Java samples for Automatic Source Language Identification.

Speech SDK 1.7.0: 2019-September release

New features

- Added beta support for Xamarin on Universal Windows Platform (UWP), Android, and iOS
- Added iOS support for Unity
- Added `Compressed` input support for ALaw, Mulaw, FLAC, on Android, iOS, and Linux
- Added `SendMessageAsync` in `Connection` class for sending a message to service
- Added `SetMessageProperty` in `Connection` class for setting property of a message
- TTS added bindings for Java (JRE and Android), Python, Swift, and Objective-C
- TTS added playback support for macOS, iOS, and Android.
- Added "word boundary" information for TTS.

Bug fixes

- Fixed IL2CPP build issue on Unity 2019 for Android
- Fixed issue with malformed headers in wav file input being processed incorrectly
- Fixed issue with UUIDs not being unique in some connection properties
- Fixed a few warnings about nullability specifiers in the Swift bindings (might require small code changes)
- Fixed a bug that caused websocket connections to be closed ungracefully under network load
- Fixed an issue on Android that sometimes results in duplicate impression IDs used by `DialogServiceConnector`
- Improvements to the stability of connections across multi-turn interactions and the reporting of failures (via `Canceled` events) when they occur with `DialogServiceConnector`
- `DialogServiceConnector` session starts will now properly provide events, including when calling `ListenOnceAsync()` during an active `StartKeywordRecognitionAsync()`

- Addressed a crash associated with `DialogServiceConnector` activities being received

Samples

- Quickstart for Xamarin
- Updated CPP Quickstart with Linux Arm64 information
- Updated Unity quickstart with iOS information

Speech SDK 1.6.0: 2019-June release

Samples

- Quickstart samples for Text To Speech on UWP and Unity
- Quickstart sample for Swift on iOS
- Unity samples for Speech & Intent Recognition and Translation
- Updated quickstart samples for `DialogServiceConnector`

Improvements / Changes

- Dialog namespace:
 - `SpeechBotConnector` has been renamed to `DialogServiceConnector`
 - `BotConfig` has been renamed to `DialogServiceConfig`
 - `BotConfig::FromChannelSecret()` has been remapped to
`DialogServiceConfig::FromBotSecret()`
 - All existing Direct Line Speech clients continue to be supported after the rename
- Update TTS REST adapter to support proxy, persistent connection
- Improve error message when an invalid region is passed
- Swift/Objective-C:
 - Improved error reporting: Methods that can result in an error are now present in two versions: One that exposes an `NSError` object for error handling, and one that raises an exception. The former are exposed to Swift. This change requires adaptations to existing Swift code.
 - Improved event handling

Bug fixes

- Fix for TTS: where `SpeakTextAsync` future returned without waiting until audio has completed rendering
- Fix for marshaling strings in C# to enable full language support

- Fix for .NET core app problem to load core library with net461 target framework in samples
- Fix for occasional issues to deploy native libraries to the output folder in samples
- Fix for web socket closing reliably
- Fix for possible crash while opening a connection under heavy load on Linux
- Fix for missing metadata in the framework bundle for macOS
- Fix for problems with `pip install --user` on Windows

Speech SDK 1.5.1

This is a bug fix release and only affecting the native/managed SDK. It isn't affecting the JavaScript version of the SDK.

Bug fixes

- Fix FromSubscription when used with Conversation Transcription.
- Fix bug in keyword spotting for Voice Assistants.

Speech SDK 1.5.0: 2019-May release

New features

- Keyword spotting (KWS) is now available for Windows and Linux. KWS functionality might work with any microphone type, official KWS support, however, is currently limited to the microphone arrays found in the Azure Kinect DK hardware or the Speech Devices SDK.
- Phrase hint functionality is available through the SDK. For more information, see [here](#).
- Conversation transcription functionality is available through the SDK.
- Add support for Voice Assistants using the Direct Line Speech channel.

Samples

- Added samples for new features or new services supported by the SDK.

Improvements / Changes

- Added various recognizer properties to adjust service behavior or service results (like masking profanity and others).

- You can now configure the recognizer through the standard configuration properties, even if you created the recognizer `FromEndpoint`.
- Objective-C: `OutputFormat` property was added to `SPXSpeechConfiguration`.
- The SDK now supports Debian 9 as a Linux distribution.

Bug fixes

- Fixed a problem where the speaker resource was destructed too early in text to speech.

Speech SDK 1.4.2

This is a bug fix release and only affecting the native/managed SDK. It isn't affecting the JavaScript version of the SDK.

Speech SDK 1.4.1

This is a JavaScript-only release. No features have been added. The following fixes were made:

- Prevent web pack from loading `https-proxy-agent`.

Speech SDK 1.4.0: 2019-April release

New features

- The SDK now supports the Text to speech service as a beta version. It's supported on Windows and Linux Desktop from C++ and C#. For more information, check the [Text to speech overview](#).
- The SDK now supports MP3 and Opus/OGG audio files as stream input files. This feature is available only on Linux from C++ and C# and is currently in beta (more details [here](#)).
- The Speech SDK for Java, .NET core, C++ and Objective-C have gained macOS support. The Objective-C support for macOS is currently in beta.
- iOS: The Speech SDK for iOS (Objective-C) is now also published as a CocoaPod.
- JavaScript: Support for non-default microphone as an input device.
- JavaScript: Proxy support for Node.js.

Samples

- Samples for using the Speech SDK with C++ and with Objective-C on macOS have been added.
- Samples demonstrating the usage of the Text to speech service have been added.

Improvements / Changes

- Python: Additional properties of recognition results are now exposed via the `properties` property.
- For additional development and debug support, you can redirect SDK logging and diagnostics information into a log file (more details [here](#)).
- JavaScript: Improve audio processing performance.

Bug fixes

- Mac/iOS: A bug that led to a long wait when a connection to the Speech service couldn't be established was fixed.
- Python: improve error handling for arguments in Python callbacks.
- JavaScript: Fixed wrong state reporting for speech ended on RequestSession.

Speech SDK 1.3.1: 2019-February refresh

This is a bug fix release and only affecting the native/managed SDK. It isn't affecting the JavaScript version of the SDK.

Bug fix

- Fixed a memory leak when using microphone input. Stream based or file input isn't affected.

Speech SDK 1.3.0: 2019-February release

New features

- The Speech SDK supports selection of the input microphone through the `AudioConfig` class. This allows you to stream audio data to the Speech service from a non-default microphone. For more information, see the documentation describing [audio input device selection](#). This feature isn't yet available from JavaScript.
- The Speech SDK now supports Unity in a beta version. Provide feedback through the issue section in the [GitHub sample repository](#). This release supports Unity on Windows x86 and x64 (desktop or Universal Windows Platform applications), and Android (ARM32/64, x86). More information is available in our [Unity quickstart](#).

- The file `Microsoft.CognitiveServices.Speech.csharp.bindings.dll` (shipped in previous releases) isn't needed anymore. The functionality is now integrated into the core SDK.

Samples

The following new content is available in our [sample repository](#):

- Additional samples for `AudioConfig.FromMicrophoneInput`.
- Additional Python samples for intent recognition and translation.
- Additional samples for using the `Connection` object in iOS.
- Additional Java samples for translation with audio output.
- New sample for use of the [Batch Transcription REST API](#).

Improvements / Changes

- Python
 - Improved parameter verification and error messages in `SpeechConfig`.
 - Add support for the `Connection` object.
 - Support for 32-bit Python (x86) on Windows.
 - The Speech SDK for Python is out of beta.
- iOS
 - The SDK is now built against the iOS SDK version 12.1.
 - The SDK now supports iOS versions 9.2 and later.
 - Improve reference documentation and fix several property names.
- JavaScript
 - Add support for the `Connection` object.
 - Add type definition files for bundled JavaScript
 - Initial support and implementation for phrase hints.
 - Return properties collection with service JSON for recognition
- Windows DLLs do now contain a version resource.
- If you create a recognizer `FromEndpoint`, you can add parameters directly to the endpoint URL. Using `FromEndpoint` you can't configure the recognizer through the standard configuration properties.

Bug fixes

- Empty proxy username and proxy password weren't handled correctly. With this release, if you set proxy username and proxy password to an empty string, they won't be submitted when connecting to the proxy.

- SessionId's created by the SDK weren't always truly random for some languages / environments. Added random generator initialization to fix this issue.
- Improve handling of authorization token. If you want to use an authorization token, specify in the `SpeechConfig` and leave the API key empty. Then create the recognizer as usual.
- In some cases, the `Connection` object wasn't released correctly. This issue has been fixed.
- The JavaScript sample was fixed to support audio output for translation synthesis also on Safari.

Speech SDK 1.2.1

This is a JavaScript-only release. No features have been added. The following fixes were made:

- Fire end of stream at `turn.end`, not at `speech.end`.
- Fix bug in audio pump that didn't schedule next send if the current send failed.
- Fix continuous recognition with auth token.
- Bug fix for different recognizer / endpoints.
- Documentation improvements.

Speech SDK 1.2.0: 2018-December release

New features

- Python
 - The Beta version of Python support (3.5 and above) is available with this release. For more information, see [here](#)([..../quickstart-python.md](#)).
- JavaScript
 - The Speech SDK for JavaScript has been open-sourced. The source code is available on [GitHub](#) ↗.
 - We now support Node.js, more info can be found [here](#).
 - The length restriction for audio sessions has been removed, reconnection will happen automatically under the cover.
- `Connection` object
 - From the `Recognizer`, you can access a `Connection` object. This object allows you to explicitly initiate the service connection and subscribe to connect and disconnect events. (This feature isn't yet available from JavaScript and Python.)
- Support for Ubuntu 18.04.
- Android

- Enabled ProGuard support during APK generation.

Improvements

- Improvements in the internal thread usage, reducing the number of threads, locks, mutexes.
- Improved error reporting / information. In several cases, error messages haven't been propagated out all the way out.
- Updated development dependencies in JavaScript to use up-to-date modules.

Bug fixes

- Fixed memory leaks due to a type mismatch in `RecognizeAsync`.
- In some cases exceptions were being leaked.
- Fixing memory leak in translation event arguments.
- Fixed a locking issue on reconnect in long running sessions.
- Fixed an issue that could lead to missing final result for failed translations.
- C#: If an `async` operation wasn't awaited in the main thread, it was possible the recognizer could be disposed before the async task was completed.
- Java: Fixed a problem resulting in a crash of the Java VM.
- Objective-C: Fixed enum mapping; `RecognizedIntent` was returned instead of `RecognizingIntent`.
- JavaScript: Set default output format to 'simple' in `SpeechConfig`.
- JavaScript: Removing inconsistency between properties on the config object in JavaScript and other languages.

Samples

- Updated and fixed several samples (for example output voices for translation, etc.).
- Added Node.js samples in the [sample repository ↗](#).

Speech SDK 1.1.0

New features

- Support for Android x86/x64.
- Proxy Support: In the `SpeechConfig` object, you can now call a function to set the proxy information (hostname, port, username, and password). This feature isn't yet available on iOS.

- Improved error code and messages. If a recognition returned an error, this did already set `Reason` (in canceled event) or `CancellationDetails` (in recognition result) to `Error`. The canceled event now contains two additional members, `ErrorCode` and `ErrorDetails`. If the server returned additional error information with the reported error, it will now be available in the new members.

Improvements

- Added additional verification in the recognizer configuration, and added additional error message.
- Improved handling of long-time silence in middle of an audio file.
- NuGet package: for .NET Framework projects, it prevents building with AnyCPU configuration.

Bug fixes

- Fixed several exceptions found in recognizers. In addition, exceptions are caught and converted into `Canceled` event.
- Fix a memory leak in property management.
- Fixed bug in which an audio input file could crash the recognizer.
- Fixed a bug where events could be received after a session stop event.
- Fixed some race conditions in threading.
- Fixed an iOS compatibility issue that could result in a crash.
- Stability improvements for Android microphone support.
- Fixed a bug where a recognizer in JavaScript would ignore the recognition language.
- Fixed a bug preventing setting the `EndpointId` (in some cases) in JavaScript.
- Changed parameter order in `AddIntent` in JavaScript, and added missing `AddIntent` JavaScript signature.

Samples

- Added C++ and C# samples for pull and push stream usage in the [sample repository](#).

Speech SDK 1.0.1

Reliability improvements and bug fixes:

- Fixed potential fatal error due to race condition in disposing recognizer
- Fixed potential fatal error when unset properties occur.

- Added additional error and parameter checking.
- Objective-C: Fixed possible fatal error caused by name overriding in NSString.
- Objective-C: Adjusted visibility of API
- JavaScript: Fixed regarding events and their payloads.
- Documentation improvements.

In our [sample repository](#), a new sample for JavaScript was added.

Azure AI Speech SDK 1.0.0: 2018-September release

New features

- Support for Objective-C on iOS. Check out our [Objective-C quickstart for iOS](#).
- Support for JavaScript in browser. Check out our [JavaScript quickstart](#).

Breaking changes

- With this release, a number of breaking changes are introduced. Check [this page](#) for details.

Azure AI Speech SDK 0.6.0: 2018-August release

New features

- UWP apps built with the Speech SDK now can pass the Windows App Certification Kit (WACK). Check out the [UWP quickstart](#).
- Support for .NET Standard 2.0 on Linux (Ubuntu 16.04 x64).
- Experimental: Support Java 8 on Windows (64-bit) and Linux (Ubuntu 16.04 x64). Check out the [Java Runtime Environment quickstart](#).

Functional change

- Expose additional error detail information on connection errors.

Breaking changes

- On Java (Android), the `SpeechFactory.configureNativePlatformBindingWithDefaultCertificate` function no longer requires a path parameter. Now the path is automatically detected on all supported platforms.
- The get-accessor of the property `EndpointUrl` in Java and C# was removed.

Bug fixes

- In Java, the audio synthesis result on the translation recognizer is implemented now.
- Fixed a bug that could cause inactive threads and an increased number of open and unused sockets.
- Fixed a problem, where a long-running recognition could terminate in the middle of the transmission.
- Fixed a race condition in recognizer shutdown.

Azure AI Speech SDK 0.5.0: 2018-July release

New features

- Support Android platform (API 23: Android 6.0 Marshmallow or higher). Check out the [Android quickstart](#).
- Support .NET Standard 2.0 on Windows. Check out the [.NET Core quickstart](#).
- Experimental: Support UWP on Windows (version 1709 or later).
 - Check out the [UWP quickstart](#).
 - Note that UWP apps built with the Speech SDK don't yet pass the Windows App Certification Kit (WACK).
- Support long-running recognition with automatic reconnection.

Functional changes

- `StartContinuousRecognitionAsync()` supports long-running recognition.
- The recognition result contains more fields. They're offset from the audio beginning and duration (both in ticks) of the recognized text and additional values that represent recognition status, for example, `InitialSilenceTimeout` and `InitialBabbleTimeout`.
- Support `AuthorizationToken` for creating factory instances.

Breaking changes

- Recognition events: `NoMatch` event type was merged into the `Error` event.
- `SpeechOutputFormat` in C# was renamed to `OutputFormat` to stay aligned with C++.
- The return type of some methods of the `AudioInputStream` interface changed slightly:
 - In Java, the `read` method now returns `long` instead of `int`.
 - In C#, the `Read` method now returns `uint` instead of `int`.
 - In C++, the `Read` and `GetFormat` methods now return `size_t` instead of `int`.
- C++: Instances of audio input streams now can be passed only as a `shared_ptr`.

Bug fixes

- Fixed incorrect return values in the result when `RecognizeAsync()` times out.
- The dependency on media foundation libraries on Windows was removed. The SDK now uses Core Audio APIs.
- Documentation fix: Added a [regions](#) page to describe the supported regions.

Known Issue

- The Speech SDK for Android doesn't report speech synthesis results for translation. This issue will be fixed in the next release.

Azure AI Speech SDK 0.4.0: 2018-June release

Functional changes

- `AudioInputStream`

A recognizer now can consume a stream as the audio source. For more information, see the related [how-to guide](#).

- Detailed output format

When you create a `SpeechRecognizer`, you can request `Detailed` or `Simple` output format. The `DetailedSpeechRecognitionResult` contains a confidence score, recognized text, raw lexical form, normalized form, and normalized form with masked profanity.

Breaking change

- Changed to `SpeechRecognitionResult.Text` from `SpeechRecognitionResult.RecognizedText` in C#.

Bug fixes

- Fixed a possible callback issue in the USP layer during shutdown.
- If a recognizer consumed an audio input file, it was holding on to the file handle longer than necessary.
- Removed several deadlocks between the message pump and the recognizer.
- Fired a `NoMatch` result when the response from service is timed out.
- The media foundation libraries on Windows are delay loaded. This library is required for microphone input only.

- The upload speed for audio data is limited to about twice the original audio speed.
- On Windows, C# .NET assemblies now are strong named.
- Documentation fix: `Region` is required information to create a recognizer.

More samples have been added and are constantly being updated. For the latest set of samples, see the [Speech SDK samples GitHub repository](#).

Azure AI Speech SDK 0.2.12733: 2018-May release

This release is the first public preview release of the Azure AI Speech SDK.

Last updated on 11/05/2025

Language and voice support for the Speech service

The following tables summarize language support for [speech to text](#), [text to speech](#), [pronunciation assessment](#), [speech translation](#), and more service features.

You can also get a list of locales and voices supported for each specific region or endpoint via:

- [Speech SDK](#)
- [Speech to text REST API](#)
- [Speech to text REST API for short audio](#)
- [Text to speech REST API](#)

Supported languages

Language support varies by Speech service functionality.

Note

See [speech containers](#) and [embedded speech](#) documentation for their supported languages.

Choose a Speech feature

 [Speech to text](#)

The table in this section summarizes the locales supported for [real-time transcription](#), [fast transcription](#), and [batch transcription](#) transcription.

More remarks for speech to text locales are included in the [custom speech](#) section of this article.

Tip

Try out the [Azure AI Speech Toolkit](#)  to easily build and run samples on Visual Studio Code.

 [Expand table](#)

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
af-ZA	Afrikaans (South Africa)	Yes	Plain text
am-ET	Amharic (Ethiopia)	Yes	Plain text
ar-AE	Arabic (United Arab Emirates)	Yes	Audio + human-labeled transcript Plain text
ar-BH	Arabic (Bahrain)	Yes	Audio + human-labeled transcript Plain text
ar-DZ	Arabic (Algeria)	No	Audio + human-labeled transcript Plain text
ar-EG	Arabic (Egypt)	Yes	Audio + human-labeled transcript Plain text Structured text
ar-IL	Arabic (Israel)	Yes	Audio + human-labeled transcript Plain text
ar-IQ	Arabic (Iraq)	Yes	Audio + human-labeled transcript Plain text
ar-JO	Arabic (Jordan)	Yes	Audio + human-labeled transcript Plain text
ar-KW	Arabic (Kuwait)	Yes	Audio + human-labeled transcript Plain text
ar-LB	Arabic (Lebanon)	Yes	Audio + human-labeled transcript

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Plain text
ar-LY	Arabic (Libya)	Yes	Audio + human-labeled transcript
			Plain text
ar-MA	Arabic (Morocco)	No	Audio + human-labeled transcript
			Plain text
ar-OM	Arabic (Oman)	Yes	Audio + human-labeled transcript
			Plain text
ar-PS	Arabic (Palestinian Authority)	Yes	Audio + human-labeled transcript
			Plain text
ar-QA	Arabic (Qatar)	Yes	Audio + human-labeled transcript
			Plain text
ar-SA	Arabic (Saudi Arabia)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Phrase list
ar-SY	Arabic (Syria)	Yes	Audio + human-labeled transcript
			Plain text
ar-TN	Arabic (Tunisia)	No	Audio + human-labeled transcript
			Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
ar-YE	Arabic (Yemen)	No	Audio + human-labeled transcript Plain text
as-IN	Assamese (India)	No	Audio + human-labeled transcript
az-AZ	Azerbaijani (Latin, Azerbaijan)	Yes	Plain text
bg-BG	Bulgarian (Bulgaria)	No	Plain text
bn-IN	Bengali (India)	Yes	Plain text
bs-BA	Bosnian (Bosnia and Herzegovina)	No	Plain text
ca-ES	Catalan	No	Plain text Pronunciation
cs-CZ	Czech (Czechia)	Yes	Audio + human-labeled transcript Plain text Structured text Pronunciation
cy-GB	Welsh (United Kingdom)	No	Plain text
da-DK	Danish (Denmark)	Yes	Audio + human-labeled transcript Plain text Structured text Output format Pronunciation
de-AT	German (Austria)	Yes	Audio + human-labeled transcript Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Structured text
			Pronunciation
de-CH	German (Switzerland)	Yes	Audio + human-labeled transcript
			Plain text
			Pronunciation
			Phrase list
de-DE	German (Germany)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
el-GR	Greek (Greece)	No	Audio + human-labeled transcript
			Plain text
			Structured text
en-AU	English (Australia)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
en-CA	English (Canada)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Output format Pronunciation Phrase list
en-GB	English (United Kingdom)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Output format Pronunciation Phrase list
en-GH	English (Ghana)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Pronunciation
en-HK	English (Hong Kong SAR)	Yes	Audio + human-labeled transcript Audio Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Structured text
			Output format
			Pronunciation
en-IE	English (Ireland)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
en-IN	English (India)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
en-KE	English (Kenya)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Pronunciation

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
en-NG	English (Nigeria)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Output format Pronunciation
en-NZ	English (New Zealand)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Output format Pronunciation
en-PH	English (Philippines)	Yes	Audio + human-labeled transcript Audio Plain text Structured text Output format Pronunciation
en-SG	English (Singapore)	Yes	Audio + human-labeled transcript Audio Plain text Structured text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Output format
			Pronunciation
en-TZ	English (Tanzania)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Pronunciation
en-US	English (United States)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
en-ZA	English (South Africa)	Yes	Audio + human-labeled transcript
			Audio
			Plain text
			Structured text
			Pronunciation
			Phrase list
es-AR	Spanish (Argentina)	Yes	Plain text
			Structured text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Pronunciation
es-BO	Spanish (Bolivia)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-CL	Spanish (Chile)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-CO	Spanish (Colombia)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-CR	Spanish (Costa Rica)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-CU	Spanish (Cuba)	Yes	Plain text
			Structured text
			Pronunciation
es-DO	Spanish (Dominican Republic)	Yes	Plain text
			Structured text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Pronunciation
es-EC	Spanish (Ecuador)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-ES	Spanish (Spain)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
es-GQ	Spanish (Equatorial Guinea)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
es-GT	Spanish (Guatemala)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-HN	Spanish (Honduras)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Pronunciation
es-MX	Spanish (Mexico)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
es-NI	Spanish (Nicaragua)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-PA	Spanish (Panama)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-PE	Spanish (Peru)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-PR	Spanish (Puerto Rico)	Yes	Audio + human-labeled transcript
			Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Structured text
			Pronunciation
es-PY	Spanish (Paraguay)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-SV	Spanish (El Salvador)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-US	Spanish (United States) ¹	Yes	Plain text
			Structured text
			Pronunciation
			Phrase list
es-UY	Spanish (Uruguay)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
es-VE	Spanish (Venezuela)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
et-EE	Estonian (Estonia)	Yes	Plain text
			Pronunciation
eu-ES	Basque	Yes	Plain text
fa-IR	Persian (Iran)	No	Plain text
fi-FI	Finnish (Finland)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
fil-PH	Filipino (Philippines)	Yes	Plain text
			Pronunciation
fr-BE	French (Belgium)	No	Plain text
fr-CA	French (Canada) ¹	No	Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
fr-CH	French (Switzerland)	No	Plain text
			Pronunciation
fr-FR	French (France)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Pronunciation
			Phrase list
ga-IE	Irish (Ireland)	Yes	Plain text
			Pronunciation
gl-ES	Galician	Yes	Plain text
gu-IN	Gujarati (India)	No	Plain text
he-IL	Hebrew (Israel)	Yes	Audio + human-labeled transcript
			Plain text
hi-IN	Hindi (India)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
hr-HR	Croatian (Croatia)	No	Plain text
			Pronunciation
hu-HU	Hungarian (Hungary)	No	Audio + human-labeled transcript
			Plain text
			Structured text
			Pronunciation
hy-AM	Armenian (Armenia)	Yes	Plain text
id-ID	Indonesian (Indonesia)	Yes	Audio + human-labeled transcript
			Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Structured text
			Pronunciation
			Phrase list
is-IS	Icelandic (Iceland)	Yes	Plain text
it-CH	Italian (Switzerland)	No	Plain text
it-IT	Italian (Italy)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
ja-JP	Japanese (Japan)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
jv-ID	Javanese (Latin, Indonesia)	Yes	Plain text
ka-GE	Georgian (Georgia)	Yes	Plain text
kk-KZ	Kazakh (Kazakhstan)	Yes	Plain text
km-KH	Khmer (Cambodia)	No	Plain text
kn-IN	Kannada (India)	No	Plain text
ko-KR	Korean (Korea)	Yes	Audio + human-labeled transcript
			Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Structured text
			Output format
			Phrase list
lo-LA	Lao (Laos)	Yes	Plain text
lt-LT	Lithuanian (Lithuania)	Yes	Plain text
			Pronunciation
lv-LV	Latvian (Latvia)	No	Plain text
			Pronunciation
mk-MK	Macedonian (North Macedonia)	Yes	Plain text
ml-IN	Malayalam (India)	Yes	Plain text
mn-MN	Mongolian (Mongolia)	Yes	Plain text
mr-IN	Marathi (India)	No	Audio + human-labeled transcript
			Plain text
ms-MY	Malay (Malaysia)	No	Plain text
mt-MT	Maltese (Malta)	Yes	Plain text
my-MM	Burmese (Myanmar)	Yes	Plain text
nb-NO	Norwegian Bokmål (Norway)	No	Plain text
			Output format
ne-NP	Nepali (Nepal)	No	Plain text
nl-BE	Dutch (Belgium)	No	Plain text
nl-NL	Dutch (Netherlands)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Output format
			Pronunciation
			Phrase list
or-IN	Odia (India)	No	Audio + human-labeled transcript
pa-IN	Punjabi (India)	No	Audio + human-labeled transcript
pl-PL	Polish (Poland)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
ps-AF	Pashto (Afghanistan)	Yes	Plain text
pt-BR	Portuguese (Brazil)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
pt-PT	Portuguese (Portugal)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Pronunciation
			Phrase list
ro-RO	Romanian (Romania)	No	Plain text
			Pronunciation
ru-RU	Russian (Russia)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Phrase list
si-LK	Sinhala (Sri Lanka)	No	Plain text
sk-SK	Slovak (Slovakia)	No	Plain text
			Pronunciation
sl-SI	Slovenian (Slovenia)	No	Plain text
			Pronunciation
so-SO	Somali (Somalia)	Yes	Plain text
sq-AL	Albanian (Albania)	Yes	Plain text
sr-RS	Serbian (Cyrillic, Serbia)	Yes	Plain text
sv-SE	Swedish (Sweden)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Pronunciation
			Phrase list
sw-KE	Kiswahili (Kenya)	Yes	Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
sw-TZ	Kiswahili (Tanzania)	No	Plain text
ta-IN	Tamil (India)	No	Audio + human-labeled transcript
			Plain text
			Structured text
te-IN	Telugu (India)	No	Audio + human-labeled transcript
			Plain text
			Structured text
th-TH	Thai (Thailand)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
tr-TR	Turkish (Türkiye)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
uk-UA	Ukrainian (Ukraine)	No	Audio + human-labeled transcript
			Plain text
ur-IN	Urdu (India)	Yes	Audio + human-labeled transcript
uz-UZ	Uzbek (Latin, Uzbekistan)	Yes	Plain text
vi-VN	Vietnamese (Vietnam)	No	Plain text

Locale (BCP-47)	Language	Fast transcription support	Custom speech support
			Phrase list
wuu-CN	Chinese (Wu, Simplified)	No	Plain text
yue-CN	Chinese (Cantonese, Simplified)	No	Plain text
zh-CN	Chinese (Mandarin, Simplified)	Yes	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
zh-CN-shandong	Chinese (Jilu Mandarin, Simplified)	No	Plain text
zh-CN-sichuan	Chinese (Southwestern Mandarin, Simplified)	No	Plain text
zh-HK	Chinese (Cantonese, Traditional)	No	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
zh-TW	Chinese (Taiwanese Mandarin, Traditional)	No	Audio + human-labeled transcript
			Plain text
			Structured text
			Output format
			Phrase list
zu-ZA	isiZulu (South Africa)	Yes	Plain text

¹ The model is bilingual and also supports English.

Custom speech

To improve speech to text recognition accuracy, customization is available for some languages and base models. Depending on the locale, you can upload audio + human-labeled transcripts, plain text, structured text, and pronunciation data. By default, plain text customization is supported for all available base models. To learn more about customization, see [custom speech](#).

These locales support the [display text format feature](#): da-DK, de-DE, en-AU, en-CA, en-GB, en-HK, en-IE, en-IN, en-NG, en-NZ, en-PH, en-SG, en-US, es-ES, es-MX, fi-FI, fr-CA, fr-FR, hi-IN, it-IT, ja-JP, ko-KR, nb-NO, nl-NL, pl-PL, pt-BR, pt-PT, sv-SE, tr-TR, zh-CN, zh-HK.

Next steps

- [Region support](#)

Last updated on 11/06/2025

Speech service supported regions

The Speech service allows your application to convert audio to text, perform speech translation, and convert text to speech. The service is available in multiple regions with unique endpoints for the Speech SDK and REST APIs.

Keep in mind the following points:

- If your application uses a [Speech SDK](#), you provide the region identifier, such as `westus`, when you create a `SpeechConfig`. Make sure the region matches the region of your Speech resource.
- If your application uses one of the Speech service REST APIs, the region is part of the endpoint URI you use when making requests.
- Keys created for a region are valid only in that region. If you attempt to use them with other regions, you get authentication errors.

(!) Note

Speech service doesn't store or process your data outside the region of your Speech resource. The data is only stored or processed in the region where the resource is created. For example, if you create an AI Foundry resource for Speech in the `westus` region, the data is only in the `westus` region.

Regions

The regions in these tables support most of the core features of the Speech service, such as speech to text, text to speech, and translation. Some features, such as fast transcription and batch synthesis require specific regions. For the features that require specific regions, the according tables indicate the regions that support them.

Geographies

[+] Expand table

Geography	Region	Region identifier
Africa	South Africa North	<code>southafricanorth</code>
Asia Pacific	East Asia	<code>eastasia</code>
Asia Pacific	Southeast Asia	<code>southeastasia</code>

Geography	Region	Region identifier
Asia Pacific	Australia East	australiaeast
Asia Pacific	Central India	centralindia
Asia Pacific	Japan East	japaneast
Asia Pacific	Japan West	japanwest
Asia Pacific	Korea Central	koreacentral
Canada	Canada Central	canadacentral
Canada	Canada East	canadaeast
Europe	North Europe	northeurope
Europe	West Europe	westeurope
Europe	France Central	francecentral
Europe	Germany West Central	germanywestcentral
Europe	Italy North	italynorth
Europe	Norway East	norwayeast
Europe	Sweden Central	swedencentral
Europe	Switzerland North	switzerlandnorth
Europe	Switzerland West	switzerlandwest
Europe	UK South	uksouth
Europe	UK West	ukwest
Middle East	UAE North	uaenorth
South America	Brazil South	brazilsouth
Qatar	Qatar Central	qatarcentral
US	Central US	centralus
US	East US	eastus
US	East US 2	eastus2
US	North Central US	northcentralus
US	South Central US	southcentralus

Geography	Region	Region identifier
US	West Central US	westcentralus
US	West US	westus
US	West US 2	westus2
US	West US 3	westus3

 **Note**

The following regions supported by a resource of kind AI Services, are currently not supported for Speech processing: `southindia`, `spaincentral`.

Related content

- [Language and voice support](#)
- [Quotas and limits](#)

Last updated on 10/31/2025

Speech service quotas and limits

09/17/2025

This article contains a quick reference and a detailed description of the quotas and limits for the Speech service in Azure AI services. The information applies to all [pricing tiers](#) of the service. It also contains some best practices to avoid request throttling.

For the free (F0) pricing tier, see also the monthly allowances at the [pricing page](#).

Quotas and limits reference

The following sections provide you with a quick guide to the quotas and limits that apply to the Speech service.

For information about adjustable quotas for Standard (S0) Speech resources, see [more explanations](#), [best practices](#), and [adjustment instructions](#). The quotas and limits for Free (F0) Speech resources aren't adjustable.

 **Important**

If you switch an AI Foundry resource for Speech from Free (F0) to Standard (S0) pricing tier, the change of the corresponding quotas may take up to several hours.

Speech to text quotas and limits per resource

This section describes speech to text quotas and limits per Speech resource. Unless otherwise specified, the limits aren't adjustable.

Real-time speech to text and speech translation

You can use real-time speech to text with the [Speech SDK](#) or the [Speech to text REST API](#) for [short audio](#).

 **Important**

These limits apply to concurrent real-time speech to text requests and speech translation requests combined. For example, if you have 60 concurrent speech to text requests and 40 concurrent speech translation requests, you'll reach the limit of 100 concurrent requests.

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
Concurrent request limit - base model endpoint	1 This limit isn't adjustable.	100 (default value) The rate is adjustable for Standard (S0) resources. See more explanations , best practices , and adjustment instructions .
Concurrent request limit - custom endpoint	1 This limit isn't adjustable.	100 (default value) The rate is adjustable for Standard (S0) resources. See more explanations , best practices , and adjustment instructions .
Max audio length for real-time diarization .	N/A	240 minutes per file

Fast transcription

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
Maximum audio input file size	N/A	< 300 MB
Maximum audio length	N/A	< 120 minutes per file
Maximum requests per minute	N/A	600

Batch transcription

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
Speech to text REST API limit	Not available for F0	100 requests per 10 seconds (600 requests per minute)
Max audio input file size	N/A	1 GB
Max number of blobs per container	N/A	10000
Max number of files per transcription request (when you're using multiple content URLs as input).	N/A	1000

Quota	Free (F0)	Standard (S0)
Max audio length for transcriptions with diarization enabled.	N/A	240 minutes per file

Model customization

The limits in this table apply per Speech resource when you create a custom speech model.

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
REST API limit	100 requests per 10 seconds (600 requests per minute)	100 requests per 10 seconds (600 requests per minute)
Max number of custom model deployments per Speech resource	1	50
Max number of speech datasets	2	500
Max acoustic dataset file size for data import	2 GB	2 GB
Max language dataset file size for data import	200 MB	1.5 GB
Max pronunciation dataset file size for data import	1 KB	1 MB
Max text size when you're using the <code>text</code> parameter in the Models_Create API request	200 KB	500 KB

Text to speech quotas and limits per resource

This section describes text to speech quotas and limits per Speech resource.

Real-time text to speech

You can use real-time text to speech with the [Speech SDK](#) or the [Text to speech REST API](#). Unless otherwise specified, the limits aren't adjustable.

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
Maximum number of transactions per time period for standard voices and custom voices.	20 transactions per 60 seconds This limit isn't adjustable.	200 transactions per second (TPS) (default value) The rate is adjustable up to 1000 TPS for Standard (S0) resources. See more explanations , best practices , and adjustment instructions .
Max audio length produced per request	10 min	10 min
Max total number of distinct <code><voice></code> and <code><audio></code> tags in SSML	50	50
Max SSML message size per turn for websocket	64 KB	64 KB

ⓘ Note

Most HTTP 429 errors with Text-to-Speech Standard Voice are caused by limited backend service capacity for a specific voice in the selected region, not by quota limits. Increasing your quota won't resolve these errors. For best results, use the voice in its native region or select a more popular voice in your current region.

Batch synthesis

These limits aren't adjustable. For more information on batch synthesis latency, see [the batch synthesis latency and best practices](#).

[+] [Expand table](#)

Quota	Free (F0)	Standard (S0)
REST API limit	Not available for F0	100 requests per 10 seconds
Max JSON payload size to create a synthesis job	N/A	2 megabytes
Concurrent active synthesis jobs	N/A	No limit
Max number of text inputs per synthesis job	N/A	10000
Max time to live for a synthesis job since it being in the final state	N/A	Up to 31 days (specified using properties)

Custom voice - professional

The limits in this table apply per Speech resource when you create a professional voice.

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
Max number of transactions per second (TPS)	Not available for F0	200 transactions per second (TPS) (default value)
Max number of datasets	N/A	500
Max number of simultaneous dataset uploads	N/A	5
Max data file size for data import per dataset	N/A	2 GB
Upload of long audio or audio without script	N/A	Yes
Max number of simultaneous model trainings	N/A	4
Max number of custom endpoints	N/A	50

Custom voice - personal voice

The limits in this table apply per Speech resource when you create a personal voice.

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
REST API limit (not including speech synthesis)	Not available for F0	50 requests per 10 seconds
Max number of transactions per second (TPS) for speech synthesis	Not available for F0	200 transactions per second (TPS) (default value)

Batch text to speech avatar

[\[+\] Expand table](#)

Quota	Free (F0)	Standard (S0)
REST API limit	Not available for F0	2 requests per 1 minute

Real-time text to speech avatar

 [Expand table](#)

Quota	Free (F0)	Standard (S0)
New connections per minute	Not available for F0	2 new connections per minute
Max connection duration with speaking	Not available for F0	30 minutes ¹
Max connection duration with idle state	Not available for F0	5 minutes

¹ To ensure continuous operation of the real-time avatar for more than 30 minutes, you can enable auto-reconnect. For information about how to set up auto-reconnect, refer to this [sample code](#) (search "auto reconnect").

Audio Content Creation tool

 [Expand table](#)

Quota	Free (F0)	Standard (S0)
File size (plain text in SSML) ¹	3,000 characters per file	20,000 characters per file
File size (lexicon file) ²	30KB per file	100KB per file
Billable characters in SSML	15,000 characters per file	100,000 characters per file
Export to audio library	1 concurrent task	N/A

¹ The limit only applies to plain text in SSML and doesn't include tags.

² The characters of lexicon file aren't charged. Only the lexicon elements in SSML are counted as billable characters. Refer to [billable characters](#) to learn more.

Speaker recognition quotas and limits per resource

Speaker recognition is limited to 20 transactions per second (TPS).

Detailed description, quota adjustment, and best practices

Some of the Speech service quotas are adjustable. This section provides more explanations, best practices, and adjustment instructions.

The following quotas are adjustable for Standard (S0) resources. The Free (F0) request limits aren't adjustable.

- Speech to text [concurrent request limit](#) for base model endpoint and custom endpoint
- Text to speech [maximum number of transactions per time period](#) for standard voices and custom voices
- Speech translation [concurrent request limit](#)

Before requesting a quota increase (where applicable), check your current TPS (transactions per second) and ensure that you need to increase the quota.

! Note

Batch transcription is an asynchronous process, and jobs are processed one-by-one in a queue. So, increasing the quota won't improve transcription performance. For performance improvements, see [Batch transcription best practices](#).

Speech service uses autoscaling technologies to bring the required computational resources in on-demand mode. At the same time, Speech service tries to keep your costs low by not maintaining an excessive amount of hardware capacity.

Let's look at an example. Suppose that your application receives response code 429, which indicates that there are too many requests. Your application receives this response even though your workload is within the limits defined by the [Quotas and limits reference](#). The most likely explanation is that Speech service is scaling up to your demand and didn't reach the required scale yet. Therefore the service doesn't immediately have enough resources to serve the request. In such cases, increasing the quota won't help. In most cases, the Speech service will scale up soon, and the issue causing response code 429 will be resolved.

General best practices to mitigate throttling during autoscaling

To minimize issues related to throttling, it's a good idea to use the following techniques:

- Implement retry logic in your application.

- Avoid sharp changes in the workload. Increase the workload gradually. For example, let's say your application is using text to speech, and your current workload is 5 TPS. The next second, you increase the load to 20 TPS (that is, four times more). Speech service immediately starts scaling up to fulfill the new load, but is unable to scale as needed within one second. Some of the requests get response code 429 (too many requests).
- Test different load increase patterns. For more information, see the [workload pattern example](#).
- Create more Speech service resources in *different* regions, and distribute the workload among them. (Creating multiple Speech service resources in the same region won't affect the performance, because all resources are served by the same backend cluster).

The next sections describe specific cases of adjusting quotas.

Example of a workload pattern best practice

Here's a general example of a good approach to take. It's meant only as a template that you can adjust as necessary for your own use.

Suppose that a Speech service resource has the concurrent request limit set to 300. Start the workload from 20 concurrent connections, and increase the load by 20 concurrent connections every 90-120 seconds. Control the service responses, and implement the logic that falls back (reduces the load) if you get too many requests (response code 429). Then, retry the load increase in one minute, and if it still doesn't work, try again in two minutes. Use a pattern of 1-2-4-4 minutes for the intervals.

Generally, it's a good idea to test the workload and the workload patterns before going to production.

Speech to text: increase real-time speech to text concurrent request limit

By default, the number of concurrent real-time speech to text and speech translation [requests combined](#) is limited to 100 per resource in the base model, and 100 per custom endpoint in the custom model. For the standard pricing tier, you can increase this amount. Before submitting the request, ensure that you're familiar with the material discussed earlier in this article, such as the best practices to mitigate throttling.

Note

Concurrent request limits for base and custom models need to be adjusted separately. You can have a Speech service resource that's associated with many custom endpoints

hosting many custom model deployments. As needed, the limit adjustments per custom endpoint must be requested separately.

Increasing the limit of concurrent requests doesn't directly affect your costs. The Speech service uses a payment model that requires that you pay only for what you use. The limit defines how high the service can scale before it starts throttle your requests.

You aren't able to see the existing value of the concurrent request limit parameter in the Azure portal, the command-line tools, or API requests. To verify the existing value, create an Azure support request.

! Note

Speech containers don't require increases of the concurrent request limit, because containers are constrained only by the CPUs of the hardware they are hosted on. Speech containers do, however, have their own capacity limitations that should be taken into account. For more information, see the [Speech containers FAQ](#).

Have the required information ready

- For the base model:
 - Speech resource ID
 - Region
- For the custom model:
 - Region
 - Custom endpoint ID

How to get information for the base model:

1. Go to the [Azure portal](#).
2. Select the Speech service resource for which you would like to increase the concurrency request limit.
3. From the **Resource Management** group, select **Properties**.
4. Copy and save the values of the following fields:
 - **Resource ID**
 - **Location** (your endpoint region)

How to get information for the custom model:

1. Go to the [Speech Studio](#) portal.
2. Sign in if necessary, and go to **Custom speech**.

3. Select your project, and go to **Deployment**.
4. Select the required endpoint.
5. Copy and save the values of the following fields:

- **Service Region** (your endpoint region)
- **Endpoint ID**

Create and submit a support request

Initiate the increase of the limit for concurrent requests for your resource, or if necessary check the current limit, by submitting a support request. Here's how:

1. Ensure you have the required information listed in the previous section.
2. Go to the [Azure portal](#).
3. Select the Speech service resource for which you would like to increase (or to check) the concurrency request limit.
4. In the **Support + troubleshooting** group, select **New support request**. A new window appears, with auto-populated information about your Azure subscription and Azure resource.
5. In **Summary**, describe what you want (for example, "Increase speech to text concurrency request limit").
6. In **Problem type**, select **Quota or Subscription issues**.
7. In **Problem subtype**, select either:
 - **Quota or concurrent requests increase** for an increase request.
 - **Quota or usage validation** to check the existing limit.
8. Select **Next: Solutions**. Proceed further with the request creation.
9. On the **Details** tab, in the **Description** field, enter the following:
 - A note that the request is about the speech to text quota.
 - Choose either the base or custom model.
 - The Azure resource information you [collected previously](#).
 - Any other required information.
10. On the **Review + create** tab, select **Create**.
11. Note the support request number in Azure portal notifications. You're contacted shortly about your request.

Text to speech: increase real-time TPS limit

For the standard pricing tier, you can increase this amount. Before submitting the request, ensure that you're familiar with the material discussed earlier in this article, such as the best

practices to mitigate throttling.

Estimating Your Needs

- **Usage Under \$10,000/month:** Typically, 32 TPS is sufficient, assuming your peak usage is within 10x of your average.
- **Default Limit:** 200 TPS is available by default, which exceeds most use cases.

Example: Call Center Scenario

If you're building a call center with 1,000 concurrent calls:

- Assume agents speak half the time.
- Average TTS response length is 5 seconds.

Required TPS: $1000 \text{ calls} / (2 \times 5 \text{ seconds}) = 100 \text{ TPS}$

Information Required for TPS Increase Request

Please provide the following details:

- **Peak TPS:**
- **Average TPS:**
- **Average TTS Request Length (in characters):**

With this data, you can estimate your monthly TTS usage with formula below:

Monthly Usage=Average TPS×Request Length×3600×24×30

Multiply the result with unit price \$15 per million characters to estimate the monthly cost.

(!) Note

If your estimated usage significantly exceeds your budget, you may be overestimating your needs.

Cost Considerations

Increasing the concurrent request limit does **not** directly affect your costs. You only pay for what you use. The limit simply defines how much the service can scale before throttling begins.

You aren't able to see the existing value of the concurrent request limit parameter in the Azure portal, the command-line tools, or API requests. To verify the existing value, create an Azure support request.

(!) Note

Speech containers don't require increases of the concurrent request limit, because containers are constrained only by the CPUs of the hardware they are hosted on.

Prepare the required information

To create an increase request, you need to provide your information.

- For the standard voice:
 - Speech resource ID
 - Region
- For the custom voice:
 - Deployment region
 - Custom endpoint ID

How to get information for the standard voice:

1. Go to the [Azure portal](#).
2. Select the Speech service resource for which you would like to increase the concurrency request limit.
3. From the **Resource Management** group, select **Properties**.
4. Copy and save the values of the following fields:
 - **Resource ID**
 - **Location** (your endpoint region)

How to get information for the custom voice:

1. Go to the [Speech Studio](#) portal.
2. Sign in if necessary, and go to **Custom voice**.
3. Select your project, and go to **Deploy model**.
4. Select the required endpoint.
5. Copy and save the values of the following fields:
 - **Service Region** (your endpoint region)
 - **Endpoint ID**

Create and submit a support request

Initiate the increase of the limit for concurrent requests for your resource, or if necessary check the current limit, by submitting a support request. Here's how:

1. Ensure you have the required information listed in the previous section.
2. Go to the [Azure portal](#).

3. Select the Speech service resource for which you would like to increase (or to check) the concurrency request limit.
4. In the **Support + troubleshooting** group, select **New support request**. A new window appears, with auto-populated information about your Azure subscription and Azure resource.
5. In **Summary**, describe what you want (for example, "Increase text to speech concurrency request limit").
6. In **Problem type**, select **Quota or Subscription issues**.
7. In **Problem subtype**, select either:
 - **Quota or concurrent requests increase** for an increase request.
 - **Quota or usage validation** to check the existing limit.
8. On the **Recommended solution** tab, select **Next**.
9. On the **Additional details** tab, fill in all the required items. And in the **Details** field, enter the following:
 - A note that the request is about the text to speech quota.
 - Choose either the standard voice or custom voice.
 - The Azure resource information you [collected previously](#).
 - Any other required information.
10. On the **Review + create** tab, select **Create**.
11. Note the support request number in Azure portal notifications. You're contacted shortly about your request.

Text to speech avatar: increase new connections limit

To increase the limit of new connections per minute for text to speech avatar, contact your sales representative to create a [ticket](#) with the following information:

- Speech resource URI
- Requested new limitation to increase to
- Justification for the increase
- Starting date for the increase
- Ending date for the increase
- Standard avatar or custom avatar

What is Speech Studio?

08/07/2025

[Speech Studio](#) is a set of UI-based tools for building and integrating features from Azure AI Speech service in your applications. You create projects in Speech Studio by using a no-code approach, and then reference those assets in your applications by using the [Speech SDK](#), the [Speech CLI](#), or the REST APIs.

💡 Tip

You can also try speech to text and text to speech in the [Azure AI Foundry portal](#) without signing up or writing any code.

Speech Studio scenarios

Explore, try out, and view sample code for some of common use cases.

- [Captioning](#): Choose a sample video clip to see real-time or offline processed captioning results. Learn how to synchronize captions with your input audio, apply profanity filters, get partial results, apply customizations, and identify spoken languages for multilingual scenarios. For more information, see the [captioning quickstart](#).
- [Call Center](#): View a demonstration on how to use the Language and Speech services to analyze call center conversations. Transcribe calls in real-time or process a batch of calls, redact personally identifying information, and extract insights such as sentiment to help with your call center use case. For more information, see the [call center quickstart](#).

For a demonstration of these scenarios in Speech Studio, view this [introductory video](#).

<https://www.youtube-nocookie.com/embed/mILVerU6DAw>

Speech Studio features

In Speech Studio, the following Speech service features are available as project types:

- [Real-time speech to text](#): Quickly test speech to text by dragging audio files here without having to use any code. Speech Studio has a demo tool for seeing how speech to text works on your audio samples. To explore the full functionality, see [What is speech to text](#).

- [Batch speech to text](#): Quickly test batch transcription capabilities to transcribe a large amount of audio in storage and receive results asynchronously. To learn more about Batch Speech-to-text, see [Batch speech to text overview](#).
- [Custom speech](#): Create speech recognition models that are tailored to specific vocabulary sets and styles of speaking. In contrast to the base speech recognition model, Custom speech models become part of your unique competitive advantage because they're not publicly accessible. To get started with uploading sample audio to create a custom speech model, see [Upload training and testing datasets](#).
- [Pronunciation assessment](#): Evaluate speech pronunciation and give speakers feedback on the accuracy and fluency of spoken audio. Speech Studio provides a sandbox for testing this feature quickly, without code. To use the feature with the Speech SDK in your applications, see the [Pronunciation assessment](#) article.
- [Speech Translation](#): Quickly test and translate speech into other languages of your choice with low latency. To explore the full functionality, see [What is speech translation](#).
- [Voice Gallery](#): Build apps and services that speak naturally. Choose from a broad portfolio of [languages, voices, and variants](#). Bring your scenarios to life with highly expressive and human-like neural voices.
- [Custom voice](#): Create custom, one-of-a-kind voices for text to speech. You supply audio files and create matching transcriptions in Speech Studio, and then use the custom voices in your applications. To create and use custom voices via endpoints, see [Create and use your voice model](#).
- [Audio Content Creation](#): A no-code approach for text to speech synthesis. You can use the output audio as-is, or as a starting point for further customization. You can build highly natural audio content for various scenarios, such as audiobooks, news broadcasts, video narrations, and chat bots. For more information, see the [Audio Content Creation](#) documentation.
- [Custom Keyword](#): A custom keyword is a word or short phrase that you can use to voice-activate a product. You create a custom keyword in Speech Studio, and then generate a binary file to [use with the Speech SDK](#) in your applications.

Next steps

- [Explore Speech Studio](#)

Speech to text documentation

Speech to text from the Speech service, also known as speech recognition, enables real-time and batch transcription of audio streams into text. With additional reference text input, it also enables real-time pronunciation assessment and gives speakers feedback on the accuracy and fluency of spoken audio.

About speech to text

OVERVIEW

[What is real-time speech to text?](#)

[What is batch speech to text?](#)

[What is custom speech?](#)

[Use the Speech CLI for speech to text with no code](#)

QUICKSTART

[Get started with speech to text](#)

[Try real-time diarization](#)

Develop with speech to text

HOW-TO GUIDE

[Use the fast transcription API](#)

[Create a custom speech project](#)

[Train a model for custom speech](#)

[Use compressed audio input formats](#)

CONCEPT

[Whisper model from OpenAI](#)

[Improve accuracy with custom speech](#)

[Display text formatting](#)

Reference



[Language support](#)

[Speech to text FAQ](#)

[Speech to text pricing ↗](#)

Help and feedback



[Support and help options](#)

What is speech to text?

Azure AI Speech service offers advanced speech to text capabilities. This feature supports both real-time and batch transcription, providing versatile solutions for converting audio streams into text.

Core Features

The speech to text service offers the following core features:

- [Real-time transcription](#): Instant transcription with intermediate results for live audio inputs.
- [Fast transcription](#): Fastest synchronous output for situations with predictable latency.
- [Batch transcription](#): Efficient processing for large volumes of prerecorded audio.
- [Custom speech](#): Models with enhanced accuracy for specific domains and conditions.

Real-time transcription

Real-time speech to text transcribes audio as it's recognized from a microphone or file. It's ideal for applications requiring immediate transcription, such as:

- Real-time audio transcription for accessibility and record-keeping.
- Evaluating and providing feedback on pronunciation accuracy.
- Providing real-time transcription to assist customer service representatives.
- Transcribing spoken words into written text for documentation purposes.
- Enabling interactive voice response systems to transcribe user queries and commands.

Real-time speech to text can be accessed via the Speech SDK, Speech CLI, and REST API, allowing integration into various applications and workflows. Real-time speech to text is available via the [Speech SDK](#), the [Speech CLI](#), and [Speech to text REST API for short audio](#).

Fast transcription

Fast transcription API is used to transcribe audio files with returning results synchronously and faster than real-time audio. Use fast transcription in the scenarios that you need the transcript of an audio recording as quickly as possible with predictable latency, such as:

- Quick audio file transcription, captions, and edit
- Meeting notes
- Voicemail

To get started with fast transcription, see [use the fast transcription API](#).

Batch transcription

[Batch transcription](#) is designed for transcribing large amounts of audio stored in files. This method processes audio asynchronously and is suited for:

- Transcriptions, captions, or subtitles for large volumes of audio files
- Analyzing call center recorded calls to extract valuable insights.

Batch transcription is available via:

- [Speech to text REST API](#): Facilitates batch processing with the flexibility of RESTful calls. To get started, see [How to use batch transcription](#) and [Batch transcription samples](#).
- [Speech CLI](#): Supports both real-time and batch transcription, making it easy to manage transcription tasks. For Speech CLI help with batch transcriptions, run the following command:

Azure CLI

```
spx help batch transcription
```

Custom speech

With [custom speech](#), you can evaluate and improve the accuracy of speech recognition for your applications and products. A custom speech model can be used for [real-time speech to text](#), [speech translation](#), and [batch transcription](#).

 Tip

A [hosted deployment endpoint](#) isn't required to use custom speech with the [Batch transcription API](#). You can conserve resources if the [custom speech model](#) is only used for batch transcription. For more information, see [Speech service pricing](#).

Out of the box, speech recognition utilizes a Universal Language Model as a base model that is trained with Microsoft-owned data and reflects commonly used spoken language. The base model is pretrained with dialects and phonetics representing various common domains. When you make a speech recognition request, the most recent base model for each [supported language](#) is used by default. The base model works well in most speech recognition scenarios.

Custom speech allows you to tailor the speech recognition model to better suit your application's specific needs. This can be particularly useful for:

- **Improving recognition of domain-specific vocabulary:** Train the model with text data relevant to your field.
- **Enhancing accuracy for specific audio conditions:** Use audio data with reference transcriptions to refine the model.

For more information about custom speech, see the [custom speech overview](#) and the [speech to text REST API](#) documentation.

For details about customization options per language and locale, see the [language and voice support for the Speech service](#) documentation.

Usage Examples

Here are some practical examples of how you can utilize Azure AI speech to text:

 Expand table

Use case	Scenario	Solution
Live meeting transcriptions and captions	A virtual event platform needs to provide real-time captions for webinars.	Integrate real-time speech to text using the Speech SDK to transcribe spoken content into captions displayed live during the event.
Customer service enhancement	A call center wants to assist agents by providing real-time transcriptions of customer calls.	Use real-time speech to text via the Speech CLI to transcribe calls, enabling agents to better understand and respond to customer queries.
Video subtitling	A video-hosting platform wants to quickly generate a set of subtitles for a video.	Use fast transcription to quickly get a set of subtitles for the entire video.
Educational tools	An e-learning platform aims to provide transcriptions for video lectures.	Apply batch transcription through the speech to text REST API to process prerecorded lecture videos, generating text transcripts for students.
Healthcare documentation	A healthcare provider needs to document patient consultations.	Use real-time speech to text for dictation, allowing healthcare professionals to speak their notes and have them transcribed instantly. Use a custom model to enhance recognition of specific medical terms.
Media and entertainment	A media company wants to create subtitles for a large	Use batch transcription to process the video files in bulk, generating accurate subtitles for each

Use case	Scenario	Solution
	archive of videos.	video.
Market research	A market research firm needs to analyze customer feedback from audio recordings.	Employ batch transcription to convert audio feedback into text, enabling easier analysis and insights extraction.

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who are affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)
- [Integration and responsible use](#)
- [Data, privacy, and security](#)

Related content

- [Get started with speech to text](#)
- [Create a batch transcription](#)
- For detailed pricing information, visit the [Speech service pricing](#) page.

Last updated on 10/24/2025

Quickstart: Recognize and convert speech to text

08/05/2025

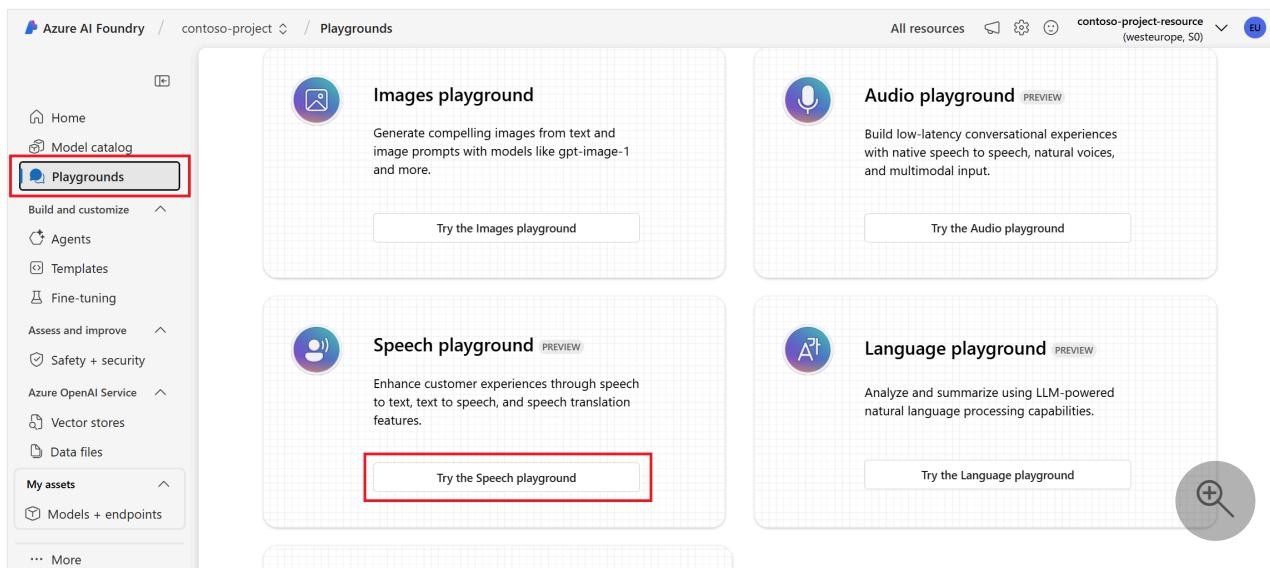
In this quickstart, you try real-time speech to text in [Azure AI Foundry](#).

Prerequisites

- An Azure subscription.
- An AI Foundry project. If you need to create a project, see [Create an Azure AI Foundry project](#).

Try real-time speech to text

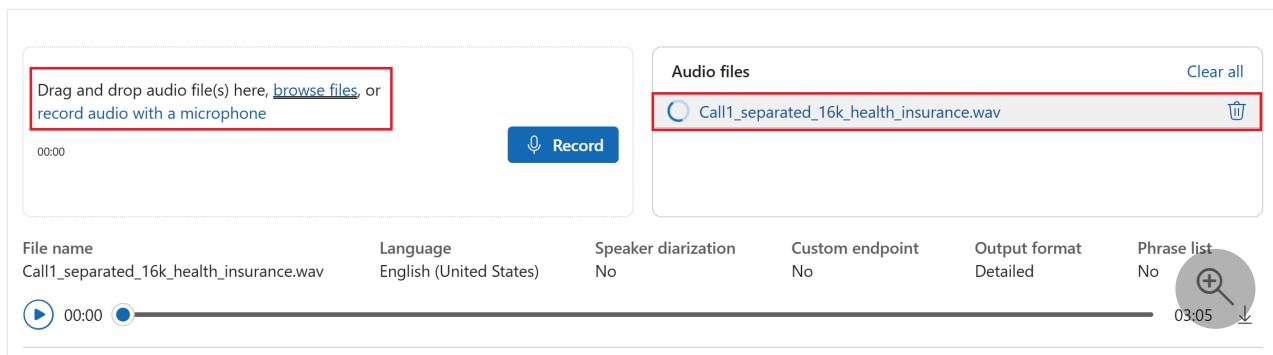
1. Go to your Azure AI Foundry project.
2. Select **Playgrounds** from the left pane and then select a playground to use. In this example, select **Try the Speech playground**.



3. Select **Real-time transcription**.
4. Select **Show advanced options** to configure speech to text options such as:
 - **Language identification:** Used to identify languages spoken in audio when compared against a list of supported languages. For more information about language identification options such as at-start and continuous recognition, see [Language identification](#).

- **Speaker diarization:** Used to identify and separate speakers in audio. Diarization distinguishes between the different speakers who participate in the conversation. The Speech service provides information about which speaker was speaking a particular part of transcribed speech. For more information about speaker diarization, see the [real-time speech to text with speaker diarization quickstart](#).
- **Custom endpoint:** Use a deployed model from custom speech to improve recognition accuracy. To use Microsoft's baseline model, leave this set to None. For more information about custom speech, see [Custom Speech](#).
- **Output format:** Choose between simple and detailed output formats. Simple output includes display format and timestamps. Detailed output includes more formats (such as display, lexical, ITN, and masked ITN), timestamps, and N-best lists.
- **Phrase list:** Improve transcription accuracy by providing a list of known phrases, such as names of people or specific locations. Use commas or semicolons to separate each value in the phrase list. For more information about phrase lists, see [Phrase lists](#).

5. Select an audio file to upload, or record audio in real-time. In this example, we use the `Call1_separated_16k_health_insurance.wav` file that's available in the [Speech SDK repository on GitHub](#). You can download the file or use your own audio file.



6. You can view the real-time transcription at the bottom of the page.

The screenshot shows the Azure AI Foundry interface with the project 'contoso-proj' selected. In the left sidebar, under 'Build and customize', 'Agents' is previewed. The main area displays the 'Speech Playground' with the 'Real-time transcription' tab selected. The configuration pane includes options for 'Language to transcribe' (English (United States)), 'Language identification' (set to 'Continuous'), 'Speaker diarization' (off), and various output settings like 'Custom endpoint', 'Output format' (Detailed), and 'Phrase list' (No). The transcript pane shows a recorded audio file named 'Call1_separated_16k_health_insurance.wav' with a duration of 00:03:05. The transcript text is displayed in JSON format:

```
[{"Text": "Hello, thank you for calling Contoso. Who am I speaking with today? Hi, my name is Mary Rondo. I'm trying to enroll myself with Contoso. Hi, Mary. Uh, are you calling because you need health insurance? Yes, Yeah, I'm calling to sign up for insurance. Great. Uh, if you can answer a few questions, we can get you signed up in the jiffy. OK. So what's your full name? So Mary Beth Rondo, last name is R like Romeo, O like ocean, N like Nancy, DD like Dog, and O like Ocean again. Rondo. Got it. And what's the best callback number in case we get disconnected? I only have a cell phone, so I can give you that. Yeah, that'll be fine. Sure. So it's 234554 and then 9312. Got it. So to confirm, it's 234-554-9312. Yep, that's right. Excellent. Uh, let's get some additional information from your application. Do you have a job? Uh, yes, I am self-employed. OK, so then you have a Social Security number as well? Yes, I do. OK. And what is your Social Security number, please? Sure. So it's 412256789. Sorry, what was that, A25 or A225 you cut out for a bit? Uh, it's 22, so 412, then another two, then 5. Alright, thank you so much. Umm, and could I have your e-mail address please? Yeah, it's Mary.rondo@gmail.com. So myfirstlastname@gmail.com. No periods, no dashes. Great. That is the last question. So let me take your information and I'll be able to get you signed up right away. Thank you for calling Contoso and I'll be able to get you signed up immediately. One of our agents will call you back in about 24 hours or so to confirm your application. That sounds great. Thank you. Absolutely. If you need anything else, please give us a call at 1-800-555-5564, ext 123. Uh, thank you very much for calling Contoso. Uh, actually, uh, so I have one more question. Uh, yes, of course. Umm, I'm curious, will I be getting a physical card as proof of coverage? So the default is a digital membership card. Uh, but we can send you a physical card if you prefer. Yes. Could you please mail it to me when it's ready? I'd like to have it shipped to Or are you from my address? Yeah. So it's 2660 Unit A on Maple Ave. SE, Lansing and then zip code is 48823. Absolutely. I've made a note on your file. Awesome. Thanks so much. You're very welcome. Thank you for calling Contoso and have a great day."}]
```

7. You can select the **JSON** tab to see the JSON output of the transcription. Properties include **offset**, **duration**, **RecognitionStatus**, **Display**, **Lexical**, **ITN**, and more.

The screenshot shows the same Azure AI Foundry interface, but the 'Text' tab is now highlighted with a red box. The JSON output is displayed in a code editor-like view:

```
[{"Text": "Hello, thank you for calling Contoso. Who am I speaking with today? Hi, my name is Mary Rondo. I'm trying to enroll myself with Contoso. Hi, Mary. Uh, are you calling because you need health insurance? Yes, Yeah, I'm calling to sign up for insurance. Great. Uh, if you can answer a few questions, we can get you signed up in the jiffy. OK. So what's your full name? So Mary Beth Rondo, last name is R like Romeo, O like ocean, N like Nancy, DD like Dog, and O like Ocean again. Rondo. Got it. And what's the best callback number in case we get disconnected? I only have a cell phone, so I can give you that. Yeah, that'll be fine. Sure. So it's 234554 and then 9312. Got it. So to confirm, it's 234-554-9312. Yep, that's right. Excellent. Uh, let's get some additional information from your application. Do you have a job? Uh, yes, I am self-employed. OK, so then you have a Social Security number as well? Yes, I do. OK. And what is your Social Security number, please? Sure. So it's 412256789. Sorry, what was that, A25 or A225 you cut out for a bit? Uh, it's 22, so 412, then another two, then 5. Alright, thank you so much. Umm, and could I have your e-mail address please? Yeah, it's Mary.rondo@gmail.com. So myfirstlastname@gmail.com. No periods, no dashes. Great. That is the last question. So let me take your information and I'll be able to get you signed up right away. Thank you for calling Contoso and I'll be able to get you signed up immediately. One of our agents will call you back in about 24 hours or so to confirm your application. That sounds great. Thank you. Absolutely. If you need anything else, please give us a call at 1-800-555-5564, ext 123. Uh, thank you very much for calling Contoso. Uh, actually, uh, so I have one more question. Uh, yes, of course. Umm, I'm curious, will I be getting a physical card as proof of coverage? So the default is a digital membership card. Uh, but we can send you a physical card if you prefer. Yes. Could you please mail it to me when it's ready? I'd like to have it shipped to Or are you from my address? Yeah. So it's 2660 Unit A on Maple Ave. SE, Lansing and then zip code is 48823. Absolutely. I've made a note on your file. Awesome. Thanks so much. You're very welcome. Thank you for calling Contoso and have a great day."}]
```

Next step

[Learn more about speech recognition](#)

How to recognize speech

07/12/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this how-to guide, you learn how to use Azure AI Speech for real-time speech to text conversion. Real-time speech recognition is ideal for applications requiring immediate transcription, such as dictation, call center assistance, and captioning for live meetings.

To learn how to set up the environment for a sample application, see [Quickstart: Recognize and convert speech to text](#).

Create a speech configuration instance

To call the Speech service by using the Speech SDK, you need to create a [SpeechConfig](#) instance. This class includes information about your Speech resource, like your key and associated region, endpoint, host, or authorization token.

1. Create an AI Foundry resource for Speech in the [Azure portal](#) ↗. Get the Speech resource key and endpoint.
2. Create a `SpeechConfig` instance by using the following code. Replace `YourSpeechKey` and `YourSpeechEndpoint` with your Speech resource key and endpoint.

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

class Program
{
    async static Task Main(string[] args)
    {
        var speechConfig = SpeechConfig.FromEndpoint(new
Uri("YourSpeechEndpoint"), "YourSpeechKey");
    }
}
```

You can initialize `SpeechConfig` in a few other ways:

- Use an endpoint, and pass in a Speech service endpoint. A key or authorization token is optional.
- Use a host, and pass in a host address. A key or authorization token is optional.

- Use an authorization token with the associated region/location.

 **Note**

Regardless of whether you're performing speech recognition, speech synthesis, translation, or intent recognition, you always create a configuration.

Recognize speech from a microphone

To recognize speech by using your device microphone, create an [AudioConfig](#) instance by using the `FromDefaultMicrophoneInput()` method. Then initialize the [SpeechRecognizer](#) object by passing `speechConfig` and `audioConfig`.

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

class Program
{
    async static Task FromMic(SpeechConfig speechConfig)
    {
        using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
        using var speechRecognizer = new SpeechRecognizer(speechConfig,
audioConfig);

        Console.WriteLine("Speak into your microphone.");
        var speechRecognitionResult = await speechRecognizer.RecognizeOnceAsync();
        Console.WriteLine($"RECOGNIZED: Text={speechRecognitionResult.Text}");
    }

    async static Task Main(string[] args)
    {
        var speechConfig = SpeechConfig.FromEndpoint(new
Uri("YourSpeechEndpoint"), "YourSpeechKey");
        await FromMic(speechConfig);
    }
}
```

If you want to use a *specific* audio input device, you need to specify the device ID in `AudioConfig`. To learn how to get the device ID, see [Select an audio input device with the Speech SDK](#).

Recognize speech from a file

If you want to recognize speech from an audio file instead of a microphone, you still need to create an `AudioConfig` instance. However, you don't call `FromDefaultMicrophoneInput()`. You call `FromWavFileInput()` and pass the file path:

```
C#  
  
using System;  
using System.IO;  
using System.Threading.Tasks;  
using Microsoft.CognitiveServices.Speech;  
using Microsoft.CognitiveServices.Speech.Audio;  
  
class Program  
{  
    async static Task FromFile(SpeechConfig speechConfig)  
    {  
        using var audioConfig = AudioConfig.FromWavFileInput("PathToFile.wav");  
        using var speechRecognizer = new SpeechRecognizer(speechConfig,  
audioConfig);  
  
        var speechRecognitionResult = await speechRecognizer.RecognizeOnceAsync();  
        Console.WriteLine($"RECOGNIZED: Text={speechRecognitionResult.Text}");  
    }  
  
    async static Task Main(string[] args)  
    {  
        var speechConfig = SpeechConfig.FromEndpoint(new  
Uri("YourSpeechEndpoint"), "YourSpeechKey");  
        await FromFile(speechConfig);  
    }  
}
```

Recognize speech from an in-memory stream

For many use cases, it's likely that your audio data comes from Azure Blob Storage, or it's otherwise already in memory as a `byte[]` instance or a similar raw data structure. The following example uses `PushAudioInputStream` to recognize speech, which is essentially an abstracted memory stream. The sample code does the following actions:

- Writes raw audio data to `PushAudioInputStream` by using the `Write()` function, which accepts a `byte[]` instance.
- Reads a `.wav` file by using `FileReader` for demonstration purposes. If you already have audio data in a `byte[]` instance, you can skip directly to writing the content to the input stream.

- The default format is 16-bit, 16-kHz mono pulse-code modulation (PCM) data. To customize the format, you can pass an `AudioStreamFormat` object to `CreatePushStream()` by using the static function `AudioStreamFormat.GetWaveFormatPCM(sampleRate, (byte)bitRate, (byte)channels)`.

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

class Program
{
    async static Task FromStream(SpeechConfig speechConfig)
    {
        var reader = new BinaryReader(File.OpenRead("PathToFile.wav"));
        using var audioConfigStream = AudioInputStream.CreatePushStream();
        using var audioConfig = AudioConfig.FromStreamInput(audioConfigStream);
        using var speechRecognizer = new SpeechRecognizer(speechConfig,
audioConfig);

        byte[] readBytes;
        do
        {
            readBytes = reader.ReadBytes(1024);
            audioConfigStream.Write(readBytes, readBytes.Length);
        } while (readBytes.Length > 0);

        var speechRecognitionResult = await speechRecognizer.RecognizeOnceAsync();
        Console.WriteLine($"RECOGNIZED: Text={speechRecognitionResult.Text}");
    }

    async static Task Main(string[] args)
    {
        var speechConfig = SpeechConfig.FromEndpoint(new
Uri("YourSpeechEndpoint"), "YourSpeechKey");
        await FromStream(speechConfig);
    }
}
```

Using a push stream as input assumes that the audio data is raw PCM and skips any headers. The API still works in certain cases if the header isn't skipped. For the best results, consider implementing logic to read off the headers so that `byte[]` begins at the *start of the audio data*.

Handle errors

The previous examples only get the recognized text from the `speechRecognitionResult.Text` property. To handle errors and other responses, you need to write some code to handle the result. The following code evaluates the `speechRecognitionResult.Reason` property and:

- Prints the recognition result: `ResultReason.RecognizedSpeech`.
- If there's no recognition match, it informs the user: `ResultReason.NoMatch`.
- If an error is encountered, it prints the error message: `ResultReason.Canceled`.

C#

```
switch (speechRecognitionResult.Reason)
{
    case ResultReason.RecognizedSpeech:
        Console.WriteLine($"RECOGNIZED: Text={speechRecognitionResult.Text}");
        break;
    case ResultReason.NoMatch:
        Console.WriteLine($"NOMATCH: Speech could not be recognized.");
        break;
    case ResultReason.Canceled:
        var cancellation =
CancellationDetails.FromResult(speechRecognitionResult);
        Console.WriteLine($"CANCELED: Reason={cancellation.Reason}");

        if (cancellation.Reason == CancellationReason.Error)
        {
            Console.WriteLine($"CANCELED: ErrorCode={cancellation.ErrorCode}");
            Console.WriteLine($"CANCELED: ErrorDetails=
{cancellation.ErrorDetails}");
            Console.WriteLine($"CANCELED: Did you set the speech resource key and
endpoint values?");
        }
        break;
}
```

Use continuous recognition

The previous examples use single-shot recognition, which recognizes a single utterance. The end of a single utterance is determined by listening for silence at the end or until a maximum of 15 seconds of audio is processed.

In contrast, you use continuous recognition when you want to control when to stop recognizing. It requires you to subscribe to the `Recognizing`, `Recognized`, and `Canceled` events to get the recognition results. To stop recognition, you must call `StopContinuousRecognitionAsync`. Here's an example of how continuous recognition is performed on an audio input file.

Start by defining the input and initializing `SpeechRecognizer`:

C#

```
using var audioConfig = AudioConfig.FromWavFileInput("YourAudioFile.wav");
using var speechRecognizer = new SpeechRecognizer(speechConfig, audioConfig);
```

Then create a `TaskCompletionSource<int>` instance to manage the state of speech recognition:

C#

```
var stopRecognition = new TaskCompletionSource<int>();
```

Next, subscribe to the events that `SpeechRecognizer` sends:

- **Recognizing**: Signal for events that contain intermediate recognition results.
- **Recognized**: Signal for events that contain final recognition results, which indicate a successful recognition attempt.
- **SessionStopped**: Signal for events that indicate the end of a recognition session (operation).
- **Canceled**: Signal for events that contain canceled recognition results. These results indicate a recognition attempt that was canceled as a result of a direct cancelation request. Alternatively, they indicate a transport or protocol failure.

C#

```
speechRecognizer.Recognizing += (s, e) =>
{
    Console.WriteLine($"RECOGNIZING: Text={e.Result.Text}");
};

speechRecognizer.Recognized += (s, e) =>
{
    if (e.Result.Reason == ResultReason.RecognizedSpeech)
    {
        Console.WriteLine($"RECOGNIZED: Text={e.Result.Text}");
    }
    else if (e.Result.Reason == ResultReason.NoMatch)
    {
        Console.WriteLine($"NOMATCH: Speech could not be recognized.");
    }
};

speechRecognizer.Canceled += (s, e) =>
{
    Console.WriteLine($"CANCELED: Reason={e.Reason}");

    if (e.Reason == CancellationReason.Error)
    {
        Console.WriteLine($"CANCELED: ErrorCode={e.ErrorCode}");
    }
};
```

```
        Console.WriteLine($"CANCELED: ErrorDetails={e.ErrorDetails}");
        Console.WriteLine($"CANCELED: Did you set the speech resource key and
endpoint values?");
    }

    stopRecognition.TrySetResult(0);
};

speechRecognizer.SessionStopped += (s, e) =>
{
    Console.WriteLine("\n      Session stopped event.");
    stopRecognition.TrySetResult(0);
};
```

With everything set up, call `StartContinuousRecognitionAsync` to start recognizing:

C#

```
await speechRecognizer.StartContinuousRecognitionAsync();

// Waits for completion. Use Task.WaitAny to keep the task rooted.
Task.WaitAny(new[] { stopRecognition.Task });

// Make the following call at some point to stop recognition:
// await speechRecognizer.StopContinuousRecognitionAsync();
```

Change the source language

A common task for speech recognition is specifying the input (or source) language. The following example shows how to change the input language to Italian. In your code, find your `SpeechConfig` instance and add this line directly below it:

C#

```
speechConfig.SpeechRecognitionLanguage = "it-IT";
```

The `SpeechRecognitionLanguage` property expects a language-locale format string. For a list of supported locales, see [Language and voice support for the Speech service](#).

Language identification

You can use language identification with speech to text recognition when you need to identify the language in an audio source and then transcribe it to text.

For a complete code sample, see [Language identification](#).

Use a custom endpoint

With [custom speech](#), you can upload your own data, test and train a custom model, compare accuracy between models, and deploy a model to a custom endpoint. The following example shows how to set a custom endpoint.

C#

```
var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
    "YourSpeechKey");
speechConfig.EndpointId = "YourEndpointId";
var speechRecognizer = new SpeechRecognizer(speechConfig);
```

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 Host URLs in [Install and run Speech containers with Docker](#).

Change how silence is handled

If a user speaks faster or slower than usual, the default behaviors for nonspeech silence in input audio might not result in what you expect. Common problems with silence handling include:

- Fast-speech that chains many sentences together into a single recognition result, instead of breaking sentences into individual results.
- Slow speech that separates parts of a single sentence into multiple results.
- A single-shot recognition that ends too quickly while waiting for speech to begin.

These problems can be addressed by setting one of two *timeout properties* on the `SpeechConfig` instance used to create a `SpeechRecognizer`:

- **Segmentation silence timeout** adjusts how much nonspeech audio is allowed within a phrase that's currently being spoken before that phrase is considered "done."
 - *Higher* values generally make results longer and allow longer pauses from the speaker within a phrase but make results take longer to arrive. They can also combine separate phrases into a single result when set too high.

- Lower values generally make results shorter and ensure more prompt and frequent breaks between phrases, but can also cause single phrases to separate into multiple results when set too low.
- This timeout can be set to integer values between 100 and 5000, in milliseconds, with 500 a typical default.
- **Initial silence timeout** adjusts how much nonspeech audio is allowed *before* a phrase before the recognition attempt ends in a "no match" result.
 - Higher values give speakers more time to react and start speaking, but can also result in slow responsiveness when nothing is spoken.
 - Lower values ensure a prompt "no match" for faster user experience and more controlled audio handling, but might cut a speaker off too quickly when set too low.
 - Because continuous recognition generates many results, this value determines how often "no match" results arrive but doesn't otherwise affect the content of recognition results.
 - This timeout can be set to any non-negative integer value, in milliseconds, or set to 0 to disable it entirely. 5000 is a typical default for single-shot recognition while 15000 is a typical default for continuous recognition.

Since there are tradeoffs when modifying these timeouts, you should only change the settings when you have a problem related to silence handling. Default values optimally handle most spoken audio and only uncommon scenarios should encounter problems.

Example: Users speaking a serial number like "ABC-123-4567" might pause between character groups long enough for the serial number to be broken into multiple results. In this case, try a higher value like 2000 milliseconds for the segmentation silence timeout:

```
C#
```

```
speechConfig SetProperty(PropertyId.Speech_SegmentationSilenceTimeoutMs, "2000");
```

Example: A recorded presenter's speech might be fast enough that several sentences in a row get combined, with large recognition results only arriving once or twice per minute. In this case, set the segmentation silence timeout to a lower value like 300 ms:

```
C#
```

```
speechConfig SetProperty(PropertyId.Speech_SegmentationSilenceTimeoutMs, "300");
```

Example: A single-shot recognition asking a speaker to find and read a serial number ends too quickly while the number is being found. In this case, try a longer initial silence timeout like 10,000 ms:

C#

```
speechConfig SetProperty(PropertyId.SpeechServiceConnection_InitialSilenceTimeoutMs, "10000");
```

Semantic segmentation

Semantic segmentation is a speech recognition segmentation strategy that's designed to mitigate issues associated with [silence-based segmentation](#):

- Under-segmentation: When users speak for a long time without pauses, they can see a long sequence of text without breaks ("wall of text"), which severely degrades their readability experience.
- Over-segmentation: When a user pauses for a short time, the silence detection mechanism can segment incorrectly.

Instead of only relying on silence timeouts, semantic segmentation mostly segments and returns final results when it detects sentence-ending punctuation (such as '.' or '?'). This improves the user experience with higher-quality, semantically complete segments and prevents long intermediate results.

To use semantic segmentation, you need to set the following property on the `SpeechConfig` instance used to create a `SpeechRecognizer`:

C#

```
speechConfig SetProperty(PropertyId.Speech_SegmentationStrategy, "Semantic");
```

Some of the limitations of semantic segmentation are as follows:

- You need the Speech SDK version 1.41 or later to use semantic segmentation.
- Semantic segmentation is only intended for use in [continuous recognition](#). This includes scenarios such as dictation and captioning. It shouldn't be used in the single recognition mode or interactive scenarios.
- Semantic segmentation isn't available for all languages and locales.
- Semantic segmentation doesn't yet support confidence scores and NBest lists. As such, we don't recommend semantic segmentation if you're using confidence scores or NBest lists.

Related content

- [Try the speech to text quickstart](#)

- Improve recognition accuracy with custom speech
- Use batch transcription

Get speech recognition results

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this how-to guide, you learn about how you can use speech recognition results.

Speech synchronization

You might want to synchronize transcriptions with an audio track, whether it's done in real-time or with a prerecording.

The Speech service returns the offset and duration of the recognized speech.

- **Offset:** The offset into the audio stream being recognized, expressed as duration. Offset is measured in ticks, starting from 0 (zero) tick, associated with the first audio byte processed by the SDK. For example, the offset begins when you start recognition, since that's when the SDK starts processing the audio stream. One tick represents one hundred nanoseconds or one ten-millionth of a second.
- **Duration:** Duration of the utterance that is being recognized. The duration in ticks doesn't include trailing or leading silence.

The end of a single utterance is determined by listening for silence at the end. You won't get the final recognition result until an utterance has completed. Recognizing events will provide intermediate results that are subject to change while an audio stream is being processed. Recognized events will provide the final transcribed text once processing of an utterance is completed.

Recognizing offset and duration

With the `Recognizing` event, you can get the offset and duration of the speech being recognized. Offset and duration per word are not available while recognition is in progress. Each `Recognizing` event comes with a textual estimate of the speech recognized so far.

This code snippet shows how to get the offset and duration from a `Recognizing` event.

C#

```
speechRecognizer.Recognizing += (object sender, SpeechRecognitionEventArgs e) =>
{
    if (e.Result.Reason == ResultReason.RecognizingSpeech)
    {
        Console.WriteLine(String.Format ("RECOGNIZING: {0}", e.Result.Text));
```

```
        Console.WriteLine(String.Format ("Offset in Ticks: {0}",  
e.Result.OffsetInTicks));  
        Console.WriteLine(String.Format ("Duration in Ticks: {0}",  
e.Result.Duration.Ticks));  
    }  
};
```

Recognized offset and duration

Once an utterance has been recognized, you can get the offset and duration of the recognized speech. With the `Recognized` event, you can also get the offset and duration per word. To request the offset and duration per word, first you must set the corresponding `SpeechConfig` property as shown here:

C#

```
speechConfig.RequestWordLevelTimestamps();
```

This code snippet shows how to get the offset and duration from a `Recognized` event.

C#

```
speechRecognizer.Recognized += (object sender, SpeechRecognitionEventArgs e) =>  
{  
    if (ResultReason.RecognizedSpeech == e.Result.Reason &&  
e.Result.Text.Length > 0)  
    {  
        Console.WriteLine($"RECOGNIZED: Text={e.Result.Text}");  
        Console.WriteLine(String.Format ("Offset in Ticks: {0}",  
e.Result.OffsetInTicks));  
        Console.WriteLine(String.Format ("Duration in Ticks: {0}",  
e.Result.Duration.Ticks));  
  
        var detailedResults = e.Result.Best();  
        if(detailedResults != null && detailedResults.Any())  
        {  
            // The first item in detailedResults corresponds to the recognized  
text.  
            // This is not necessarily the item with the highest confidence  
number.  
            var bestResults = detailedResults?.ToList()[0];  
            Console.WriteLine(String.Format ("\tConfidence: {0}\n\tText:  
{1}\n\tLexicalForm: {2}\n\tNormalizedForm: {3}\n\tMaskedNormalizedForm: {4}",  
bestResults.Confidence, bestResults.Text,  
bestResults.LexicalForm, bestResults.NormalizedForm,  
bestResults.MaskedNormalizedForm));  
            // You must set speechConfig.RequestWordLevelTimestamps() to get  
word-level timestamps.  
            Console.WriteLine($" \tWord-level timing:");  
        }  
    }  
};
```

```

        Console.WriteLine($"\\t\\tWord | Offset | Duration");
        Console.WriteLine($"\\t\\t----- | ----- | ----- ");

        foreach (var word in bestResults.Words)
        {
            Console.WriteLine($"\\t\\t{word.Word} | {word.Offset} |
{word.Duration}");
        }
    }
};

```

Example offset and duration

The following table shows potential offset and duration in ticks when a speaker says "Welcome to Applied Mathematics course 201." In this example, the offset doesn't change throughout the `Recognizing` and `Recognized` events. However, don't rely on the offset to remain the same between the `Recognizing` and `Recognized` events, since the final result could be different.

[] Expand table

Event	Text	Offset (in ticks)	Duration (in ticks)
RECOGNIZING	welcome	17000000	5000000
RECOGNIZING	welcome to	17000000	6400000
RECOGNIZING	welcome to applied math	17000000	13600000
RECOGNIZING	welcome to applied mathematics	17000000	17200000
RECOGNIZING	welcome to applied mathematics course	17000000	23700000
RECOGNIZING	welcome to applied mathematics course 2	17000000	26700000
RECOGNIZING	welcome to applied mathematics course 201	17000000	33400000
RECOGNIZED	Welcome to applied Mathematics course 201.	17000000	34500000

The total duration of the first utterance was 3.45 seconds. It was recognized at 1.7 to 5.15 seconds offset from the start of the audio stream being recognized (00:00:01.700 --> 00:00:05.150).

If the speaker continues then to say "Let's get started," a new offset is calculated from the start of the audio stream being recognized, to the start of the new utterance. The following table shows potential offset and duration for an utterance that started two seconds after the previous utterance ended.

Event	Text	Offset (in ticks)	Duration (in ticks)
RECOGNIZING	OK	71500000	3100000
RECOGNIZING	OK now	71500000	10300000
RECOGNIZING	OK now let's	71500000	14700000
RECOGNIZING	OK now let's get started	71500000	18500000
RECOGNIZED	OK, now let's get started.	71500000	20600000

The total duration of the second utterance was 2.06 seconds. It was recognized at 7.15 to 9.21 seconds offset from the start of the audio stream being recognized (00:00:07.150 --> 00:00:09.210).

Next steps

- [Try the speech to text quickstart](#)
- [Improve recognition accuracy with custom speech](#)
- [Use batch transcription](#)

Quickstart: Create real-time diarization

07/17/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this quickstart, you run an application for speech to text transcription with real-time diarization. Diarization distinguishes between the different speakers who participate in the conversation. The Speech service provides information about which speaker was speaking a particular part of transcribed speech.

The speaker information is included in the result in the speaker ID field. The speaker ID is a generic identifier assigned to each conversation participant by the service during the recognition as different speakers are being identified from the provided audio content.

💡 Tip

You can try real-time speech to text in [Speech Studio](#) ↗ without signing up or writing any code. However, the Speech Studio doesn't yet support diarization.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#) ↗ .
- ✓ [Create an AI Services resource for Speech](#) ↗ in the Azure portal.
- ✓ Get the Speech resource key and endpoint. After your Speech resource is deployed, select [Go to resource](#) to view and manage keys.

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

You need to authenticate your application to access Azure AI services. This article shows you how to use environment variables to store your credentials. You can then access the environment variables from your code to authenticate your application. For production, use a more secure way to store and access your credentials.

Important

We recommend Microsoft Entra ID authentication with [managed identities for Azure resources](#) to avoid storing credentials with your applications that run in the cloud.

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If using API keys, store them securely in Azure Key Vault, rotate the keys regularly, and restrict access to Azure Key Vault using role based access control and network access restrictions. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

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

- To set the `SPEECH_KEY` environment variable, replace *your-key* with one of the keys for your resource.
- To set the `ENDPOINT` environment variable, replace *your-endpoint* with one of the endpoints for your resource.

Windows

Console

```
setx SPEECH_KEY your-key  
setx ENDPOINT your-endpoint
```

Note

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

After you add the environment variables, you might need to restart any programs that need to read the environment variables, including the console window. For example, if you're using Visual Studio as your editor, restart Visual Studio before you run the example.

Implement diarization from file with conversation transcription

Follow these steps to create a console application and install the Speech SDK.

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
```

This command creates the *Program.cs* file in your project directory.

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

```
.NET CLI
```

```
dotnet add package Microsoft.CognitiveServices.Speech
```

3. Replace the contents of *Program.cs* with the following code.

```
C#
```

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Microsoft.CognitiveServices.Speech.Transcription;

class Program
{
    // This example requires environment variables named "SPEECH_KEY" and
    "ENDPOINT"
    static string speechKey =
        Environment.GetEnvironmentVariable("SPEECH_KEY");
    static string endpoint = Environment.GetEnvironmentVariable("ENDPOINT");

    async static Task Main(string[] args)
    {
        var filepath = "katiesteve.wav";
        var speechConfig = SpeechConfig.FromEndpoint(speechKey, endpoint);
        speechConfig.SpeechRecognitionLanguage = "en-US";

        speechConfig SetProperty(PropertyId.SpeechServiceResponse_DiarizeIntermediate
        Results, "true");

        var stopRecognition = new TaskCompletionSource<int>
            (TaskCreationOptions.RunContinuationsAsynchronously);
```

```

        // Create an audio stream from a wav file or from the default
microphone
        using (var audioConfig = AudioConfig.FromWavFileInput(filepath))
        {
            // Create a conversation transcriber using audio stream input
            using (var conversationTranscriber = new
ConversationTranscriber(speechConfig, audioConfig))
            {
                conversationTranscriber.Transcribing += (s, e) =>
                {
                    Console.WriteLine($"TRANSCRIBING: Text={e.Result.Text}
Speaker ID={e.Result.SpeakerId}");
                };

                conversationTranscriber.Transcribed += (s, e) =>
                {
                    if (e.Result.Reason == ResultReason.RecognizedSpeech)
                    {
                        Console.WriteLine();
                        Console.WriteLine($"TRANSCRIBED: Text={e.Result.Text}
Speaker ID={e.Result.SpeakerId}");
                        Console.WriteLine();
                    }
                    else if (e.Result.Reason == ResultReason.NoMatch)
                    {
                        Console.WriteLine($"NOMATCH: Speech could not be
transcribed.");
                    }
                };
            }

            conversationTranscriber.Canceled += (s, e) =>
            {
                Console.WriteLine($"CANCELED: Reason={e.Reason}");

                if (e.Reason == CancellationReason.Error)
                {
                    Console.WriteLine($"CANCELED: ErrorCode=
{e.ErrorCode}");
                    Console.WriteLine($"CANCELED: ErrorDetails=
{e.ErrorDetails}");
                    Console.WriteLine($"CANCELED: Did you set the speech
resource key and endpoint values?");
                    stopRecognition.TrySetResult(0);
                }

                stopRecognition.TrySetResult(0);
            };
        }

        conversationTranscriber.SessionStopped += (s, e) =>
        {
            Console.WriteLine("\n    Session stopped event.");
            stopRecognition.TrySetResult(0);
        };

        await conversationTranscriber.StartTranscribingAsync();
    }
}

```

```
// Waits for completion. Use Task.WaitAny to keep the task rooted.  
rooted.  
    Task.WaitAny(new[] { stopRecognition.Task });  
  
    await conversationTranscriber.StopTranscribingAsync();  
}  
}  
}  
}
```

4. Get the [sample audio file](#) or use your own `.wav` file. Replace `katiesteve.wav` with the path and name of your `.wav` file.

The application recognizes speech from multiple participants in the conversation. Your audio file should contain multiple speakers.

5. 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` if you don't specify a language. For details about how to identify one of multiple languages that might be spoken, see [language identification](#).
6. Run your console application to start conversation transcription:

.NET CLI

```
dotnet run
```

Important

Make sure that you set the `SPEECH_KEY` and `ENDPOINT` [environment variables](#). If you don't set these variables, the sample fails with an error message.

The transcribed conversation should be output as text:

Output

```
TRANSCRIBING: Text=good morning steve Speaker ID=Unknown  
TRANSCRIBING: Text=good morning steve how are Speaker ID=Guest-1  
TRANSCRIBING: Text=good morning steve how are you doing today Speaker ID=Guest-1  
  
TRANSCRIBED: Text=Good morning, Steve. How are you doing today? Speaker ID=Guest-1  
  
TRANSCRIBING: Text=good morning katie Speaker ID=Unknown  
TRANSCRIBING: Text=good morning katie i hope Speaker ID=Guest-2  
TRANSCRIBING: Text=good morning katie i hope you're having a great Speaker
```

ID=Guest-2
TRANSCRIBING: Text=good morning katie i hope you're having a great start to Speaker ID=Guest-2
TRANSCRIBING: Text=good morning katie i hope you're having a great start to your day Speaker ID=Guest-2

TRANSCRIBED: Text=Good morning, Katie. I hope you're having a great start to your day. Speaker ID=Guest-2

TRANSCRIBING: Text=have you tried Speaker ID=Unknown
TRANSCRIBING: Text=have you tried the latest Speaker ID=Unknown
TRANSCRIBING: Text=have you tried the latest real time Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can tell you Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can tell you who said Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can tell you who said what Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can tell you who said what in Speaker ID=Guest-1
TRANSCRIBING: Text=have you tried the latest real time diarization in microsoft speech service which can tell you who said what in real time Speaker ID=Guest-1

TRANSCRIBED: Text=Have you tried the latest real time diarization in Microsoft Speech Service which can tell you who said what in real time? Speaker ID=Guest-1

TRANSCRIBING: Text=not yet Speaker ID=Unknown
TRANSCRIBING: Text=not yet i Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch trans Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch transcription with Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch transcription with diarization Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch transcription with diarization function Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch transcription with diarization functionality Speaker ID=Guest-2
TRANSCRIBING: Text=not yet i've been using the batch transcription with

ID=Guest-2

TRANSCRIBED: Text=Not yet. I've been using the batch transcription with diarization functionality, but it produces diarization results after the whole audio is processed. Is the new feature able to diarize in real time? Speaker ID=Guest-2

TRANSCRIBING: Text=absolutely Speaker ID=Unknown

TRANSCRIBING: Text=absolutely i Speaker ID=Unknown

TRANSCRIBING: Text=absolutely i recom Speaker ID=Guest-1

TRANSCRIBING: Text=absolutely i recommend Speaker ID=Guest-1

TRANSCRIBING: Text=absolutely i recommend you give it a try Speaker ID=Guest-1

TRANSCRIBED: Text=Absolutely, I recommend you give it a try. Speaker ID=Guest-1

TRANSCRIBING: Text=that's exc Speaker ID=Unknown

TRANSCRIBING: Text=that's exciting Speaker ID=Unknown

TRANSCRIBING: Text=that's exciting let me try Speaker ID=Guest-2

TRANSCRIBING: Text=that's exciting let me try it right now Speaker ID=Guest-2

TRANSCRIBED: Text=That's exciting. Let me try it right now. Speaker ID=Guest-2

Speakers are identified as Guest-1, Guest-2, and so on, depending on the number of speakers in the conversation.

Note

You might see `Speaker ID=Unknown` in some of the early intermediate results when the speaker isn't yet identified. Without intermediate diarization results (if you don't set the `PropertyId.SpeechServiceResponse_DiarizeIntermediateResults` property to "true"), the speaker ID is always "Unknown."

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Speech resource you created.

Next step

[Learn more about speech recognition](#)

Use the fast transcription API with Azure AI Speech

Fast transcription API is used to transcribe audio files with returning results synchronously and faster than real-time. Use fast transcription in the scenarios that you need the transcript of an audio recording as quickly as possible with predictable latency, such as:

- Quick audio or video transcription, subtitles, and edit.
- Meeting notes
- Voicemail

Unlike the batch transcription API, fast transcription API only produces transcriptions in the display (not lexical) form. The display form is a more human-readable form of the transcription that includes punctuation and capitalization.

💡 Tip

You can also use the latest LLM-powered speech transcription and speech translation with [LLM speech](#).

Prerequisites

- An Azure AI Speech resource in one of the regions where the fast transcription API is available. For the current list of supported regions, see the [Speech service regions table](#).
- An audio file (less than 2 hours long and less than 300 MB in size) in one of the formats and codecs supported by the batch transcription API: WAV, MP3, OPUS/OGG, FLAC, WMA, AAC, ALAW in WAV container, MULAW in WAV container, AMR, WebM, and SPEEX. For more information about supported audio formats, see [supported audio formats](#).

Upload audio

You can provide audio data to fast transcription in the following ways:

- Inline audio upload

```
--form 'audio=@"YourAudioFile"'
```

- Audio from a public URL

```
--form 'definition={"audioUrl": "https://crbn.us/hello.wav"}'"
```

In the sections below, inline audio upload is used as an example.

Use the fast transcription API

💡 Tip

Try out fast transcription in the [Azure AI Foundry portal](#). We learn how to use the fast transcription API (via [Transcriptions - Transcribe](#)) with the following scenarios:

- [Known locale specified](#): Transcribe an audio file with a specified locale. If you know the locale of the audio file, you can specify it to improve transcription accuracy and minimize the latency.
- [Language identification on](#): Transcribe an audio file with language identification on. If you're not sure about the locale of the audio file, you can turn on language identification to let the Speech service identify the locale (one locale per audio).
- [Multi-lingual transcription \(preview\)](#): Transcribe an audio file with the latest multi-lingual speech transcription model. If your audio contains multi-lingual contents that you want to transcribe continuously and accurately, you can use the latest multi-lingual speech transcription model without specifying the locale codes.
- [Diarization on](#): Transcribe an audio file with diarization on. Diarization distinguishes between different speakers in the conversation. The Speech service provides information about which speaker was speaking a particular part of the transcribed speech.
- [Multi-channel on](#): Transcribe an audio file that has one or two channels. Multi-channel transcriptions are useful for audio files with multiple channels, such as audio files with multiple speakers or audio files with background noise. By default, the fast transcription API merges all input channels into a single channel and then performs the transcription. If this isn't desirable, channels can be transcribed independently without merging.

Known locale specified

Make a multipart/form-data POST request to the `transcriptions` endpoint with the audio file and the request body properties.

The following example shows how to transcribe an audio file with a specified locale. If you know the locale of the audio file, you can specify it to improve transcription accuracy and minimize the latency.

- Replace `YourSpeechResourceKey` with your Speech resource key.
- Replace `YourServiceRegion` with your Speech resource region.
- Replace `YourAudioFile` with the path to your audio file.

Important

For the recommended keyless authentication with Microsoft Entra ID, replace `--header 'Ocp-Apim-Subscription-Key: YourSpeechResourceKey'` with `--header "Authorization: Bearer YourAccessToken"`. For more information about keyless authentication, see the [role-based access control](#) how-to guide.

Azure CLI

```
curl --location  
'https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions:transcribe?api-version=2025-10-15' \  
--header 'Content-Type: multipart/form-data' \  
--header 'Ocp-Apim-Subscription-Key: YourSpeechResourceKey' \  
--form 'audio=@"YourAudioFile"' \  
--form 'definition="{  
    \"locales\":[\"en-US\"]}"'
```

Construct the form definition according to the following instructions:

- Set the optional (but recommended) `locales` property that should match the expected locale of the audio data to transcribe. In this example, the locale is set to `en-US`. For more information about the supported locales, see [speech to text supported languages](#).

For more information about `locales` and other properties for the fast transcription API, see the [request configuration options](#) section later in this guide.

The response includes `durationMilliseconds`, `offsetMilliseconds`, and more. The `combinedPhrases` property contains the full transcriptions for all speakers.

JSON

```
{  
    "durationMilliseconds": 182439,  
    "combinedPhrases": [  
        {  
            "text": "Good afternoon. This is Sam. Thank you for calling Contoso.  
How can I help? Hi there. My name is Mary. I'm currently living in Los Angeles,  
but I'm planning to move to Las Vegas. I would like to apply for a loan. Okay. I  
see you're currently living in California. Let me make sure I understand you"}]
```

correctly. Uh You'd like to apply for a loan even though you'll be moving soon. Is that right? Yes, exactly. So I'm planning to relocate soon, but I would like to apply for the loan first so that I can purchase a new home once I move there. And are you planning to sell your current home? Yes, I will be listing it on the market soon and hopefully it'll sell quickly. That's why I'm applying for a loan now, so that I can purchase a new house in Nevada and close on it quickly as well once my current home sells. I see. Would you mind holding for a moment while I take your information down? Yeah, no problem. Thank you for your help. Mm-hmm. Just one moment. All right. Thank you for your patience, ma'am. May I have your first and last name, please? Yes, my name is Mary Smith. Thank you, Ms. Smith. May I have your current address, please? Yes. So my address is 123 Main Street in Los Angeles, California, and the zip code is 90923. Sorry, that was a 90 what? 90923. 90923 on Main Street. Got it. Thank you. May I have your phone number as well, please? Uh Yes, my phone number is 504-529-2351 and then yeah. 2351. Got it. And do you have an e-mail address we I can associate with this application? uh Yes, so my e-mail address is mary.a.sm78@gmail.com. Mary.a, was that a S-N as in November or M as in Mike? M as in Mike. Mike78, got it. Thank you. Ms. Smith, do you currently have any other loans? Uh Yes, so I currently have two other loans through Contoso. So my first one is my car loan and then my other is my student loan. They total about 1400 per month combined and my interest rate is 8%. I see. And you're currently paying those loans off monthly, is that right? Yes, of course I do. OK, thank you. Here's what I suggest we do. Let me place you on a brief hold again so that I can talk with one of our loan officers and get this started for you immediately. In the meantime, it would be great if you could take a few minutes and complete the remainder of the secure application online at www.contosoloans.com. Yeah, that sounds good. I can go ahead and get started. Thank you for your help. Thank you."

```
        },
    ],
    "phrases": [
        {
            "offsetMilliseconds": 960,
            "durationMilliseconds": 640,
            "text": "Good afternoon.",
            "words": [
                {
                    "text": "Good",
                    "offsetMilliseconds": 960,
                    "durationMilliseconds": 240
                },
                {
                    "text": "afternoon.",
                    "offsetMilliseconds": 1200,
                    "durationMilliseconds": 400
                }
            ],
            "locale": "en-US",
            "confidence": 0.93554276
        },
        {
            "offsetMilliseconds": 1600,
            "durationMilliseconds": 640,
            "text": "This is Sam.",
            "words": [

```

```
{
    "text": "This",
    "offsetMilliseconds": 1600,
    "durationMilliseconds": 240
},
{
    "text": "is",
    "offsetMilliseconds": 1840,
    "durationMilliseconds": 120
},
{
    "text": "Sam.",
    "offsetMilliseconds": 1960,
    "durationMilliseconds": 280
}
],
"locale": "en-US",
"confidence": 0.93554276
},
{
    "offsetMilliseconds": 2240,
    "durationMilliseconds": 1040,
    "text": "Thank you for calling Contoso.",
    "words": [
        {
            "text": "Thank",
            "offsetMilliseconds": 2240,
            "durationMilliseconds": 200
        },
        {
            "text": "you",
            "offsetMilliseconds": 2440,
            "durationMilliseconds": 80
        },
        {
            "text": "for",
            "offsetMilliseconds": 2520,
            "durationMilliseconds": 120
        },
        {
            "text": "calling",
            "offsetMilliseconds": 2640,
            "durationMilliseconds": 200
        },
        {
            "text": "Contoso.",
            "offsetMilliseconds": 2840,
            "durationMilliseconds": 440
        }
    ],
    "locale": "en-US",
    "confidence": 0.93554276
},
{
    "offsetMilliseconds": 3280,
```

```
"durationMilliseconds": 640,
"text": "How can I help?",  

"words": [  

  {  

    "text": "How",  

    "offsetMilliseconds": 3280,  

    "durationMilliseconds": 120  

  },  

  {  

    "text": "can",  

    "offsetMilliseconds": 3440,  

    "durationMilliseconds": 120  

  },  

  {  

    "text": "I",  

    "offsetMilliseconds": 3560,  

    "durationMilliseconds": 40  

  },  

  {  

    "text": "help?",  

    "offsetMilliseconds": 3600,  

    "durationMilliseconds": 320  

  }  

],  

"locale": "en-US",  

"confidence": 0.93554276
},  

{  

  "offsetMilliseconds": 5040,  

  "durationMilliseconds": 400,  

  "text": "Hi there.",  

  "words": [  

    {  

      "text": "Hi",  

      "offsetMilliseconds": 5040,  

      "durationMilliseconds": 240  

    },  

    {  

      "text": "there.",  

      "offsetMilliseconds": 5280,  

      "durationMilliseconds": 160  

    }  

],  

"locale": "en-US",  

"confidence": 0.93554276
},  

{  

  "offsetMilliseconds": 5440,  

  "durationMilliseconds": 800,  

  "text": "My name is Mary.",  

  "words": [  

    {  

      "text": "My",  

      "offsetMilliseconds": 5440,  

      "durationMilliseconds": 80
    }
]
```

```
        },
        {
            "text": "name",
            "offsetMilliseconds": 5520,
            "durationMilliseconds": 120
        },
        {
            "text": "is",
            "offsetMilliseconds": 5640,
            "durationMilliseconds": 80
        },
        {
            "text": "Mary.",
            "offsetMilliseconds": 5720,
            "durationMilliseconds": 520
        }
    ],
    "locale": "en-US",
    "confidence": 0.93554276
},
// More transcription results...
// Redacted for brevity
{
    "offsetMilliseconds": 180320,
    "durationMilliseconds": 680,
    "text": "Thank you for your help.",
    "words": [
        {
            "text": "Thank",
            "offsetMilliseconds": 180320,
            "durationMilliseconds": 160
        },
        {
            "text": "you",
            "offsetMilliseconds": 180480,
            "durationMilliseconds": 80
        },
        {
            "text": "for",
            "offsetMilliseconds": 180560,
            "durationMilliseconds": 120
        },
        {
            "text": "your",
            "offsetMilliseconds": 180680,
            "durationMilliseconds": 120
        },
        {
            "text": "help.",
            "offsetMilliseconds": 180800,
            "durationMilliseconds": 200
        }
    ],
    "locale": "en-US",
    "confidence": 0.92022026
```

```

},
{
  "offsetMilliseconds": 181960,
  "durationMilliseconds": 280,
  "text": "Thank you.",
  "words": [
    {
      "text": "Thank",
      "offsetMilliseconds": 181960,
      "durationMilliseconds": 200
    },
    {
      "text": "you.",
      "offsetMilliseconds": 182160,
      "durationMilliseconds": 80
    }
  ],
  "locale": "en-US",
  "confidence": 0.92022026
}
]
}

```

! Note

Speech service is an elastic service. If you receive 429 error code (too many requests), please follow the [best practices to mitigate throttling during autoscaling](#).

Request configuration options

Here are some property options to configure a transcription when you call the [Transcriptions - Transcribe](#) operation.

[] [Expand table](#)

Property	Description	Required or optional
<code>channels</code>	<p>The list of zero-based indices of the channels to be transcribed separately. Up to two channels are supported unless diarization is enabled. By default, the fast transcription API merges all input channels into a single channel and then performs the transcription. If this isn't desirable, channels can be transcribed independently without merging.</p> <p>If you want to transcribe the channels from a stereo audio file separately, you need to specify <code>[0,1]</code>, <code>[0]</code>, or <code>[1]</code>.</p>	Optional

Property	Description	Required or optional
	<p>Otherwise, stereo audio is merged to mono and only a single channel is transcribed.</p> <p>If the audio is stereo and diarization is enabled, then you can't set the <code>channels</code> property to <code>[0,1]</code>. The Speech service doesn't support diarization of multiple channels.</p> <p>For mono audio, the <code>channels</code> property is ignored, and the audio is always transcribed as a single channel.</p>	
<code>diarization</code>	<p>The diarization configuration. Diarization is the process of recognizing and separating multiple speakers in one audio channel. For example, specify <code>"diarization":</code></p> <pre>{"maxSpeakers": 2, "enabled": true}</pre> <p>Then the transcription file contains <code>speaker</code> entries (such as <code>"speaker": 0</code> or <code>"speaker": 1</code>) for each transcribed phrase.</p>	Optional
<code>locales</code>	<p>The list of locales that should match the expected locale of the audio data to transcribe.</p> <p>If you know the locale of the audio file, you can specify it to improve transcription accuracy and minimize the latency. If a single locale is specified, that locale is used for transcription.</p> <p>But if you're not sure about the locale, you can specify multiple locales to use language identification. Language identification might be more accurate with a more precise list of candidate locales.</p> <p>If you don't specify any locale, then the Speech service will use the latest multi-lingual model to identify the locale and transcribe continuously.</p> <p>You can get the latest supported languages via the Transcriptions - List Supported Locales REST API (API version 2024-11-15 or later). For more information about locales, see the Speech service language support documentation.</p>	Optional but recommended if you know the expected locale.
<code>phraseList</code>	<p>Phrase list is a list of words or phrases provided ahead of time to help improve their recognition. Adding a phrase to a phrase list increases its importance, thus making it more likely to be recognized. For example, specify <code>phraseList":</code></p> <pre>{"phrases": ["Contoso", "Jessie", "Rehaan"]}</pre> <p>Phrase List is supported via API version 2025-10-15. For more information, see Improve recognition accuracy with phrase list.</p>	Optional

Property	Description	Required or optional
profanityFilterMode	<p>Specifies how to handle profanity in recognition results. Accepted values are <code>None</code> to disable profanity filtering, <code>Masked</code> to replace profanity with asterisks, <code>Removed</code> to remove all profanity from the result, or <code>Tags</code> to add profanity tags. The default value is <code>Masked</code>.</p>	Optional

Related content

- [Fast transcription REST API reference](#)
- [Speech to text supported languages](#)
- [Batch transcription](#)

Last updated on 10/24/2025

What is batch transcription?

08/27/2025

Batch transcription is used to transcribe a large amount of audio data in storage. Both the [Speech to text REST API](#) and [Speech CLI](#) support batch transcription.

You should provide multiple files per request or point to an Azure Blob Storage container with the audio files to transcribe. The batch transcription service can handle a large number of submitted transcriptions. The service transcribes the files concurrently, which reduces the turnaround time.

How does it work?

With batch transcriptions, you submit the audio data, and then retrieve transcription results asynchronously. The service transcribes the audio data and stores the results in a storage container. You can then retrieve the results from the storage container.

Tip

For a low or no-code solution, you can use the [Batch Speech to text Connector](#) in Power Platform applications such as Power Automate, Power Apps, and Logic Apps. See the [Power automate batch transcription](#) guide to get started.

To use the batch transcription REST API:

1. [Locate audio files for batch transcription](#) - You can upload your own data or use existing audio files via public URI or [shared access signature \(SAS\)](#) URI.
2. [Create a batch transcription](#) - Submit the transcription job with parameters such as the audio files, the transcription language, and the transcription model.
3. [Get batch transcription results](#) - Check transcription status and retrieve transcription results asynchronously.

Important

Batch transcription jobs are scheduled on a best-effort basis. At peak hours it might take up to 30 minutes or longer for a transcription job to start processing. See how to check the current status of a batch transcription job in [this section](#).

Best practices for improving performance

Request size: Batch transcription is asynchronous, and requests are processed one at a time in each region. Submitting jobs at a higher rate does not speed up processing. For example, sending 600 or 6,000 requests per minute has no effect on throughput. We recommend submitting ~1000 files in a single `Transcription_Create` request. This way, you send fewer requests overall.

Time distribution: Distribute your requests over time: Submit them across several hours rather than sending them all within a few minutes. Backend processing maintains a stable performance level due to fixed bandwidth, so sending requests too quickly doesn't improve performance.

Job monitoring: [When monitoring job status](#), polling every few seconds is unnecessary. If you submit multiple jobs, only the first job will be processed initially; subsequent jobs will wait until the first job completes. Polling all jobs frequently increases system load without benefit. Checking the status every 10 minutes is sufficient, and polling more often than once per minute is not recommended.

- Because of the sequential processing, you can get job status by checking only a subset of the files: check the first 100 files, and if they're not completed, later batches are likely not completed either. We recommend waiting at least one minute (ideally five minutes) before checking again.

Avoid peak traffic for API calls: The `ListFiles`, `Update`, and `Get` API calls behave similarly to the `Create` call and should be minimized during peak traffic times.

Load balancing: To optimize throughput for large-scale batch transcription, consider distributing your jobs across multiple supported Azure regions. This approach can help balance load and reduce overall processing time, provided your data and compliance requirements allow for multi-region usage. Review [region availability](#) and ensure your storage and resources are accessible from each region you plan to use.

Related content

- [Locate audio files for batch transcription](#)
- [Create a batch transcription](#)
- [Get batch transcription results](#)
- [See batch transcription code samples at GitHub ↗](#)

Locate audio files for batch transcription

05/25/2025

Batch transcription is used to transcribe a large amount of audio in storage. Batch transcription can access audio files from inside or outside of Azure.

When source audio files are stored outside of Azure, they can be accessed via a public URI (such as "<https://crbn.us/hello.wav>"). Files should be directly accessible; URIs that require authentication or that invoke interactive scripts before the file can be accessed aren't supported.

Audio files that are stored in Azure Blob storage can be accessed via one of two methods:

- [Trusted Azure services security mechanism](#)
- [Shared access signature \(SAS\) URI](#).

You can specify one or multiple audio files when creating a transcription. We recommend that you provide multiple files per request or point to an Azure Blob storage container with the audio files to transcribe. The batch transcription service can handle a large number of submitted transcriptions. The service transcribes the files concurrently, which reduces the turnaround time.

Supported audio formats and codecs

The [batch transcription API](#) and [fast transcription API](#) support multiple formats and codecs, such as:

- WAV
- MP3
- OPUS/OGG
- FLAC
- WMA
- AAC
- ALAW in WAV container
- MULAW in WAV container
- AMR
- WebM
- SPEEX

Note

Batch transcription service integrates [GStreamer](#) and might accept more formats and codecs without returning errors. We suggest using lossless formats such as WAV (PCM encoding) and FLAC to ensure best transcription quality.

Upload to Azure Blob Storage

When audio files are located in an [Azure Blob Storage](#) account, you can request transcription of individual audio files or an entire Azure Blob Storage container. You can also [write transcription results](#) to a Blob container.

 **Note**

For blob and container limits, see [batch transcription quotas and limits](#).

Azure portal

Follow these steps to create a storage account and upload wav files from your local directory to a new container.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. [Create a Storage account resource](#) in the Azure portal. Use the same subscription and resource group as your Speech resource.
3. Select the Storage account.
4. In the **Data storage** group in the left pane, select **Containers**.
5. Select **+ Container**.
6. Enter a name for the new container and select **Create**.
7. Select the new container.
8. Select **Upload**.
9. Choose the files to upload and select **Upload**.

Trusted Azure services security mechanism

This section explains how to set up and limit access to your batch transcription source audio files in an Azure Storage account using the [trusted Azure services security mechanism](#).

 **Note**

With the trusted Azure services security mechanism, you need to use [Azure Blob storage](#) to store audio files. Usage of [Azure Files](#) isn't supported.

If you perform all actions in this section, your Storage account is configured as follows:

- Access to all external network traffic is prohibited.
- Access to Storage account using Storage account key is prohibited.
- Access to Storage account blob storage using [shared access signatures \(SAS\)](#) is prohibited.
- Access to the selected Speech resource is allowed using the resource [system assigned managed identity](#).

So in effect your Storage account becomes completely "locked" and can't be used in any scenario apart from transcribing audio files that were already present by the time the new configuration was applied. You should consider this configuration as a model as far as the security of your audio data is concerned and customize it according to your needs.

For example, you can allow traffic from selected public IP addresses and Azure Virtual networks. You can also set up access to your Storage account using [private endpoints](#) (see as well [this tutorial](#)), re-enable access using Storage account key, allow access to other Azure trusted services, etc.

Note

Using [private endpoints for Speech](#) isn't required to secure the storage account. You can use a private endpoint for batch transcription API requests, while separately accessing the source audio files from a secure storage account, or the other way around.

By following the steps below, you severely restrict access to the storage account. Then you assign the minimum required permissions for Speech resource managed identity to access the Storage account.

Enable system assigned managed identity for the Speech resource

Follow these steps to enable system assigned managed identity for the Speech resource that you use for batch transcription.

1. Go to the [Azure portal](#)  and sign in to your Azure account.
2. Select the Speech resource.

3. In the **Resource Management** group in the left pane, select **Identity**.

4. On the **System assigned** tab, select **On** for the status.

ⓘ Important

User assigned managed identity won't meet requirements for the batch transcription storage account scenario. Be sure to enable system assigned managed identity.

5. Select **Save**

Now the managed identity for your Speech resource can be granted access to your storage account.

Restrict access to the storage account

Follow these steps to restrict access to the storage account.

ⓘ Important

Upload audio files in a Blob container before locking down the storage account access.

1. Go to the [Azure portal](#) and sign in to your Azure account.

2. Select the Storage account.

3. In the **Settings** group in the left pane, select **Configuration**.

4. Select **Disabled** for **Allow Blob public access**.

5. Select **Disabled** for **Allow storage account key access**

6. Select **Save**.

For more information, see [Prevent anonymous public read access to containers and blobs](#) and [Prevent Shared Key authorization for an Azure Storage account](#).

Configure Azure Storage firewall

Having restricted access to the Storage account, you need to grant access to specific managed identities. Follow these steps to add access for the Speech resource.

1. Go to the [Azure portal](#) and sign in to your Azure account.

2. Select the Storage account.

3. In the **Security + networking** group in the left pane, select **Networking**.

4. In the **Firewalls and virtual networks** tab, select **Enabled from selected virtual networks and IP addresses**.
5. Deselect all check boxes.
6. Make sure **Microsoft network routing** is selected.
7. Under the **Resource instances** section, select **Microsoft.CognitiveServices/accounts** as the resource type and select your Speech resource as the instance name.
8. Select **Save**.

 **Note**

It might take up to 5 min for the network changes to propagate.

Although by now the network access is permitted, the Speech resource can't yet access the data in the Storage account. You need to assign a specific access role for Speech resource managed identity.

Assign resource access role

Follow these steps to assign the **Storage Blob Data Reader** role to the managed identity of your Speech resource.

 **Important**

You need to be assigned the *Owner* role of the Storage account or higher scope (like Subscription) to perform the operation in the next steps. This is because only the *Owner* role can assign roles to others. See details [here](#).

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. Select **Access Control (IAM)** menu in the left pane.
4. Select **Add role assignment** in the **Grant access to this resource** tile.
5. Select **Storage Blob Data Reader** under **Role** and then select **Next**.
6. Select **Managed identity** under **Members > Assign access to**.
7. Assign the managed identity of your Speech resource and then select **Review + assign**.

Add role assignment

[Got feedback?](#)

Role Members Conditions (optional) **Review + assign**

Role	Storage Blob Data Reader		
Scope	/subscriptions/<redacted>/resourcegroups/speech-rg/providers/Microsoft.Storage/storageAccounts/contosostorage		
Members	Name	Object ID	Type
	contoso-speech	<redacted>	Speech service ⓘ
Description	No description		
Condition	None		

Review + assign [Previous](#)

8. After confirming the settings, select **Review + assign**

Now the Speech resource managed identity has access to the Storage account and can access the audio files for batch transcription.

With system assigned managed identity, you use a plain Storage Account URL (no SAS or other additions) when you [create a batch transcription](#) request. For example:

JSON

```
{
  "contentContainerUrl":
  "https://<storage_account_name>.blob.core.windows.net/<container_name>"
}
```

You could otherwise specify individual files in the container. For example:

JSON

```
{
  "contentUrls": [
    "https://<storage_account_name>.blob.core.windows.net/<container_name>/<file_name_1>",
    "https://<storage_account_name>.blob.core.windows.net/<container_name>/<file_name_2>",
    ]
}
```

SAS URL for batch transcription

A shared access signature (SAS) is a URI that grants restricted access to an Azure Storage container. Use it when you want to grant access to your batch transcription files for a specific time range without sharing your storage account key.

Tip

If the container with batch transcription source files should only be accessed by your Speech resource, use the [trusted Azure services security mechanism](#) instead.

Azure portal

Follow these steps to generate a SAS URL that you can use for batch transcriptions.

1. Complete the steps in [Azure Blob Storage upload](#) to create a Storage account and upload audio files to a new container.
2. Select the new container.
3. In the **Settings** group in the left pane, select **Shared access tokens**.
4. Select **+ Container**.
5. Select **Read and List** for Permissions.

The screenshot shows the Azure portal interface for managing a storage container named 'contoso-container'. The 'Shared access tokens' section is selected in the left sidebar. The main area displays the configuration for generating a SAS token. It includes fields for 'Signing method' (set to 'Account key'), 'Key 1' (selected), and 'Stored access policy' (set to 'None'). Under 'Permissions', 'Read' and 'List' are checked. Two time ranges are defined: from 3:06:36 PM to 11:06:36 PM, both covering '(US & Canada)'. Below this, 'Allowed IP addresses' is set to 'for example, 168.1.5.65 or 168.1.5.65-168.1...'. At the bottom, 'Generate SAS token and URL' is a blue button.

6. Enter the start and expiry times for the SAS URI, or leave the defaults.

7. Select Generate SAS token and URL.

You use the SAS URL when you [create a batch transcription](#) request. For example:

JSON

```
{  
    "contentContainerUrl":  
    "https://<storage_account_name>.blob.core.windows.net/<container_name>?SAS_TOKEN"  
}
```

You could otherwise specify individual files in the container. You must generate and use a different SAS URL with read (r) permissions for each file. For example:

JSON

```
{  
    "contentUrls": [  
  
        "https://<storage_account_name>.blob.core.windows.net/<container_name>/<file_name_1>?SAS_TOKEN_1",  
  
        "https://<storage_account_name>.blob.core.windows.net/<container_name>/<file_name_2>?SAS_TOKEN_2"  
    ]  
}
```

Related content

- [Learn more about batch transcription](#)
- [Create a batch transcription](#)
- [Get batch transcription results](#)
- [See batch transcription code samples at GitHub ↗](#)

Create a batch transcription

With batch transcriptions, you submit [audio data](#) in a batch. The service transcribes the audio data and stores the results in a storage container. You can then [retrieve the results](#) from the storage container.

Batch transcription completion can take several minutes to hours, depending on the size of the audio data and the number of files submitted. Even the same size of audio data can take different amounts of time to transcribe, depending on service load and other factors. The service doesn't provide a way to estimate the time it takes to transcribe a batch of audio data.

💡 Tip

If you need consistent fast speed for audio files less than 2 hours long and less than 300 MB in size, consider using the [fast transcription API](#) instead.

Prerequisites

You need an [Azure AI Foundry resource for Speech](#).

Create a transcription job

To create a batch transcription job, use the [Transcriptions - Submit](#) operation of the [speech to text REST API](#). Construct the request body according to the following instructions:

- You must set either the `contentContainerUrl` or `contentUrls` property. For more information about Azure blob storage for batch transcription, see [Locate audio files for batch transcription](#).
- Set the required `locale` property. This value should match the expected locale of the audio data to transcribe. You can't change the locale later.
- Set the required `displayName` property. Choose a transcription name that you can refer to later. The transcription name doesn't have to be unique and can be changed later.
- Set the required `timeToLiveHours` property. This property specifies how long the transcription should be kept in the system after it completed. The shortest supported duration is 6 hours, the longest supported duration is 31 days. The recommended value is 48 hours (two days) when data is consumed directly.
- Optionally, to use a model other than the base model, set the `model` property to the model ID. For more information, see [Use a custom model](#) and [Use a Whisper model](#).
- Optionally, set the `wordLevelTimestampsEnabled` property to `true` to enable word-level timestamps in the transcription results. The default value is `false`. For Whisper models,

set the `displayFormWordLevelTimestampsEnabled` property instead. Whisper is a display-only model, so the lexical field isn't populated in the transcription.

- Optionally, set the `languageIdentification` property. Language identification is used to identify languages spoken in audio when compared against a list of [supported languages](#). If you set the `languageIdentification` property, then you must also set `languageIdentification.candidateLocales` with candidate locales.

For more information, see [Request configuration options](#).

Make an HTTP POST request that uses the URI as shown in the following [Transcriptions - Submit](#) example.

- Replace `YourSpeechResourceKey` with your Azure AI Foundry resource key.
- Replace `YourServiceRegion` with your Azure AI Foundry resource region.
- Set the request body properties as previously described.

Azure CLI

```
curl -v -X POST -H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey" -H "Content-Type: application/json" -d '{  
    "contentUrls": [  
        "https://crbn.us/hello.wav",  
        "https://crbn.us/whatstheweatherlike.wav"  
    ],  
    "locale": "en-US",  
    "displayName": "My Transcription",  
    "model": null,  
    "properties": {  
        "wordLevelTimestampsEnabled": true,  
        "languageIdentification": {  
            "candidateLocales": [  
                "en-US", "de-DE", "es-ES"  
            ],  
            "mode": "Continuous"  
        },  
        "timeToLiveHours": 48  
    }  
}'  
https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions:submit?api-version=2024-11-15
```

You should receive a response body in the following format:

JSON

```
{  
    "self":  
        "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/788a1f24-f980-4809-8978-e5cf41f77b35?api-version=2024-11-15",  
    "status": "Queued",  
    "id": "788a1f24-f980-4809-8978-e5cf41f77b35",  
    "language": "en-US",  
    "display_name": "My Transcription",  
    "model": null,  
    "lexical_form": null,  
    "word_level_timestamps": true,  
    "candidate_locales": ["en-US", "de-DE", "es-ES"],  
    "mode": "Continuous",  
    "time_to_live_hours": 48, ...}
```

```
"displayName": "My Transcription 2",
"locale": "en-US",
"createdDateTime": "2025-05-24T03:20:39Z",
"lastActionDateTime": "2025-05-24T03:20:39Z",
"links": {
  "files": "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/788a1f24-f980-4809-8978-e5cf41f77b35/files?api-version=2024-11-15"
},
"properties": {
  "wordLevelTimestampsEnabled": true,
  "displayFormWordLevelTimestampsEnabled": false,
  "channels": [
    0,
    1
  ],
  "punctuationMode": "DictatedAndAutomatic",
  "profanityFilterMode": "Masked",
  "timeToLiveHours": 48,
  "languageIdentification": {
    "candidateLocales": [
      "en-US",
      "de-DE",
      "es-ES"
    ],
    "mode": "Continuous"
  }
},
"status": "NotStarted"
}
```

The top-level `self` property in the response body is the transcription's URI. Use this URI to [get](#) details such as the URI of the transcriptions and transcription report files. You also use this URI to [update](#) or [delete](#) a transcription.

You can query the status of your transcriptions with the [Transcriptions - Get](#) operation.

Call [Transcriptions - Delete](#) regularly from the service, after you retrieve the results.

Alternatively, set the `timeToLive` property to ensure the eventual deletion of the results.

💡 Tip

You can also try the Batch Transcription API using Python, C#, or Node.js on [GitHub](#).

Request configuration options

Here are some property options to configure a transcription when you call the [Transcriptions - Submit](#) operation. You can find more examples on the same page, such as [creating a](#)

transcription with language identification.

 Expand table

Property	Description
<code>channels</code>	An array of channel numbers to process. Channels <code>0</code> and <code>1</code> are transcribed by default.
<code>contentContainerUrl</code>	<p>You can submit individual audio files or a whole storage container.</p> <p>You must specify the audio data location by using either the <code>contentContainerUrl</code> or <code>contentUrls</code> property. For more information about Azure blob storage for batch transcription, see Locate audio files for batch transcription.</p> <p>This property isn't returned in the response.</p>
<code>contentUrls</code>	<p>You can submit individual audio files or a whole storage container.</p> <p>You must specify the audio data location by using either the <code>contentContainerUrl</code> or <code>contentUrls</code> property. For more information, see Locate audio files for batch transcription.</p> <p>This property isn't returned in the response.</p>
<code>destinationContainerUrl</code>	The result can be stored in an Azure container. If you don't specify a container, the Speech service stores the results in a container managed by Microsoft. When the transcription job is deleted, the transcription result data is also deleted. For more information, such as the supported security scenarios, see Specify a destination container URL .
<code>diarization</code>	<p>Indicates that the Speech service should attempt diarization analysis on the input, which is expected to be a mono channel that contains multiple voices. The feature isn't available with stereo recordings.</p> <p>Diarization is the process of separating speakers in audio data. The batch pipeline can recognize and separate multiple speakers on mono channel recordings.</p> <p>Specify the minimum and maximum number of people who might be speaking. You must also set the <code>diarizationEnabled</code> property to <code>true</code>. The transcription file contains a <code>speaker</code> entry for each transcribed phrase.</p> <p>You need to use this property when you expect three or</p>

Property	Description
	<p>more speakers. For two speakers, setting <code>diarizationEnabled</code> property to <code>true</code> is enough. For an example of the property usage, see Transcriptions - Submit.</p>
	<p>The maximum number of speakers for diarization must be less than 36 and more or equal to the <code>minCount</code> property. For an example, see Transcriptions - Submit.</p>
	<p>When this property is selected, source audio length can't exceed 240 minutes per file.</p>
	<p>Note: This property is only available with Speech to text REST API version 3.1 and later. If you set this property with any previous version, such as version 3.0, it's ignored and only two speakers are identified.</p>
<code>diarizationEnabled</code>	<p>Specifies that the Speech service should attempt diarization analysis on the input, which is expected to be a mono channel that contains two voices. The default value is <code>false</code>.</p>
	<p>For three or more voices you also need to use property <code>diarization</code>. Use only with Speech to text REST API version 3.1 and later.</p>
	<p>When this property is selected, source audio length can't exceed 240 minutes per file.</p>
<code>displayName</code>	<p>The name of the batch transcription. Choose a name that you can refer to later. The display name doesn't have to be unique.</p>
	<p>This property is required.</p>
<code>displayFormWordLevelTimestampsEnabled</code>	<p>Specifies whether to include word-level timestamps on the display form of the transcription results. The results are returned in the <code>displayWords</code> property of the transcription file. The default value is <code>false</code>.</p>
	<p>Note: This property is only available with Speech to text REST API version 3.1 and later.</p>
<code>languageIdentification</code>	<p>Language identification is used to identify languages spoken in audio when compared against a list of supported languages.</p>
	<p>If you set the <code>languageIdentification</code> property, then you must also set its enclosed <code>candidateLocales</code> property.</p>

Property	Description
<code>languageIdentification.candidateLocales</code>	<p>The candidate locales for language identification, such as <code>"properties": { "languageIdentification": { "candidateLocales": ["en-US", "de-DE", "es-ES"] } }</code>. A minimum of two and a maximum of ten candidate locales, including the main locale for the transcription, is supported.</p>
<code>locale</code>	<p>The locale of the batch transcription. This value should match the expected locale of the audio data to transcribe. The locale can't be changed later.</p> <p>This property is required.</p>
<code>model</code>	<p>You can set the <code>model</code> property to use a specific base model or custom speech model. If you don't specify the <code>model</code>, the default base model for the locale is used. For more information, see Use a custom model and Use a Whisper model.</p>
<code>profanityFilterMode</code>	<p>Specifies how to handle profanity in recognition results. Accepted values are <code>None</code> to disable profanity filtering, <code>Masked</code> to replace profanity with asterisks, <code>Removed</code> to remove all profanity from the result, or <code>Tags</code> to add profanity tags. The default value is <code>Masked</code>.</p>
<code>punctuationMode</code>	<p>Specifies how to handle punctuation in recognition results. Accepted values are <code>None</code> to disable punctuation, <code>Dictated</code> to imply explicit (spoken) punctuation, <code>Automatic</code> to let the decoder deal with punctuation, or <code>DictatedAndAutomatic</code> to use dictated and automatic punctuation. The default value is <code>DictatedAndAutomatic</code>.</p> <p>This property isn't applicable for Whisper models.</p>
<code>timeToLiveHours</code>	<p>This required property specifies how long the transcription should be kept in the system after it completed.</p> <p>Once the transcription reaches the time to live after completion (successful or failed) it's automatically deleted.</p> <p>The shortest supported duration is 6 hours, the longest supported duration is 31 days. The recommended value is 48 hours (two days) when data is consumed directly.</p> <p>As an alternative, you can call Transcriptions - Delete regularly after you retrieve the transcription results.</p>
<code>wordLevelTimestampsEnabled</code>	<p>Specifies if word level timestamps should be included in the output. The default value is <code>false</code>.</p>

Property	Description
	This property isn't applicable for Whisper models. Whisper is a display-only model, so the lexical field isn't populated in the transcription.

Use a custom model

Batch transcription uses the default base model for the locale that you specify. You don't need to set any properties to use the default base model.

Optionally, you can modify the previous [create transcription example](#) by setting the `model` property to use a specific base model or [custom speech](#) model.

Azure CLI

```
curl -v -X POST -H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey" -H "Content-Type: application/json" -d '{
  "contentUrls": [
    "https://crbn.us/hello.wav",
    "https://crbn.us/whatstheweatherlike.wav"
  ],
  "locale": "en-US",
  "displayName": "My Transcription",
  "model": {
    "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base/ccccdddd-2222-eeee-3333-fffff4444aaaa"
  },
  "properties": {
    "wordLevelTimestampsEnabled": true,
  }
}'
"https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions:submit?api-version=2024-11-15"
```

To use a custom speech model for batch transcription, you need the model's URI. The top-level `self` property in the response body is the model's URI. You can retrieve the model location when you create or get a model. For more information, see the JSON response example in [Create a model](#).

Tip

A [hosted deployment endpoint](#) isn't required to use custom speech with the batch transcription service. You can conserve resources if you use the [custom speech model](#) only for batch transcription.

Batch transcription requests for expired models fail with a 4xx error. Set the `model` property to a base model or custom model that isn't expired. Otherwise don't include the `model` property to always use the latest base model. For more information, see [Choose a model](#) and [Custom speech model lifecycle](#).

Use a Whisper model

Azure AI Speech supports OpenAI's Whisper model by using the batch transcription API. You can use the Whisper model for batch transcription.

ⓘ Note

Azure OpenAI in Azure AI Foundry Models also supports OpenAI's Whisper model for speech to text with a synchronous REST API. To learn more, see [Speech to text with the Azure OpenAI Whisper model](#). For more information about when to use Azure AI Speech vs. Azure OpenAI in Azure AI Foundry Models, see [What is the Whisper model?](#)

To use a Whisper model for batch transcription, you need to set the `model` property. Whisper is a display-only model, so the lexical field isn't populated in the response.

ⓘ Important

Batch transcription using Whisper models is available in a subset of regions that support batch transcription. For the current list of supported regions, see the [Speech service regions table](#). Note that Whisper model support may be limited to specific regions within those that support batch transcription.

You can make a [Models - List Base Models](#) request to get available base models for all locales.

Make an HTTP GET request as shown in the following example for the `eastus` region. Replace `YourSpeechResourceKey` with your Azure AI Foundry resource key. Replace `eastus` if you're using a different region.

Azure CLI

```
curl -v -X GET "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base?api-version=2024-11-15" -H "Ocp-Apim-Subscription-Key: YourSpeechResoureKey"
```

By default, only the 100 oldest base models are returned. Use the `skip` and `top` query parameters to page through the results. For example, the following request returns the next 100 base models after the first 100.

Azure CLI

```
curl -v -X GET "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base?api-version=2024-11-15&skip=100&top=100" -H "Ocp-Apim-Subscription-Key: YourSpeechResoureKey"
```

The `displayName` property of a Whisper model contains "Whisper" as shown in this example. Whisper is a display-only model, so the lexical field isn't populated in the transcription.

JSON

```
{
  "links": {
    "manifest": "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base/69adf293-9664-4040-932b-02ed16332e00/manifest?api-version=2024-11-15"
  },
  "properties": {
    "deprecationDates": {
      "adaptationDateTime": "2025-04-15T00:00:00Z",
      "transcriptionDateTime": "2026-04-15T00:00:00Z"
    },
    "features": {
      "supportsAdaptationsWith": [
        "Acoustic"
      ],
      "supportsTranscriptionsSubmit": true,
      "supportsTranscriptionsTranscribe": false,
      "supportsEndpoints": false,
      "supportsTranscriptionsOnSpeechContainers": false,
      "supportedOutputFormats": [
        "Display"
      ]
    },
    "chargeForAdaptation": true
  },
  "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base/69adf293-9664-4040-932b-02ed16332e00?api-version=2024-11-15",
  "displayName": "20240228 Whisper Large V2",
  "description": "OpenAI Whisper Model in Azure AI Speech (Whisper v2-large)",
  "locale": "en-US",
  "createdDateTime": "2024-02-29T15:46:31Z",
  "lastActionDateTime": "2024-02-29T15:51:53Z",
```

```
    "status": "Succeeded"  
},
```

You set the full model URI as shown in this example for the `eastus` region. Replace `YourSpeechResourceKey` with your Azure AI Foundry resource key. Replace `eastus` if you're using a different region.

Azure CLI

```
curl -v -X POST -H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey" -H "Content-Type: application/json" -d '{  
    "contentUrls": [  
        "https://crbn.us/hello.wav",  
        "https://crbn.us/whatstheweatherlike.wav"  
    ],  
    "locale": "en-US",  
    "displayName": "My Transcription",  
    "model": {  
        "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/models/base/69adf293-9664-4040-932b-02ed16332e00?api-version=2024-11-15"  
    },  
    "properties": {  
        "wordLevelTimestampsEnabled": true,  
    },  
}' "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions:submit?api-version=2024-11-15"
```

Specify a destination container URL

The transcription result can be stored in an Azure container. If you don't specify a container, the Speech service stores the results in a container managed by Microsoft. In that case, when the transcription job is deleted, the transcription result data is also deleted.

You can store the results of a batch transcription to a writable Azure Blob storage container using option `destinationContainerUrl` in the [batch transcription creation request](#). This option uses only an [ad hoc SAS](#) URI and doesn't support [Trusted Azure services security mechanism](#). This option also doesn't support Access policy based SAS. **The Storage account resource of the destination container must allow all external traffic.**

If you want to store the transcription results in an Azure Blob storage container by using the [Trusted Azure services security mechanism](#), consider using [Bring your own storage \(BYOS\)](#). For more information, see [Use the Bring your own storage \(BYOS\) Azure AI Foundry resource for speech to text](#).

Related content

- [Learn more about batch transcription](#)
 - [Locate audio files for batch transcription](#)
 - [Get batch transcription results](#)
 - [See batch transcription code samples at GitHub ↗](#)
-

Last updated on 10/31/2025

Get batch transcription results

05/25/2025

To get transcription results, first check the [status](#) of the transcription job. If the job is completed, you can [retrieve](#) the transcriptions and transcription report.

Get transcription status

To get the status of the transcription job, call the [Transcriptions - Get](#) operation of the Speech to text REST API.

Important

Batch transcription jobs are scheduled on a best-effort basis. At peak hours, it might take up to 30 minutes or longer for a transcription job to start processing. Most of the time during the execution the transcription status is `Running`. The reason is because the job is assigned the `Running` status the moment it moves to the batch transcription backend system. When the base model is used, this assignment happens almost immediately; it's slightly slower for custom models. Thus, the amount of time a transcription job spends in the `Running` state doesn't correspond to the actual transcription time but also includes waiting time in the internal queues.

Make an HTTP GET request using the URI as shown in the following example. Replace `YourTranscriptionId` with your transcription ID, replace `YourSpeechResourceKey` with your Speech resource key, and replace `YourServiceRegion` with your Speech resource region.

Azure CLI

```
curl -v -X GET  
"https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions/  
YourTranscriptionId?api-version=2024-11-15" -H "Ocp-Apim-Subscription-Key:  
YourSpeechResourceKey"
```

You should receive a response body in the following format:

JSON

```
{  
  "self":  
    "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/5cff1d03-  
118f-4c4c-b3ba-e1f1cd88c14d?api-version=2024-11-15",  
  "displayName": "My Transcription",
```

```
"locale": "en-US",
"createdDateTime": "2025-05-24T13:36:57Z",
"lastActionDateTime": "2025-05-24T13:37:13Z",
"links": {
  "files": "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/5cff1d03-118f-4c4c-b3ba-e1f1cd88c14d/files?api-version=2024-11-15"
},
"properties": {
  "wordLevelTimestampsEnabled": true,
  "displayFormWordLevelTimestampsEnabled": false,
  "channels": [
    0,
    1
  ],
  "punctuationMode": "DictatedAndAutomatic",
  "profanityFilterMode": "Masked",
  "timeToLiveHours": 48,
  "languageIdentification": {
    "candidateLocales": [
      "en-US",
      "de-DE",
      "es-ES"
    ],
    "mode": "Continuous"
  },
  "durationMilliseconds": 3000
},
"status": "Succeeded"
}
```

The `status` property indicates the current status of the transcriptions. The transcriptions and transcription report are available when the transcription status is `Succeeded`.

Get transcription results

The [Transcriptions - List Files](#) operation returns a list of result files for a transcription. A [transcription report](#) file is provided for each submitted batch transcription job. In addition, one [transcription](#) file (the end result) is provided for each successfully transcribed audio file.

Make an HTTP GET request using the "files" URI from the previous response body. Replace `YourTranscriptionId` with your transcription ID, replace `YourSpeechResourceKey` with your Speech resource key, and replace `YourServiceRegion` with your Speech resource region.

Azure CLI

```
curl -v -X GET
"https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions
```

```
/YourTranscriptionId/files?api-version=2024-11-15" -H "Ocp-Apim-Subscription-Key: YourSpeechResoureKey"
```

You should receive a response body in the following format:

JSON

```
{
  "values": [
    {
      "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/5cff1d03-118f-4c4c-b3ba-e1f1cd88c14d/files/ec226a24-d3c7-4ae4-b59e-49d5bdab492e?api-version=2024-11-15",
      "name": "contenturl_0.json",
      "kind": "Transcription",
      "links": {
        "contentUrl": "YourTranscriptionUrl"
      },
      "properties": {
        "size": 1230
      },
      "createdDateTime": "2025-05-24T13:37:12Z"
    },
    {
      "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/5cff1d03-118f-4c4c-b3ba-e1f1cd88c14d/files/078cd816-7944-4619-a6a6-bc52fb000f8c?api-version=2024-11-15",
      "name": "contenturl_1.json",
      "kind": "Transcription",
      "links": {
        "contentUrl": "YourTranscriptionUrl"
      },
      "properties": {
        "size": 2413
      },
      "createdDateTime": "2025-05-24T13:37:12Z"
    },
    {
      "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/transcriptions/5cff1d03-118f-4c4c-b3ba-e1f1cd88c14d/files/5baff707-8d68-4c69-850e-48775c57c982?api-version=2024-11-15",
      "name": "report.json",
      "kind": "TranscriptionReport",
      "links": {
        "contentUrl": "YourTranscriptionReportUrl"
      },
      "properties": {
        "size": 747
      },
      "createdDateTime": "2025-05-24T13:37:12Z"
    }
  ]
}
```

```
        }
    ]
}
```

The location of each transcription and transcription report files with more details are returned in the response body. The `contentUrl` property contains the URL to the [transcription](#) (`"kind": "Transcription"`) or [transcription report](#) (`"kind": "TranscriptionReport"`) file.

If you didn't specify a container in the `destinationContainerUrl` property of the transcription request, the results are stored in a container managed by Microsoft. When the transcription job is deleted, the transcription result data is also deleted.

Transcription report file

One transcription report file is provided for each submitted batch transcription job. The transcription report file is identified by the `"kind": "TranscriptionReport"` property in the response body of the [Transcriptions - List Files](#) operation.

The contents of each transcription result file are formatted as JSON, as shown in this example.

JSON

```
{
  "successfulTranscriptionsCount": 2,
  "failedTranscriptionsCount": 0,
  "details": [
    {
      "source": "https://crbn.us/hello.wav",
      "status": "Succeeded"
    },
    {
      "source": "https://crbn.us/whatstheweatherlike.wav",
      "status": "Succeeded"
    }
  ]
}
```

Transcription result file

One transcription result file is provided for each successfully transcribed audio file. The transcription result file is identified by the `"kind": "Transcription"` property in the response body of the [Transcriptions - List Files](#) operation.

The contents of each transcription result file are formatted as JSON, as shown in this example.

JSON

```
{  
    "source": "...",  
    "timestamp": "2025-05-24T13:37:05Z",  
    "durationInTicks": 25800000,  
    "durationMilliseconds": 2580,  
    "duration": "PT2.58S",  
    "combinedRecognizedPhrases": [  
        {  
            "channel": 0,  
            "lexical": "hello world",  
            "itn": "hello world",  
            "maskedITN": "hello world",  
            "display": "Hello world."  
        }  
    ],  
    "recognizedPhrases": [  
        {  
            "recognitionStatus": "Success",  
            "channel": 0,  
            "offset": "PT0.76S",  
            "duration": "PT1.32S",  
            "offsetInTicks": 7600000.0,  
            "durationInTicks": 13200000.0,  
            "nBest": [  
                {  
                    "confidence": 0.5643338,  
                    "lexical": "hello world",  
                    "itn": "hello world",  
                    "maskedITN": "hello world",  
                    "display": "Hello world.",  
                    "displayWords": [  
                        {  
                            "displayText": "Hello",  
                            "offset": "PT0.76S",  
                            "duration": "PT0.76S",  
                            "offsetInTicks": 7600000.0,  
                            "durationInTicks": 7600000.0  
                        },  
                        {  
                            "displayText": "world.",  
                            "offset": "PT1.52S",  
                            "duration": "PT0.56S",  
                            "offsetInTicks": 15200000.0,  
                            "durationInTicks": 5600000.0  
                        }  
                    ]  
                },  
                {  
                    "confidence": 0.1769063,  
                    "lexical": "helloworld",  
                    "itn": "helloworld",  
                    "maskedITN": "helloworld",  
                    "display": "Hello world."  
                }  
            ]  
        }  
    ]  
}
```

```

        "display": "helloworld"
    },
    {
        "confidence": 0.49964225,
        "lexical": "hello worlds",
        "itn": "hello worlds",
        "maskedITN": "hello worlds",
        "display": "hello worlds"
    },
    {
        "confidence": 0.4995761,
        "lexical": "hello worm",
        "itn": "hello worm",
        "maskedITN": "hello worm",
        "display": "hello worm"
    },
    {
        "confidence": 0.49418187,
        "lexical": "hello word",
        "itn": "hello word",
        "maskedITN": "hello word",
        "display": "hello word"
    }
]
}
]
}

```

Depending in part on the request parameters set when you created the transcription job, the transcription file can contain the following result properties.

[Expand table](#)

Property	Description
<code>channel</code>	The channel number of the results. For stereo audio streams, the left and right channels are split during the transcription. A JSON result file is created for each input audio file.
<code>combinedRecognizedPhrases</code>	The concatenated results of all phrases for the channel.
<code>confidence</code>	The confidence value for the recognition.
<code>display</code>	The display form of the recognized text. Added punctuation and capitalization are included.
<code>displayWords</code>	The timestamps for each word of the transcription. The <code>displayFormWordLevelTimestampsEnabled</code> request property must be set to <code>true</code> . Otherwise this property isn't present.
<code>duration</code>	The audio duration. The value is an ISO 8601 encoded duration.

Property	Description
durationInTicks	The audio duration in ticks (one tick is 100 nanoseconds).
durationMilliseconds	The audio duration in milliseconds.
itn	The inverse text normalized (ITN) form of the recognized text. Abbreviations such as "Doctor Smith" to "Dr Smith", phone numbers, and other transformations are applied.
lexical	The actual words recognized.
locale	The locale identified from the input the audio. The <code>languageIdentification</code> request property must be set. Otherwise this property isn't present.
maskedITN	The ITN form with profanity masking applied.
nBest	A list of possible transcriptions for the current phrase with confidences.
offset	The offset in audio of this phrase. The value is an ISO 8601 encoded duration.
offsetInTicks	The offset in audio of this phrase in ticks (one tick is 100 nanoseconds).
recognitionStatus	The recognition state. For example: "Success" or "Failure".
recognizedPhrases	The list of results for each phrase.
source	The URL that was provided as the input audio source. The source corresponds to the <code>contentUrls</code> or <code>contentContainerUrl</code> request property. The <code>source</code> property is the only way to confirm the audio input for a transcription.
speaker	The identified speaker. The <code>diarization</code> and <code>diarizationEnabled</code> request properties must be set. Otherwise this property isn't present.
timestamp	The creation date and time of the transcription. The value is an ISO 8601 encoded timestamp.
words	A list of results with lexical text for each word of the phrase. The <code>wordLevelTimestampsEnabled</code> request property must be set to <code>true</code> . Otherwise this property isn't present.

Related content

- [Learn more about batch transcription](#)
- [Locate audio files for batch transcription](#)
- [Create a batch transcription](#)
- [See batch transcription code samples at GitHub ↗](#)

Power automate batch transcription

09/16/2025

This article describes how to use [Power Automate](#) and the [Azure AI services for Batch Speech to text connector](#) to transcribe audio files from an Azure Storage container. The connector uses the [Batch Transcription REST API](#), but you don't need to write any code to use it. If the connector doesn't meet your requirements, you can still use the [REST API](#) directly.

In addition to [Power Automate](#), you can use the [Azure AI services for Batch Speech to text connector](#) with [Power Apps](#) and [Logic Apps](#).

Tip

Try more Speech features in [Speech Studio](#) without signing up or writing any code.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#).
- ✓ [Create an AI Foundry resource for Speech](#) 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.

Create the Azure Blob Storage container

In this example, you transcribe audio files that are located in an [Azure Blob Storage](#) account.

Follow these steps to create a new storage account and container.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. [Create a Storage account resource](#) in the Azure portal. Use the same subscription and resource group as your Speech resource.
3. Select the Storage account.
4. In the **Data storage** group in the left pane, select **Containers**.
5. Select **+ Container**.
6. Enter a name for the new container such as "batchtranscription" and select **Create**.
7. Select **Access keys** in the **Security + networking** group in the left pane. View and take note of the **key1** (or **key2**) value. You need the access key later when you [configure the connector](#).

Later you'll [upload files to the container](#) after the connector is configured, since the events of adding and modifying files kick off the transcription process.

Create a Power Automate flow

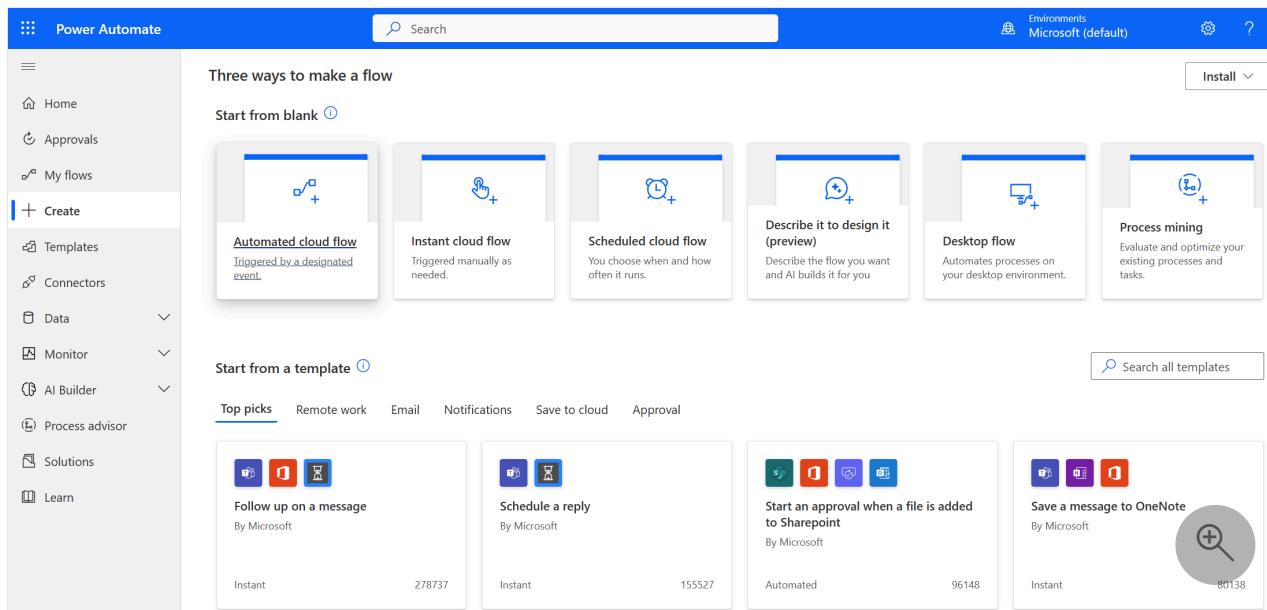
The steps to create your power automate flow are:

1. [Create a new flow](#)
2. [Configure the flow trigger](#)
3. [Create SAS URI by path](#)
4. [Create transcription](#)
5. [Test the flow](#)

Create a new flow

To create a new flow, follow these steps:

1. [Sign in to power automate](#)
2. From the collapsible menu on the left, select **Create**.
3. Select **Automated cloud flow** to start from a blank flow that can be triggered by a designated event.

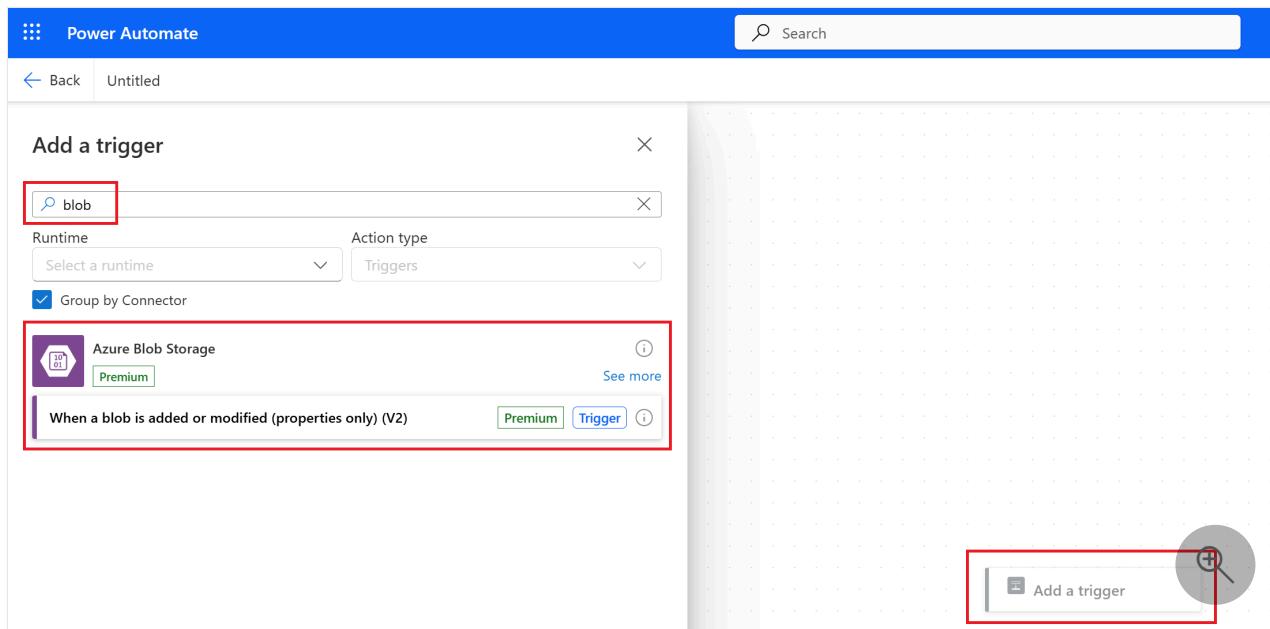


4. In the **Build an automated cloud flow** dialog, enter a name for your flow such as "BatchSTT".
5. Select **Skip** to exit the dialog and continue without choosing a trigger.

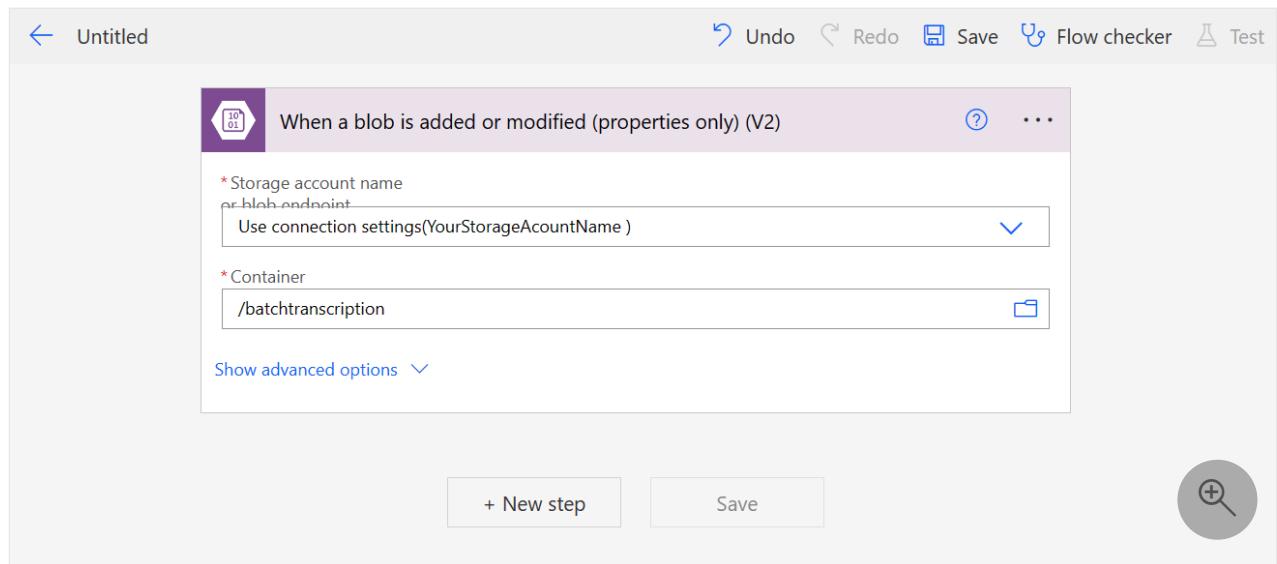
Configure the flow trigger

To configure the flow trigger, follow these steps:

1. Select **Add a trigger** to configure the event that starts the flow.
2. Choose a trigger from the [Azure Blob Storage connector](#). For this example, enter "blob" in the search connectors and triggers box to narrow the results.
3. Under the **Azure Blob Storage** connector, select the **When a blob is added or modified** trigger.



4. Configure the Azure Blob Storage connection.
 - a. From the **Authentication type** drop-down list, select **Access Key**.
 - b. Enter the account name and access key of the Azure Storage account that you [created previously](#).
 - c. Select **Create new** to continue.
5. Configure the **When a blob is added or modified** trigger.



- a. From the **Storage account name or blob endpoint** drop-down list, select **Use connection settings**. You should see the storage account name as a component of the connection string.
- b. Under **Container** select the folder icon. Choose the container that you [created previously](#).

Create SAS URI by path

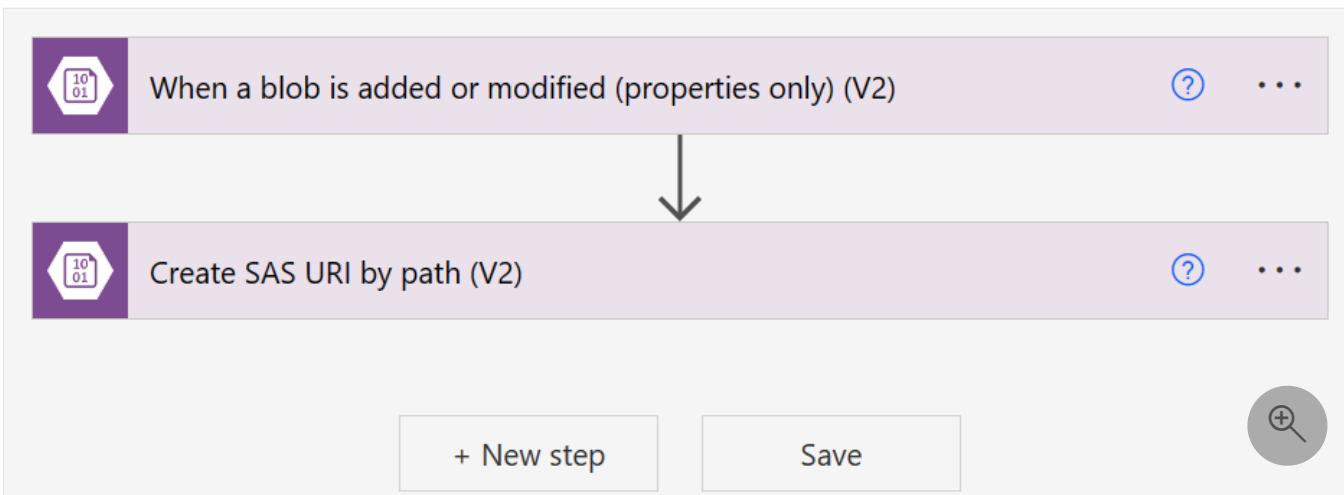
To transcribe an audio file that's in your [Azure Blob Storage container](#), you need a [Shared Access Signature \(SAS\) URI](#) for the file.

The [Azure Blob Storage connector](#) supports SAS URIs for individual blobs, but not for entire containers.

To create a SAS URI for a blob, follow these steps:

1. Select **+ New step** to begin adding a new operation for the Azure Blob Storage connector.
2. Enter "blob" in the search connectors and actions box to narrow the results.
3. Under the **Azure Blob Storage** connector, select the **Create SAS URI by path** trigger.
4. Under the **Storage account name or blob endpoint** drop-down, choose the same connection that you used for the **When a blob is added or modified** trigger.
5. Select **Path** as dynamic content for the **Blob path** field.

By now, you should have a flow that looks like this:

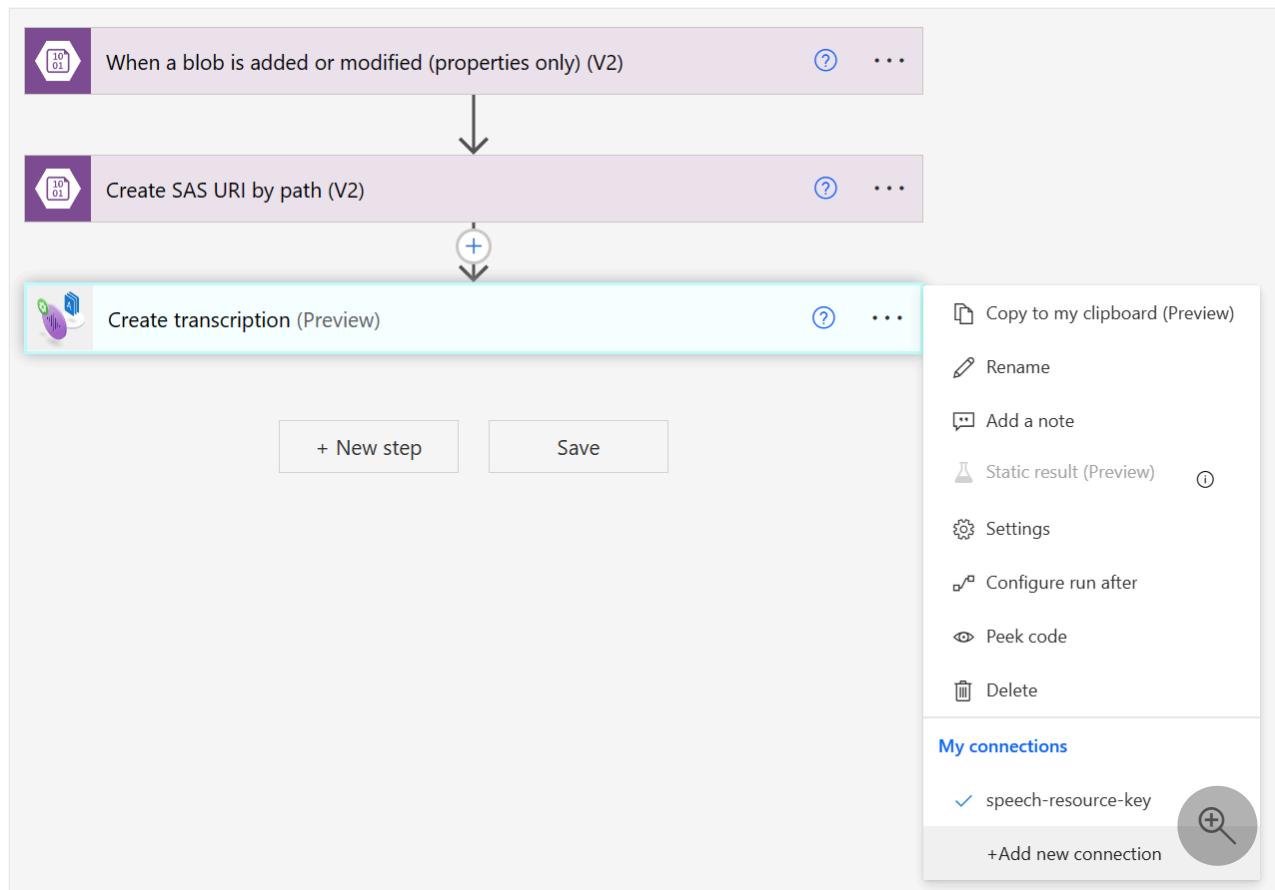


Create transcription

To create a transcription, follow these steps:

1. Select **+ New step** to begin adding a new operation for the [batch speech to text connector](#).
2. Enter "batch speech to text" in the search connectors and actions box to narrow the results.
3. Select the **Azure AI services for Batch Speech to text** connector.
4. Select the **Create transcription** action.
5. Create a new connection to the Speech resource that you [created previously](#). The connection is available throughout the Power Automate environment. For more information, see [Manage connections in Power Automate](#).
 - a. Enter a name for the connection such as "speech-resource-key". You can choose any name that you like.
 - b. In the **API Key** field, enter the Speech resource key.

Optionally you can select the connector ellipses (...) to view available connections. If you weren't prompted to create a connection, then you already have a connection that's selected by default.



6. Configure the **Create transcription** action.

- In the locale field, enter the expected locale of the audio data to transcribe.
- Select **DisplayName** as dynamic content for the **displayName** field. You can choose any name that you would like to refer to later.
- Select **Web Url** as dynamic content for the **contentUrls Item - 1** field. This is the SAS URI output from the **Create SAS URI by path** action.

💡 Tip

For more information about create transcription parameters, see the [Azure AI services for Batch Speech to text](#) documentation.

7. From the top navigation menu, select **Save**.

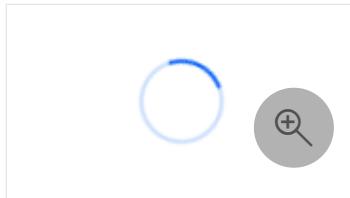
Test the flow

To test the flow, follow these steps:

- From the top navigation menu, select **Flow checker**. In the side panel that appears, you shouldn't see any errors or warnings. If you do, then you should fix them before continuing.

2. From the top navigation menu, save the flow and select **Test the flow**. In the window that appears, select **Test**.
3. In the side panel that appears, select **Manually** and then select **Test**.

After a few seconds, you should see an indication that the flow is in progress.



The flow is waiting for a file to be added or modified in the Azure Blob Storage container. That's the [trigger that you configured](#) earlier.

To trigger the test flow, upload an audio file to the Azure Blob Storage container as described next.

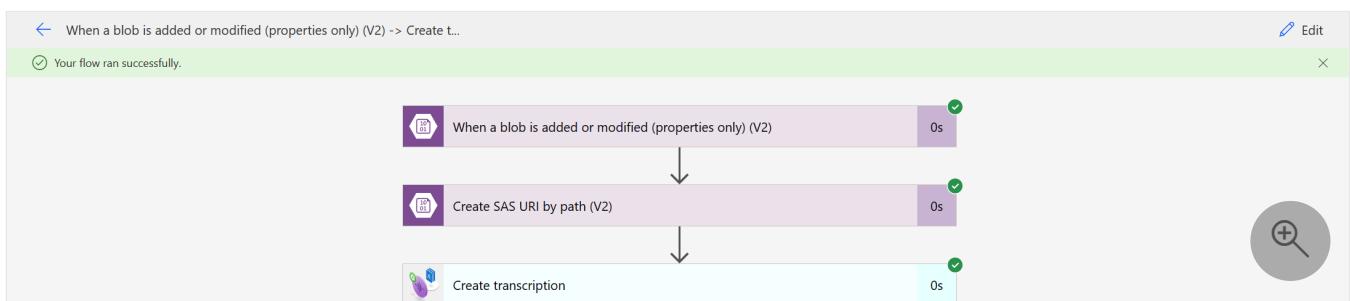
Upload files to the container

Follow these steps to upload [wav, mp3, or ogg](#) files from your local directory to the Azure Storage container that you [created previously](#).

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. [Create a Storage account resource](#) in the Azure portal. Use the same subscription and resource group as your Speech resource.
3. Select the Storage account.
4. Select the new container.
5. Select **Upload**.
6. Choose the files to upload and select **Upload**.

View the transcription flow results

After you upload the audio file to the Azure Blob Storage container, the flow should run and complete. Return to your test flow in the Power Automate portal to view the results.



You can select and expand the **Create transcription** to see detailed input and output results.

Next steps

- [Batch speech to text connector](#)
- [Azure Blob Storage connector](#)
- [Power Platform](#)

What is custom speech?

08/13/2025

With custom speech, you can evaluate and improve the accuracy of speech recognition for your applications and products. A custom speech model can be used for [real-time speech to text](#), [speech translation](#), and [batch transcription](#).

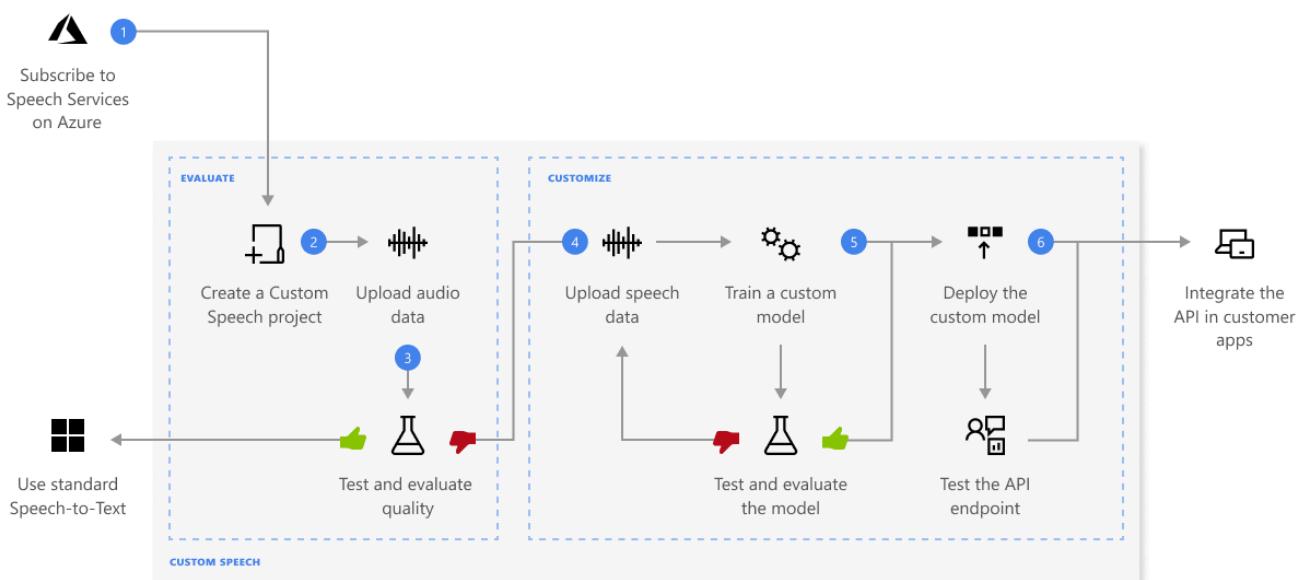
Out of the box, speech recognition utilizes a Universal Language Model as a base model that is trained with Microsoft-owned data and reflects commonly used spoken language. The base model is pre-trained with dialects and phonetics representing various common domains. When you make a speech recognition request, the most recent base model for each [supported language](#) is used by default. The base model works well in most speech recognition scenarios.

A custom model can be used to augment the base model to improve recognition of domain-specific vocabulary specific to the application by providing text data to train the model. It can also be used to improve recognition based for the specific audio conditions of the application by providing audio data with reference transcriptions.

You can also train a model with structured text when the data follows a pattern, to specify custom pronunciations, and to customize display text formatting with custom inverse text normalization, custom rewrite, and custom profanity filtering.

How does it work?

With custom speech, you can upload your own data, test and train a custom model, compare accuracy between models, and deploy a model to a custom endpoint.



Here's more information about the sequence of steps shown in the previous diagram:

1. [Create a project](#) and choose a model. If you train a custom model with audio data, select a service resource in a region with dedicated hardware for training audio data. For more information, see footnotes in the [regions](#) table.
2. [Upload test data](#). Upload test data to evaluate the speech to text offering for your applications, tools, and products.
3. [Train a model](#). Provide written transcripts and related text, along with the corresponding audio data. Testing a model before and after training is optional but recommended.

 **Note**

You pay for custom speech model usage and [endpoint hosting](#). You'll also be charged for custom speech model training if the base model was created on October 1, 2023 and later. You're not charged for training if the base model was created prior to October 2023. For more information, see [Azure AI Speech pricing](#) and the [Charge for adaptation section in the speech to text 3.2 migration guide](#).

4. [Test recognition quality](#). Use the [Speech Studio](#) to play back uploaded audio and inspect the speech recognition quality of your test data.
5. [Test model quantitatively](#). Evaluate and improve the accuracy of the speech to text model. The Speech service provides a quantitative word error rate (WER), which you can use to determine if more training is required.
6. [Deploy a model](#). Once you're satisfied with the test results, deploy the model to a custom endpoint. Except for [batch transcription](#), you must deploy a custom endpoint to use a custom speech model.

 **Tip**

A hosted deployment endpoint isn't required to use custom speech with the [Batch transcription API](#). You can conserve resources if the custom speech model is only used for batch transcription. For more information, see [Speech service pricing](#).

Choose your model

There are a few approaches to using custom speech models:

- The base model provides accurate speech recognition out of the box for a range of [scenarios](#). Base models are updated periodically to improve accuracy and quality. We

recommend that if you use base models, use the latest default base models. If a required customization capability is only available with an older model, then you can choose an older base model.

- A custom model augments the base model to include domain-specific vocabulary shared across all areas of the custom domain.
- Multiple custom models can be used when the custom domain has multiple areas, each with a specific vocabulary.

One recommended way to see if the base model suffices is to analyze the transcription produced from the base model and compare it with a human-generated transcript for the same audio. You can compare the transcripts and obtain a [word error rate \(WER\)](#) score. If the WER score is high, training a custom model to recognize the incorrectly identified words is recommended.

Multiple models are recommended if the vocabulary varies across the domain areas. For instance, Olympic commentators report on various events, each associated with its own vernacular. Because each Olympic event vocabulary differs significantly from others, building a custom model specific to an event increases accuracy by limiting the utterance data relative to that particular event. As a result, the model doesn't need to sift through unrelated data to make a match. Regardless, training still requires a decent variety of training data. Include audio from various commentators who have different accents, gender, age, etcetera.

Model stability and lifecycle

A base model or custom model deployed to an endpoint using custom speech is fixed until you decide to update it. The speech recognition accuracy and quality remain consistent, even when a new base model is released. This allows you to lock in the behavior of a specific model until you decide to use a newer model.

Whether you train your own model or use a snapshot of a base model, you can use the model for a limited time. For more information, see [Model and endpoint lifecycle](#).

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who are affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)
- [Integration and responsible use](#)

- Data, privacy, and security

Next steps

- Create a project
- Upload test data
- Train a model

Customize speech models with fine-tuning

07/31/2025

With custom speech, you can enhance speech recognition accuracy for your applications by using a custom model for real-time speech to text, speech translation, and batch transcription.

💡 Tip

Bring your custom speech models from [Speech Studio](#) to the [Azure AI Foundry portal](#). In Azure AI Foundry portal, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

You create a custom speech model by fine-tuning an Azure AI Speech base model with your own data. You can upload your data, test and train a custom model, compare accuracy between models, and deploy a model to a custom endpoint.

This article shows you how to use fine-tuning to create a custom speech model. For more information about custom speech, see the [custom speech overview](#) documentation.

💡 Tip

You can bring your custom speech models from [Speech Studio](#) to the [Azure AI Foundry portal](#). In Azure AI Foundry, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

Start fine-tuning

Custom speech fine-tuning includes models, training and testing datasets, and deployment endpoints. Each project is specific to a [locale](#). For example, you might fine-tune for English in the United States.

In the [Azure AI Foundry portal](#), you can fine-tune some Azure AI services models. For example, you can fine-tune a model for custom speech. Each custom model is specific to a [locale](#). For example, you might fine-tune a model for English in the United States.

1. Go to your project in the [Azure AI Foundry portal](#). If you need to create a project, see [Create an Azure AI Foundry project](#).

2. Select **Fine-tuning** from the left pane.

3. Select **AI Service fine-tuning > + Fine-tune**.

The screenshot shows the Azure AI Foundry interface. The top navigation bar includes the Azure AI Foundry logo, project name 'contoso-project', 'Fine-tuning' selected in the breadcrumb, 'All resources', and account information 'contoso-project-resource (westeurope, S0) EU'. The left sidebar has a tree view with 'Overview', 'Model catalog', 'Playgrounds', 'Build and customize' (expanded), 'Agents', 'Templates' (expanded), 'Fine-tuning' (selected and highlighted with a red box), 'Assess and improve' (expanded), 'Tracing PREVIEW', 'Monitoring', 'Protect and govern' (expanded), 'Introduction', 'Evaluation PREVIEW', and 'Guardrails + controls'. The main content area is titled 'Fine-tune with your own data' with a sub-section 'Generative AI fine-tuning' and 'AI Service fine-tuning' (selected and highlighted with a red box). Below are buttons for '+ Fine-tune' (highlighted with a red box), 'Refresh', and 'Reset view'. To the right are 'Filter' and 'Columns' buttons. A central image shows two stacked white cubes with a colorful gradient in the middle. Below it is a section titled 'Create AI Service fine-tuning' with the sub-instruction 'Fine-tune Microsoft's task specific models for your AI solutions.' and a magnifying glass icon.

4. In the wizard, select **Custom Speech (speech to text fine-tuning)** for custom speech.

Then select **Next**.

5. Enter the language, name, and description for the fine-tuning job. Then select **Create**.

Continue fine-tuning

Go to the Azure AI Speech documentation to learn how to continue fine-tuning your custom speech model:

- [Upload training and testing datasets](#)
- [Train a model](#)
- [Test model quantitatively and **test model qualitatively**](#)
- [Deploy a model](#)

View fine-tuned models

After fine-tuning, you can access your custom speech models and deployments from the **Fine-tuning** page.

1. Sign in to the [Azure AI Foundry portal](#).

2. Select **Fine-tuning** from the left pane.

3. Select AI Service fine-tuning.

Model name	Base model	Task	Status	Deployment
Contoso custom speech model	20241001	Speech to text	Succeeded	1

Get the project ID for the REST API

When you use the speech to text REST API for custom speech, you need to set the `project` property to the ID of your custom speech project. You need to set the `project` property so that you can manage fine-tuning in the [Azure AI Foundry portal](#).

Important

The project ID for custom speech isn't the same as the ID of the Azure AI Foundry project.

You can find the project ID in the URL after you select or start fine-tuning a custom speech model.

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane.
3. Select **AI Service fine-tuning**.
4. Select the custom model that you want to check from the **Model name** column.
5. Inspect the URL in your browser. The project ID is part of the URL. For example, the project ID is `00001111-aaaa-2222-bbbb-3333cccc4444` in the following URL:

`https`

https://ai.azure.com/build/models/aiservices/speech/customspeech/00001111-aaaa-2222-bbbb-3333cccc4444/<REDACTED_FOR_BREVITY>

Related content

- [Training and testing datasets](#)
- [Train a model](#)
- [Test model quantitatively](#)

Upload training and testing datasets for custom speech

07/31/2025

You need audio or text data for testing the accuracy of speech recognition or training your custom models. For information about the data types supported for testing or training your model, see [Training and testing datasets](#).

💡 Tip

You can also use the [online transcription editor](#) to create and refine labeled audio datasets.

Upload datasets

Follow these steps to upload datasets for training (fine-tuning) your custom speech model.

ⓘ Important

Repeat the steps to upload testing datasets (such as **Audio** only) that you need [later when you create a test](#). You can upload multiple datasets for training and testing.

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom speech fine-tuning task (by model name) that you [started as described in the how to start custom speech fine-tuning article](#).
4. Select **Manage data > Add dataset**.

Add data for training and testing

Add your dataset

Upload prepared training and/or test datasets for your use case.

+ Add dataset

5. In the **Add data** wizard, select the type of training data you want to add. In this example, we select **Audio + human-labeled transcript**. Then select **Next**.

Add data

Choose data type

Plain text

Structured text

Audio + human-labeled transcript

Training Testing

Training Testing

Training Testing

Learn more about preparing training and testing data

Download sample datasets

Next

Upload Cancel

6. On the **Upload your data** page, select local files, Azure Blob Storage, or other shared web locations. Then select **Next**.

If you select a remote location and you don't use trusted Azure services security mechanism, then the remote location should be a URL that can be retrieved with a simple anonymous GET request. For example, a [SAS URL](#) or a publicly accessible URL. URLs that require extra authorization or expect user interaction aren't supported.

Note

If you use Azure Blob URL, you can ensure maximum security of your dataset files by using trusted Azure services security mechanism. You use the same techniques as for batch transcription and plain Storage Account URLs for your dataset files. See details [here](#).

7. Enter a name and description for the data. Then select **Next**.
8. Review the data and select **Upload**. You're taken back to the **Manage data** page. The status of the data is **Processing**.

The screenshot shows the 'Manage data' page for a 'Contoso custom speech model'. On the left, there's a sidebar with links like 'Getting started', 'Manage data' (which is selected), 'Train model', 'Test models', and 'Deploy models'. The main area has a title 'Upload speech data for testing or training a speech recognition model' and a button 'Upload data'. Below that is a table with columns: Name, Description, Type, Quantity, Created on, and Status. A single row is shown: 'audio-and-transcription-dataset', 'Audio + transcript', '--', '5/18/2025, 12:44 PM', and 'Processing'. A red box highlights the 'Processing' status cell. There's also a magnifying glass icon at the bottom right of the table.

Name	Description	Type	Quantity	Created on	Status
audio-and-transcription-dataset		Audio + transcript	--	5/18/2025, 12:44 PM	Processing

9. Repeat the steps to upload testing datasets (such as **Audio** only) that you need [later when you create a test](#). You can upload multiple datasets for training and testing.
10. Repeat the previous steps to upload audio data [that you use later for testing](#). In the **Add data** wizard, select **Audio** for the type of data you want to add.

ⓘ Important

Connecting a dataset to a custom speech project isn't required to train and test a custom model using the REST API or Speech CLI. But if the dataset isn't connected to any project, you can't select it for training or testing in the [Azure AI Foundry portal](#).

Next steps

- [Test recognition quality](#)
- [Test model quantitatively](#)
- [Train a custom model](#)

Train a custom speech model

07/31/2025

In this article, you learn how to train a custom model to improve recognition accuracy from the Microsoft base model. The speech recognition accuracy and quality of a custom speech model remains consistent, even when a new base model is released.

Note

You pay for custom speech model usage and [endpoint hosting](#). You'll also be charged for custom speech model training if the base model was created on October 1, 2023 and later. You are not charged for training if the base model was created prior to October 2023. For more information, see [Azure AI Speech pricing](#) and the [Charge for adaptation section in the speech to text 3.2 migration guide](#).

Training a model is typically an iterative process. You first select a base model that is the starting point for a new model. You train a model with [datasets](#) that can include text and audio, and then you test. If the recognition quality or accuracy doesn't meet your requirements, you can create a new model with more or modified training data, and then test again.

You can use a custom model for a limited time after it was trained. You must periodically recreate and adapt your custom model from the latest base model to take advantage of the improved accuracy and quality. For more information, see [Model and endpoint lifecycle](#).

Important

If you train a custom model with audio data, select a service resource in a region with dedicated hardware for training audio data. After a model is trained, you can [copy it to an AI Foundry resource for Speech](#) in another region as needed.

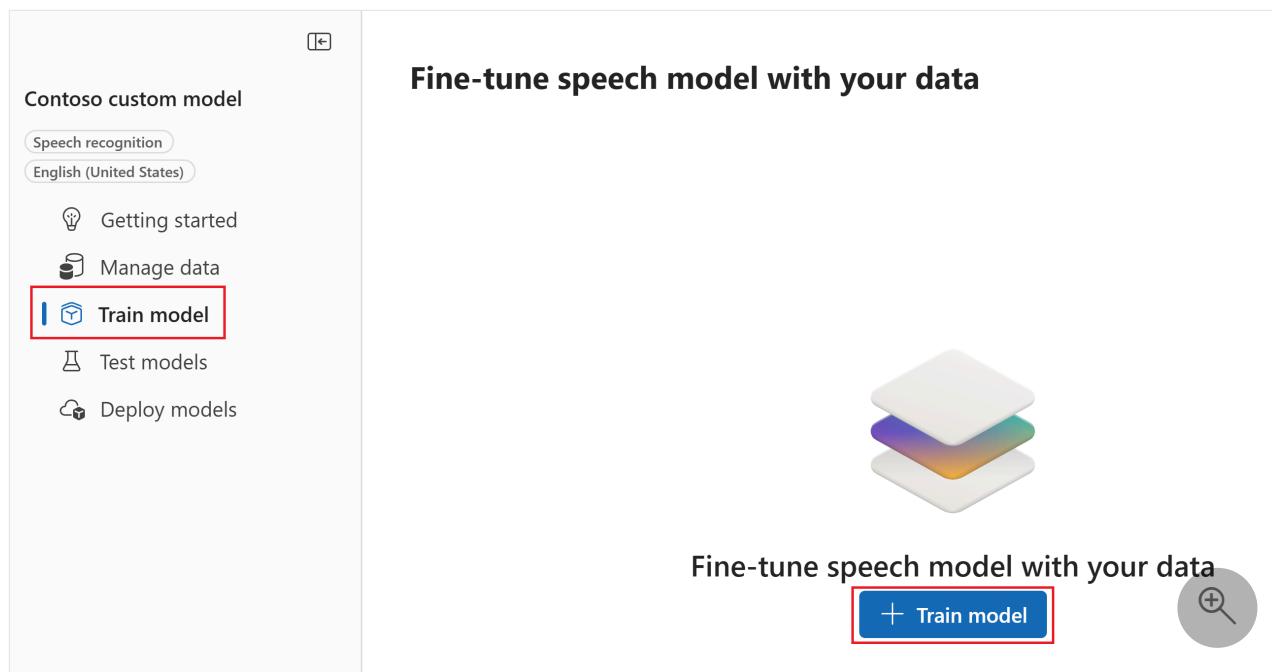
In regions with dedicated hardware for custom speech training, the Speech service will use up to 100 hours of your audio training data, and can process about 10 hours of data per day. See footnotes in the [regions](#) table for more information.

Create a model

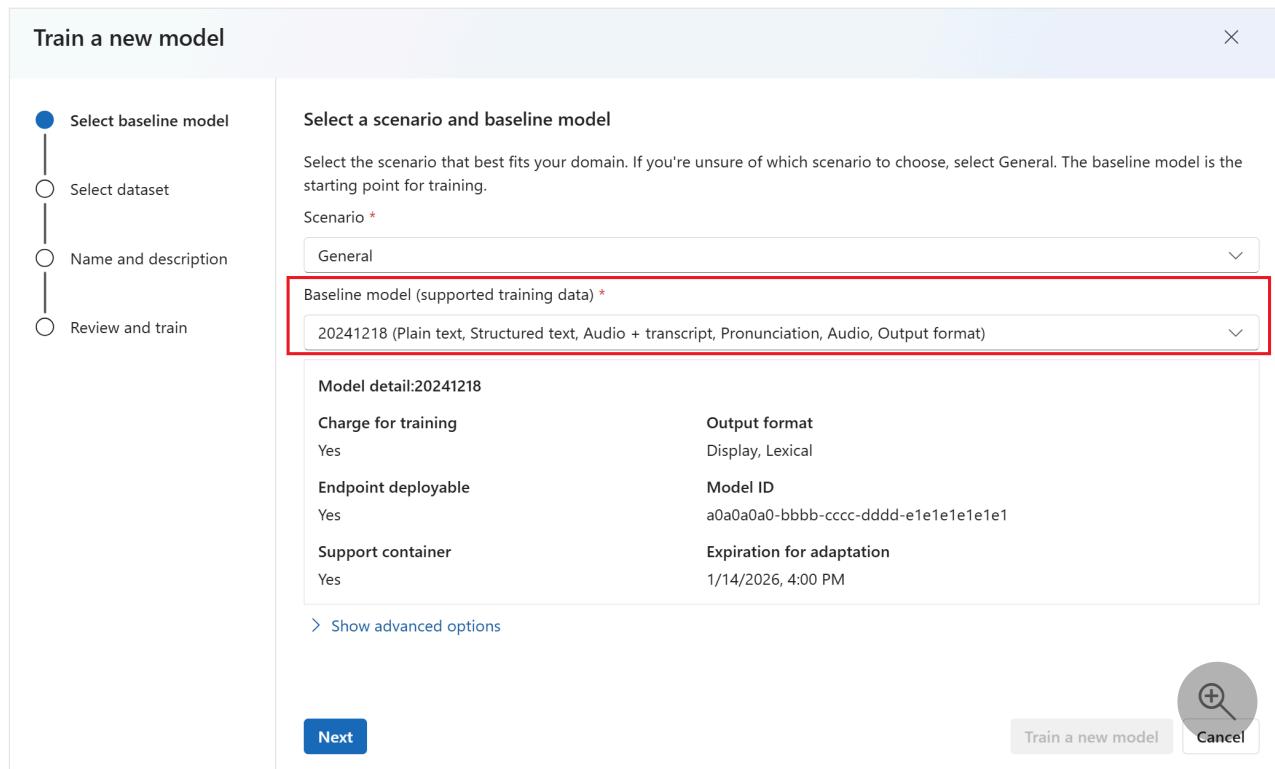
Tip

Bring your custom speech models from [Speech Studio](#) to the [Azure AI Foundry portal](#). In Azure AI Foundry portal, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom speech fine-tuning task (by model name) that you [started as described in the how to start custom speech fine-tuning article](#).
4. Select **Train model** > **+ Train model**.



5. In the **Train a new model** wizard, select the base model that you want to fine-tune. Then select **Next**.



6. Select the data that you want to use for training. Then select **Next**.
7. Enter a name and description for the model. Then select **Next**.
8. Review the settings and select **Train a new model**. You're taken back to the **Train model** page. The status of the data is **Processing**.

Name	Description	Baseline	Expiration	Created on	Endpoint depl...	Status
Contoso custom speech model	English (United States)	20241001	7/14/2027, 5:00 PM	5/18/2025, 12:51 PM	Yes	

Copy a model

You can copy a model to another project that uses the same locale. For example, after a model is trained with audio data in a [region](#) with dedicated hardware for training, you can copy it to an AI Foundry resource for Speech in another region as needed.

Connect a model

Models might have been copied from one project using the Speech CLI or REST API, without being connected to another project. Connecting a model is a matter of updating the model with a reference to the project.

Next steps

- Test recognition quality
- Test model quantitatively
- Deploy a model

Test recognition quality of a custom speech model

07/31/2025

You can inspect the recognition quality of a custom speech model. You can play back uploaded audio and determine if the provided recognition result is correct. After a test is successfully created, you can see how a model transcribed the audio dataset, or compare results from two models side by side.

Side-by-side model testing is useful to validate which speech recognition model is best for an application. For an objective measure of accuracy, which requires transcription datasets input, see [Test model quantitatively](#).

Important

When testing, the system will perform a transcription. This is important to keep in mind, as pricing varies per service offering and subscription level. Always refer to the official Azure AI services pricing for [the latest details](#).

Create a test

After you [upload training and testing datasets](#), you can create a test.

To test your fine-tuned custom speech model, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom speech fine-tuning task (by model name) that you [started as described in the how to start custom speech fine-tuning article](#).
4. Select **Test models > + Create test**.

The screenshot shows the 'Contoso custom speech' section of the Azure Custom Speech Service. On the left, a sidebar lists several options: 'Speech recognition' (selected), 'English (United States)', 'Getting started', 'Manage data', 'Train model', 'Test models' (highlighted with a red border), and 'Deploy models'. Below the sidebar is a large button labeled '+ Create test' with a magnifying glass icon. To the right of the button is a link 'Learn more about Speech test'. At the bottom left is a 'Back to fine-tuning' button. A circular search icon is located at the bottom right.

Model tests for inspecting recognition quality and accuracy

Contoso custom speech

Speech recognition
English (United States)

Getting started
Manage data
Train model
Test models
Deploy models

+ Create test

Learn more about Speech test

Back to fine-tuning

5. In the **Create a new test** wizard, select the test type. For a quality test, select **Inspect quality (Audio-only data)**. Then select **Next**.
6. Select the data that you want to use for testing. Then select **Next**.
7. Select up to two models to evaluate and compare accuracy. In this example, we select the model that we trained and the base model. Then select **Next**.

The screenshot shows the 'Create a new test' wizard on the 'Models to test' step. On the left, a vertical navigation bar shows steps: 'Test type' (checked), 'Test data' (checked), 'Models to test' (checked), 'Name and description' (unchecked), and 'Review and create' (unchecked). The main area is titled 'Choose models to test' with the sub-instruction 'Select up to two models to evaluate and compare accuracy.' A red box highlights the 'Model' field, which contains 'Contoso custom speech model'. Below it is a 'Model to compare (optional)' field containing '20241218'. At the bottom are 'Back' and 'Next' buttons, and a 'Create test' button with a magnifying glass icon.

Create a new test

Test type
Test data
Models to test
Name and description
Review and create

Choose models to test

Select up to two models to evaluate and compare accuracy.

Model *

Contoso custom speech model

Model to compare (optional)

20241218

Back Next

Create test

Cancel

8. Enter a name and description for the test. Then select **Next**.

9. Review the settings and select **Create test**. You're taken back to the **Test models** page.

The status of the data is **Processing**.

Get test results

You should get the test results and [inspect](#) the audio datasets compared to transcription results for each model.

When the test status is **Succeeded**, you can view the results. Select the test to view the results.

Compare transcription with audio

You can inspect the transcription output by each model tested, against the audio input dataset. If you included two models in the test, you can compare their transcription quality side by side.

Next steps

- [Test model quantitatively](#)
- [Train your model](#)
- [Deploy your model](#)

Test accuracy of a custom speech model

07/31/2025

In this article, you learn how to quantitatively measure and improve the accuracy of the base speech to text model or your own custom models. [Audio + human-labeled transcript](#) data is required to test accuracy. You should provide from 30 minutes to 5 hours of representative audio.

Important

When testing, the system will perform a transcription. This is important to keep in mind, as pricing varies per service offering and subscription level. Always refer to the official Azure AI services pricing for [the latest details](#).

Create a test

Tip

Bring your custom speech models from [Speech Studio](#) to the [Azure AI Foundry portal](#). In Azure AI Foundry portal, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

You can test the accuracy of your custom model by creating a test. A test requires a collection of audio files and their corresponding transcriptions. You can compare a custom model's accuracy with a speech to text base model or another custom model. After you [get](#) the test results, [evaluate the word error rate \(WER\)](#) compared to speech recognition results.

After you [upload training and testing datasets](#), you can create a test.

To test your fine-tuned custom speech model, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom speech fine-tuning task (by model name) that you [started as described in the how to start custom speech fine-tuning article](#).
4. Select **Test models > + Create test**.

Contoso custom speech

Speech recognition
English (United States)

- 💡 Getting started
- 📅 Manage data
- 🏡 Train model
- Test models**
- Deploy models



Test model performance

Quantitatively measure the accuracy of the base speech to text model or your own custom models with your use case data.

+ Create test

Learn more about Speech test [↗](#)

→ Back to fine-tuning

🔍

5. In the **Create a new test** wizard, select the test type. For an accuracy (quantitative) test, select **Evaluate accuracy (Audio + transcript data)**. Then select **Next**.

Create a new test

Choose a test type

You can inspect model quality on selected audio (**Inspect quality**) or evaluate and compare the accuracy of a selected model with another custom model or baseline model using selected data (**Evaluate accuracy**).

Inspect quality (Audio-only data)
Visually inspect the recognition quality of audio data using the model.

Evaluate accuracy (Audio + transcript data)
Accurately measure model performance by leveraging human-labeled transcripts from audio + transcript data.

Learn more about evaluating accuracy [↗](#)

Next

Create test

Cancel

🔍

6. Select the data that you want to use for testing. Then select **Next**.
7. Select up to two models to evaluate and compare accuracy. In this example, we select the model that we trained and the base model. Then select **Next**.

Create a new test

Test type
Test data
Models to test
Name and description
Review and create

Choose models to test
Select up to two models to evaluate and compare accuracy.

Model *
Contoso custom speech model

Model to compare (optional)
20241218

Back Next Create test Cancel

8. Enter a name and description for the test. Then select **Next**.

9. Review the settings and select **Create test**. You're taken back to the **Test models** page.
The status of the data is **Processing**.

Contoso custom speech model

Speech recognition
English (United States)

Name	Description	Type	Word error rate	Token error rate	Created on ↓	Status
Contoso custom speech evaluate accuracy		Evaluation	-- (Contoso custom speech model) -- (20241001)	-- (Contoso custom speech model) -- (20241001)	5/18/2025, 7:59 PM	Processing

Get test results

You should get the test results and [evaluate](#) the word error rate (WER) compared to speech recognition results.

When the test status is **Succeeded**, you can view the results. Select the test to view the results.

Evaluate word error rate (WER)

The industry standard for measuring model accuracy is [word error rate \(WER\)](#). WER counts the number of incorrect words identified during recognition, and divides the sum by the total number of words provided in the human-labeled transcript (N).

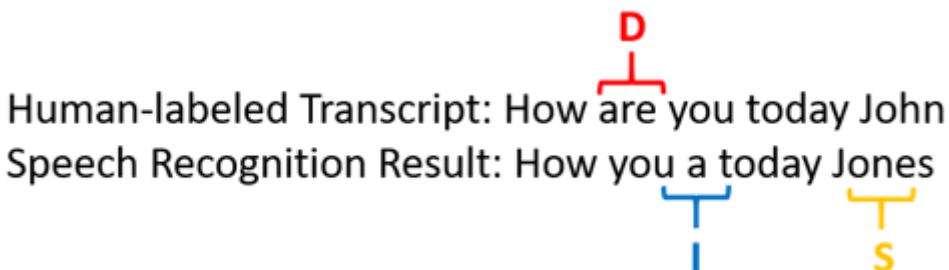
Incorrectly identified words fall into three categories:

- Insertion (I): Words that are incorrectly added in the hypothesis transcript
- Deletion (D): Words that are undetected in the hypothesis transcript
- Substitution (S): Words that were substituted between reference and hypothesis

In the [Azure AI Foundry portal](#) and [Speech Studio](#), the quotient is multiplied by 100 and shown as a percentage. The Speech CLI and REST API results aren't multiplied by 100.

$$WER = \frac{I + D + S}{N} \times 100$$

Here's an example that shows incorrectly identified words, when compared to the human-labeled transcript:



The speech recognition result erred as follows:

- Insertion (I): Added the word "a"
- Deletion (D): Deleted the word "are"
- Substitution (S): Substituted the word "Jones" for "John"

The word error rate from the previous example is 60%.

If you want to replicate WER measurements locally, you can use the sclite tool from the [NIST Scoring Toolkit \(SCTK\)](#).

Resolve errors and improve WER

You can use the WER calculation from the machine recognition results to evaluate the quality of the model you're using with your app, tool, or product. A WER of 5-10% is considered to be good quality and is ready to use. A WER of 20% is acceptable, but you might want to consider more training. A WER of 30% or more signals poor quality and requires customization and training.

How the errors are distributed is important. When many deletion errors are encountered, it's usually because of weak audio signal strength. To resolve this issue, you need to collect audio data closer to the source. Insertion errors mean that the audio was recorded in a noisy

environment and crosstalk might be present, causing recognition issues. Substitution errors are often encountered when an insufficient sample of domain-specific terms is provided as either human-labeled transcriptions or related text.

By analyzing individual files, you can determine what type of errors exist, and which errors are unique to a specific file. Understanding issues at the file level helps you target improvements.

Evaluate token error rate (TER)

Besides [word error rate](#), you can also use the extended measurement of **Token Error Rate (TER)** to evaluate quality on the final end-to-end display format. In addition to the lexical format (~~That will cost \$900.~~ instead of ~~that will cost nine hundred dollars~~), TER takes into account the display format aspects such as punctuation, capitalization, and ITN. Learn more about [Display output formatting with speech to text](#).

TER counts the number of incorrect tokens identified during recognition, and divides the sum by the total number of tokens provided in the human-labeled transcript (N).

$$TER = \frac{I + D + S}{N} \times 100$$

The formula of TER calculation is also similar to WER. The only difference is that TER is calculated based on the token level instead of word level.

- Insertion (I): Tokens that are incorrectly added in the hypothesis transcript
- Deletion (D): Tokens that are undetected in the hypothesis transcript
- Substitution (S): Tokens that were substituted between reference and hypothesis

In a real-world case, you can analyze both WER and TER results to get the desired improvements.

Note

To measure TER, you need to make sure the [audio + transcript testing data](#) includes transcripts with display formatting such as punctuation, capitalization, and ITN.

Example scenario outcomes

Speech recognition scenarios vary by audio quality and language (vocabulary and speaking style). The following table examines four common scenarios:

[Expand table](#)

Scenario	Audio quality	Vocabulary	Speaking style
Call center	Low, 8 kHz, could be two people on one audio channel, could be compressed	Narrow, unique to domain and products	Conversational, loosely structured
Voice assistant, such as Cortana, or a drive-through window	High, 16 kHz	Entity-heavy (song titles, products, locations)	Clearly stated words and phrases
Dictation (instant message, notes, search)	High, 16 kHz	Varied	Note-taking
Video closed captioning	Varied, including varied microphone use, added music	Varied, from meetings, recited speech, musical lyrics	Read, prepared, or loosely structured

Different scenarios produce different quality outcomes. The following table examines how content from these four scenarios rates in the [WER](#). The table shows which error types are most common in each scenario. The insertion, substitution, and deletion error rates help you determine what kind of data to add to improve the model.

[Expand table](#)

Scenario	Speech recognition quality	Insertion errors	Deletion errors	Substitution errors
Call center	Medium (< 30% WER)	Low, except when other people talk in the background	Can be high. Call centers can be noisy, and overlapping speakers can confuse the model	Medium. Products and people's names can cause these errors
Voice assistant	High (can be < 10% WER)	Low	Low	Medium, due to song titles, product names, or locations
Dictation	High (can be < 10% WER)	Low	Low	High
Video closed captioning	Depends on video type (can be < 50% WER)	Low	Can be high because of music, noises, microphone quality	Jargon might cause these errors

Next steps

- [Train a model](#)
- [Deploy a model](#)
- [Use the online transcription editor](#)

Deploy a custom speech model

07/31/2025

In this article, you learn how to deploy an endpoint for a custom speech model. Except for [batch transcription](#), you must deploy a custom endpoint to use a custom speech model.

💡 Tip

A hosted deployment endpoint isn't required to use custom speech with the [Batch transcription API](#). You can conserve resources if the [custom speech model](#) is only used for batch transcription. For more information, see [Speech service pricing](#).

You can deploy an endpoint for a base or custom model, and then [update](#) the endpoint later to use a better trained model.

❗ Note

Endpoints used by [F0](#) Speech resources are deleted after seven days.

Add a deployment endpoint

💡 Tip

Bring your custom speech models from [Speech Studio](#) to the [Azure AI Foundry portal](#). In Azure AI Foundry portal, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom speech fine-tuning task (by model name) that you [started as described in the how to start custom speech fine-tuning article](#).
4. Select **Deploy models > + Deploy models**.

Contoso custom model

Speech recognition
English (United States)

- Getting started
- Manage data
- Train model
- Test models
- Deploy models**

Manage fine-tuned model deployments for your solutions



Deploy your model

Deploy a custom endpoint to use a custom speech model, except for batch transcription.

+ Deploy model

Learn more about Speech test

5. In the **Deploy a new model** wizard, select the model that you want to deploy.

Deploy model

Model *

Contoso custom speech model

Expiration for decoding: 1/14/2027, 4:00 PM

Name *

Contoso custom speech deployment

Description

Describe your deployment (max 500 characters)

Content logging (audio and diagnostic information)

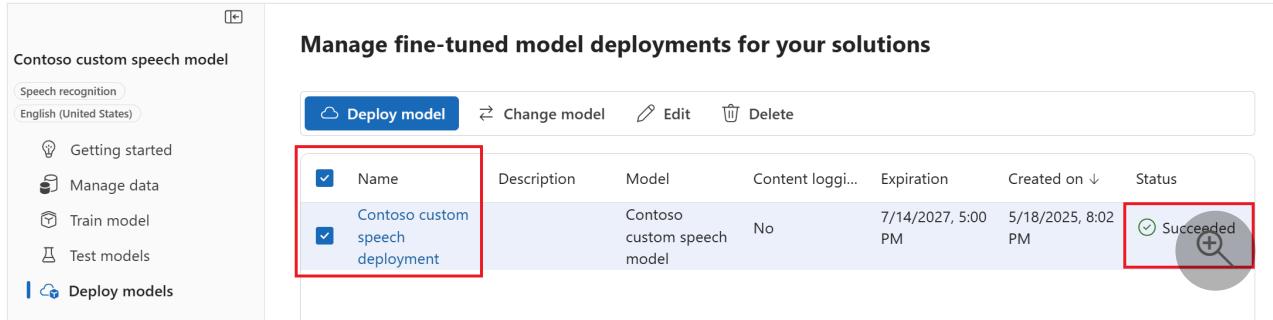
No

I accept the [terms of use](#) ↗ and I acknowledge that endpoint hosting will incur usage to my account. [See pricing](#) ↗ *

Deploy **Cancel**

6. Enter a name and description for the deployment. Select the box to agree to the terms of use. Then select **Deploy**.

7. After the deployment status is **Succeeded**, you can view the deployment details. Select the deployment to view the details like the endpoint ID.



The screenshot shows the Azure portal interface for managing model deployments. On the left, there's a sidebar for 'Contoso custom speech model' with options like 'Speech recognition', 'English (United States)', 'Getting started', 'Manage data', 'Train model', 'Test models', and 'Deploy models'. The main area is titled 'Manage fine-tuned model deployments for your solutions'. It has buttons for 'Deploy model', 'Change model', 'Edit', and 'Delete'. A table lists deployments with columns: Name, Description, Model, Content loggi..., Expiration, Created on, and Status. One row is selected, highlighted with a red box, showing 'Contoso custom speech deployment' as the name, 'Contoso custom speech model' as the model, and 'No' as the content logging status. The expiration date is 7/14/2027, 5:00 PM, and it was created on 5/18/2025, 8:02 PM. The status is 'Succeeded', indicated by a green checkmark icon in a grey circle with a magnifying glass. Another red box highlights the 'Succeeded' status icon.

Name	Description	Model	Content loggi...	Expiration	Created on	Status
Contoso custom speech deployment		Contoso custom speech model	No	7/14/2027, 5:00 PM	5/18/2025, 8:02 PM	 Succeeded

Change model and redeploy endpoint

An endpoint can be updated to use another model that was created by the same Speech resource. As previously mentioned, you must update the endpoint's model before the [model expires](#).

The redeployment takes several minutes to complete. In the meantime, your endpoint uses the previous model without interruption of service.

View logging data

Logging data is available for export if you configured it while creating the endpoint.

Logging data is available on Microsoft-owned storage for 30 days, and then it's removed. If your own storage account is linked to the Azure AI services subscription, the logging data isn't automatically deleted.

Related content

- [Custom speech overview](#)
- [Custom speech model lifecycle](#)

Training and testing datasets

05/19/2025

In a custom speech project, you can upload datasets for training, qualitative inspection, and quantitative measurement. This article covers the types of training and testing data that you can use for custom speech.

Text and audio that you use to test and train a custom model should include samples from a diverse set of speakers and scenarios that you want your model to recognize. Consider these factors when you're gathering data for custom model testing and training:

- Include text and audio data to cover the kinds of verbal statements that your users make when they're interacting with your model. For example, a model that raises and lowers the temperature needs training on statements that people might make to request such changes.
- Include all speech variances that you want your model to recognize. Many factors can vary speech, including accents, dialects, language-mixing, age, gender, voice pitch, stress level, and time of day.
- Include samples from different environments, for example, indoor, outdoor, and road noise, where your model is used.
- Record audio with hardware devices that the production system uses. If your model must identify speech recorded on devices of varying quality, the audio data that you provide to train your model must also represent these diverse scenarios.
- Keep the dataset diverse and representative of your project requirements. You can add more data to your model later.
- Only include data that your model needs to transcribe. Including data that isn't within your custom model's recognition requirements can harm recognition quality overall.

Data types

The following table lists accepted data types, when each data type should be used, and the recommended quantity. Not every data type is required to create a model. Data requirements vary depending on whether you're creating a test or training a model.

[] Expand table

Data type	Used for testing	Recommended for testing	Used for training	Recommended for training
Audio only	Yes (visual inspection)	5+ audio files	Yes (Preview for	1-100 hours of audio

Data type	Used for testing	Recommended for testing	Used for training	Recommended for training
(en-US)				
Audio + human-labeled transcripts	Yes (evaluation of accuracy)	0.5-5 hours of audio	Yes	1-100 hours of audio
Plain text	No	Not applicable	Yes	1-200 MB of related text
Structured text	No	Not applicable	Yes	Up to 10 classes with up to 4,000 items and up to 50,000 training sentences
Pronunciation	No	Not applicable	Yes	1 KB to 1 MB of pronunciation text
Display format	No	Not applicable	Yes	Up to 200 lines for ITN, 1,000 lines for rewrite, 1,000 lines for profanity filter

Training with plain text or structured text usually finishes within a few minutes.

Tip

Start with plain-text data or structured-text data. This data will improve the recognition of special terms and phrases. Training with text is much faster than training with audio (minutes versus days).

Start with small sets of sample data that match the language, acoustics, and hardware where your model will be used. Small datasets of representative data can expose problems before you invest in gathering larger datasets for training. For sample custom speech data, see [this GitHub repository](#).

If you train a custom model with audio data, select a service resource in a region with dedicated hardware for training audio data. For more information, see footnotes in the [regions](#) table. In regions with dedicated hardware for custom speech training, the Speech service uses up to 100 hours of your audio training data, and can process about 10 hours of data per day. After the model is trained, you can copy the model to another region as needed with the [Models_CopyTo](#) REST API.

Consider datasets by scenario

A model trained on a subset of scenarios can perform well in only those scenarios. Carefully choose data that represents the full scope of scenarios that you need your custom model to

recognize. The following table shows datasets to consider for some speech recognition scenarios:

[Expand table](#)

Scenario	Plain text data and structured text data	Audio + human-labeled transcripts	New words with pronunciation
Call center	Marketing documents, website, product reviews related to call center activity	Call center calls transcribed by humans	Terms that have ambiguous pronunciations (see the <i>Xbox</i> example in the preceding section)
Voice assistant	Lists of sentences that use various combinations of commands and entities	Recorded voices speaking commands into device, transcribed into text	Names (movies, songs, products) that have unique pronunciations
Dictation	Written input, such as instant messages or emails	Similar to preceding examples	Similar to preceding examples
Video closed captioning	TV show scripts, movies, marketing content, video summaries	Exact transcripts of videos	Similar to preceding examples

To help determine which dataset to use to address your problems, refer to the following table:

[Expand table](#)

Use case	Data type
Improve recognition accuracy on industry-specific vocabulary and grammar, such as medical terminology or IT jargon.	Plain text or structured text data
Define the phonetic and displayed form of a word or term that has nonstandard pronunciation, such as product names or acronyms.	Pronunciation data or phonetic pronunciation in structured text
Improve recognition accuracy on speaking styles, accents, or specific background noises.	Audio + human-labeled transcripts

Audio + human-labeled transcript data for training or testing

You can use audio + human-labeled transcript data for both [training](#) and [testing](#) purposes. You must provide human-labeled transcriptions (word by word) for comparison:

- To improve the acoustic aspects like slight accents, speaking styles, and background noises.
- To measure the accuracy of Microsoft's speech to text accuracy when, it's processing your audio files.

For a list of base models that support training with audio data, see [Language support](#). Even if a base model does support training with audio data, the service might use only part of the audio. And it still uses all the transcripts.

Important

If a base model doesn't support customization with audio data, only the transcription text will be used for training. If you switch to a base model that supports customization with audio data, the training time may increase from several hours to several days. The change in training time would be most noticeable when you switch to a base model in a [region](#) without dedicated hardware for training. If the audio data is not required, you should remove it to decrease the training time.

Audio with human-labeled transcripts offers the greatest accuracy improvements if the audio comes from the target use case. Samples must cover the full scope of speech. For example, a call center for a retail store would get the most calls about swimwear and sunglasses during summer months. Ensure that your sample includes the full scope of speech that you want to detect.

Consider these details:

- Training with audio brings the most benefits if the audio is also hard to understand for humans. In most cases, you should start training by using only related text.
- If you use one of the most heavily used languages, such as US English, it's unlikely that you would need to train with audio data. For such languages, the base models already offer good recognition results in most scenarios, so it's probably enough to train with related text.
- Custom speech can capture word context only to reduce substitution errors, not insertion or deletion errors.
- Avoid samples that include transcription errors, but do include a diversity of audio quality.
- Avoid sentences that are unrelated to your problem domain. Unrelated sentences can harm your model.
- When the transcript quality varies, you can duplicate exceptionally good sentences, such as excellent transcriptions that include key phrases, to increase their weight.
- The Speech service automatically uses the transcripts to improve the recognition of domain-specific words and phrases, as though they were added as related text.

- It can take several days for a training operation to finish. To improve the speed of training, be sure to create your Speech service subscription in a region with dedicated hardware for training.

A large training dataset is required to improve recognition. Generally, we recommend that you provide word-by-word transcriptions for 1 to 100 hours of audio (up to 20 hours for older models that do not charge for training). However, even as little as 30 minutes can help improve recognition results. Although creating human-labeled transcription can take time, improvements in recognition are only as good as the data that you provide. You should upload only high-quality transcripts.

Audio files can have silence at the beginning and end of the recording. If possible, include at least a half-second of silence before and after speech in each sample file. Although audio with low recording volume or disruptive background noise isn't helpful, it shouldn't limit or degrade your custom model. Always consider upgrading your microphones and signal processing hardware before gathering audio samples.

ⓘ Important

For more information about the best practices of preparing human-labeled transcripts, see [Human-labeled transcripts with audio](#).

Custom speech projects require audio files with these properties:

ⓘ Important

These are requirements for Audio + human-labeled transcript training and testing. They differ from the ones for Audio only training and testing. If you want to use Audio only training and testing, [see this section](#).

 Expand table

Property	Value
File format	RIFF (WAV)
Sample rate	8,000 Hz or 16,000 Hz
Channels	1 (mono)
Maximum length per audio	Two hours (testing) / 40 s (training) Training with audio has a maximum audio length of 40 seconds per file (up to 30

Property	Value
	seconds for Whisper customization). For audio files longer than 40 seconds, only the corresponding text from the transcription files is used for training. If all audio files are longer than 40 seconds, the training fails.
Sample format	PCM, 16-bit
Archive format	.zip
Maximum zip size	2 GB or 10,000 files

Plain-text data for training

You can add plain text sentences of related text to improve the recognition of domain-specific words and phrases. Related text sentences can reduce substitution errors related to misrecognition of common words and domain-specific words by showing them in context. Domain-specific words can be uncommon or made-up words, but their pronunciation must be straightforward to be recognized.

Provide domain-related sentences in a single text file. Use text data that's close to the expected spoken utterances. Utterances don't need to be complete or grammatically correct, but they must accurately reflect the spoken input that you expect the model to recognize. When possible, try to have one sentence or keyword controlled on a separate line. To increase the weight of a term such as product names, add several sentences that include the term. But don't copy too much - it could affect the overall recognition rate.

! Note

Avoid related text sentences that include noise such as unrecognizable characters or words.

Use this table to ensure that your plain text dataset file is formatted correctly:

[] [Expand table](#)

Property	Value
Text encoding	UTF-8 BOM
Number of utterances per line	1

Property	Value
Maximum file size	200 MB

You must also adhere to the following restrictions:

- Avoid repeating characters, words, or groups of words more than three times. For example, don't use "aaaa," "yeah yeah yeah yeah," or "that's it that's it that's it that's it." The Speech service might drop lines with too many repetitions.
- Don't use special characters or UTF-8 characters above `U+00A1`.
- URIs will be rejected.
- For some languages such as Japanese or Korean, importing large amounts of text data can take a long time or can time out. Consider dividing the dataset into multiple text files with up to 20,000 lines in each.

Structured-text data for training

ⓘ Note

Structured-text data for training is in public preview.

Use structured text data when your data follows a particular pattern in particular utterances that differ only by words or phrases from a list. To simplify the creation of training data and to enable better modeling inside the Custom Language model, you can use a structured text in Markdown format to define lists of items and phonetic pronunciation of words. You can then reference these lists inside your training utterances.

Expected utterances often follow a certain pattern. One common pattern is that utterances differ only by words or phrases from a list. Examples of this pattern could be:

- "I have a question about `product`," where `product` is a list of possible products.
- "Make that `object` `color`," where `object` is a list of geometric shapes and `color` is a list of colors.

For a list of supported base models and locales for training with structured text, see [Language support](#). You must use the latest base model for these locales. For locales that don't support training with structured text, the service will take any training sentences that don't reference any classes as part of training with plain-text data.

The structured-text file should have an .md extension. The maximum file size is 200 MB, and the text encoding must be UTF-8 BOM. The syntax of the Markdown is the same as that from the Language Understanding models, in particular list entities and example utterances. For

more information about the complete Markdown syntax, see the [Language Understanding Markdown](#).

Here are key details about the supported Markdown format:

 Expand table

Property	Description	Limits
@list	A list of items that can be referenced in an example sentence.	Maximum of 20 lists. Maximum of 35,000 items per list.
speech:phoneticlexicon	A list of phonetic pronunciations according to the Universal Phone Set . Pronunciation is adjusted for each instance where the word appears in a list or training sentence. For example, if you have a word that sounds like "cat" and you want to adjust the pronunciation to "k ae t", you would add - cat/k ae t to the speech:phoneticlexicon list.	Maximum of 15,000 entries. Maximum of two pronunciations per word.
#ExampleSentences	A pound symbol (#) delimits a section of example sentences. The section heading can only contain letters, digits, and underscores. Example sentences should reflect the range of speech that your model should expect. A training sentence can refer to items under a @list by using surrounding left and right curly braces ({@list name}). You can refer to multiple lists in the same training sentence, or none at all.	Maximum file size of 200 MB.
//	Comments follow a double slash (//).	Not applicable

Here's an example structured text file:

```
markdown

// This is a comment because it follows a double slash (^//^).

// Here are three separate lists of items that can be referenced in an example sentence. You can have up to 10 of these.

@ list food =
- pizza
- burger
- ice cream
- soda

@ list pet =
- cat
- dog
```

```

- fish

@ list sports =
- soccer
- tennis
- cricket
- basketball
- baseball
- football

// List of phonetic pronunciations
@ speech:phoneticlexicon
- cat/k ae t
- fish/f ih sh

// Here are two sections of training sentences.

#TrainingSentences_Section1
- you can include sentences without a class reference
- what {@pet} do you have
- I like eating {@food} and playing {@sports}
- my {@pet} likes {@food}

#TrainingSentences_Section2
- you can include more sentences without a class reference
- or more sentences that have a class reference like {@pet}

```

Pronunciation data for training

Specialized or made up words might have unique pronunciations. These words can be recognized if they can be broken down into smaller words to pronounce them. For example, to recognize "Xbox", pronounce it as "X box". This approach won't increase overall accuracy, but can improve recognition of this and other keywords.

You can provide a custom pronunciation file to improve recognition. Don't use custom pronunciation files to alter the pronunciation of common words. For a list of languages that support custom pronunciation, see [language support](#).

Note

You can use a pronunciation file alongside any other training dataset except structured text training data. To use pronunciation data with structured text, it must be within a structured text file.

The spoken form is the phonetic sequence spelled out. It can be composed of letters, words, syllables, or a combination of all three. This table includes some examples:

[Expand table](#)

Recognized displayed form	Spoken form
3CPO	three c p o
CNTK	c n t k
IEEE	i triple e

You provide pronunciations in a single text file. Include the spoken utterance and a custom pronunciation for each. Each row in the file should begin with the recognized form, then a tab character, and then the space-delimited phonetic sequence.

tsv
3CPO three c p o
CNTK c n t k
IEEE i triple e

Refer to the following table to ensure that your pronunciation dataset files are valid and correctly formatted.

[Expand table](#)

Property	Value
Text encoding	UTF-8 BOM (ANSI is also supported for English)
Number of pronunciations per line	1
Maximum file size	1 MB (1 KB for free tier)

Audio data for training or testing

Audio data is optimal for testing the accuracy of Microsoft's baseline speech to text model or a custom model. Keep in mind that audio data is used to inspect the accuracy of speech regarding a specific model's performance. If you want to quantify the accuracy of a model, use [audio + human-labeled transcripts](#).

Note

Audio only data for training is available in preview for the `en-US` locale. For other locales, to train with audio data you must also provide [human-labeled transcripts](#).

Custom speech projects require audio files with these properties:

ⓘ Important

These are requirements for Audio only training and testing. They differ from the ones for Audio + human-labeled transcript training and testing. If you want to use Audio + human-labeled transcript training and testing, [see this section](#).

 Expand table

Property	Value
File format	RIFF (WAV)
Sample rate	8,000 Hz or 16,000 Hz
Channels	1 (mono)
Maximum length per audio	Two hours
Sample format	PCM, 16-bit
Archive format	.zip
Maximum archive size	2 GB or 10,000 files

! Note

When you're uploading training and testing data, the .zip file size can't exceed 2 GB. If you require more data for training, divide it into several .zip files and upload them separately. Later, you can choose to train from *multiple* datasets. However, you can test from only a *single* dataset.

Use [SoX](#) to verify audio properties or convert existing audio to the appropriate formats. Here are some example SoX commands:

 Expand table

Activity	SoX command
Check the audio file format.	<code>sox --i <filename></code>
Convert the audio file to single channel, 16-bit, 16 KHz.	<code>sox <input> -b 16 -e signed-integer -c 1 -r 16k -t wav <output>.wav</code>

Custom display text formatting data for training

Learn more about [preparing display text formatting data](#) and [display text formatting with speech to text](#).

Automatic Speech Recognition output display format is critical to downstream tasks and one-size doesn't fit all. Adding Custom Display Format rules allows users to define their own lexical-to-display format rules to improve the speech recognition service quality on top of Microsoft Azure custom speech Service.

It allows you to fully customize the display outputs such as add rewrite rules to capitalize and reformulate certain words, add profanity words and mask from output, define advanced ITN rules for certain patterns such as numbers, dates, email addresses; or preserve some phrases and kept them from any Display processes.

For example:

[+] Expand table

Custom formatting	Display text
None	My financial number from contoso is 8BEV3
Capitalize "Contoso" (via <code>#rewrite</code> rule) Format financial number (via <code>#itn</code> rule)	My financial number from Contoso is 8B-EV-3

For a list of supported base models and locales for training with structured text, see Language support. The Display Format file should have an .md extension. The maximum file size is 10 MB, and the text encoding must be UTF-8 BOM. For more information about customizing Display Format rules, see [Display Formatting Rules Best Practice](#).

[+] Expand table

Property	Description	Limits
#ITN	A list of invert-text-normalization rules to define certain display patterns such as numbers, addresses, and dates.	Maximum of 200 lines
#rewrite	A list of rewrite pairs to replace certain words for reasons such as capitalization and spelling correction.	Maximum of 1,000 lines
#profanity	A list of unwanted words that will be masked as <code>*****</code> from Display and Masked output, on top of Microsoft built-in profanity lists.	Maximum of 1,000 lines
#test	A list of unit test cases to validate if the display rules work as expected, including the lexical format input and the expected display format	Maximum file size of 10 MB

Property	Description	Limits
	output.	

Here's an example display format file:

```
text

// this is a comment line
// each section must start with a '#' character
#itn
// list of ITN pattern rules, one rule for each line
\d-\d-\d
\d-\l-\l-\d
#rewrite
// list of rewrite rules, each rule has two phrases, separated by a tab character
old phrase    new phrase
#profanity
// list of profanity phrases to be tagged/removed/masked, one line one phrase
fakeprofanity
#test
// list of test cases, each test case has two sentences, input lexical and
expected display output
// the two sentences are separated by a tab character
// the expected sentence is the display output of DPP+CDPP models
Mask the fakeprofanity word Mask the ***** word
```

Next steps

- [Upload your data](#)
- [Test model quantitatively](#)
- [Train a custom model](#)

How to create human-labeled transcriptions

05/19/2025

Human-labeled transcriptions are word-by-word transcriptions of an audio file. You use human-labeled transcriptions to evaluate model accuracy and to improve recognition accuracy, especially when words are deleted or incorrectly replaced. This guide can help you create high-quality transcriptions.

A representative sample of transcription data is recommended to evaluate model accuracy. The data should cover various speakers and utterances that are representative of what users say to the application. For test data, the maximum duration of each individual audio file is 2 hours.

A large sample of transcription data is required to improve recognition. We suggest providing between 1 and 100 hours of audio data. The Speech service uses up to 100 hours of audio for training (up to 20 hours for older models that don't charge for training). Each individual audio file shouldn't be longer than 40 seconds (up to 30 seconds for Whisper customization).

This guide has sections for US English, Mandarin Chinese, and German locales.

The transcriptions for all WAV files are contained in a single plain-text file (.txt or .tsv). Each line of the transcription file contains the name of one of the audio files, followed by the corresponding transcription. The file name and transcription are separated by a tab (\t).

For example:

txt	
speech01.wav	speech recognition is awesome
speech02.wav	the quick brown fox jumped all over the place
speech03.wav	the lazy dog was not amused

The transcriptions are text-normalized so the system can process them. However, you must do some important normalizations before you upload the dataset.

Human-labeled transcriptions for languages other than English and Mandarin Chinese, must be UTF-8 encoded with a byte-order marker. For other locales transcription requirements, see the following sections.

en-US

Human-labeled transcriptions for English audio must be provided as plain text, only using ASCII characters. Avoid the use of Latin-1 or Unicode punctuation characters. These characters are often inadvertently added when copying text from a word-processing application or scraping data from web pages. If these characters are present, make sure to update them with the appropriate ASCII substitution.

Here are a few examples:

 Expand table

Characters to avoid	Substitution	Notes
"Hello world"	"Hello world"	The opening and closing quotations marks are substituted with appropriate ASCII characters.
John's day	John's day	The apostrophe is substituted with the appropriate ASCII character.
It was good—no, it was great!	it was good--no, it was great!	The em dash is substituted with two hyphens.

Text normalization for US English

Text normalization is the transformation of words into a consistent format used when training a model. Some normalization rules are applied to text automatically, however, we recommend using these guidelines as you prepare your human-labeled transcription data:

- Write out abbreviations in words.
- Write out nonstandard numeric strings in words (such as accounting terms).
- Nonalphanumeric characters or mixed alphanumeric characters should be transcribed as pronounced.
- Abbreviations that are pronounced as words shouldn't be edited (such as "radar", "laser", "RAM", or "NATO").
- Write out abbreviations that are pronounced as separate letters with each letter separated by a space.
- If you use audio, transcribe numbers as words that match the audio (for example, "101" could be pronounced as "one oh one" or "one hundred and one").
- Avoid repeating characters, words, or groups of words more than three times, such as "yeah yeah yeah yeah". The Speech service might drop lines with such repetition.

Here are a few examples of normalization that you should perform on the transcription:

 Expand table

Original text	Text after normalization (human)
Dr. Bruce Banner	Doctor Bruce Banner
James Bond, 007	James Bond, double oh seven
Ke\$ha	Kesha
How long is the 2x4	How long is the two by four
The meeting goes from 1-3pm	The meeting goes from one to three pm
My blood type is O+	My blood type is O positive
Water is H2O	Water is H 2 O
Play OU812 by Van Halen	Play O U 8 1 2 by Van Halen
UTF-8 with BOM	U T F 8 with BOM
It costs \$3.14	It costs three fourteen

The following normalization rules are automatically applied to transcriptions:

- Use lowercase letters.
- Remove all punctuation except apostrophes within words.
- Expand numbers into words/spoken form, such as dollar amounts.

Here are a few examples of normalization automatically performed on the transcription:

[\[+\] Expand table](#)

Original text	Text after normalization (automatic)
"Holy cow!" said Batman.	holy cow said batman
"What?" said Batman's sidekick, Robin.	what said batman's sidekick robin
Go get -em!	go get em
I'm double-jointed	I'm double jointed
104 Elm Street	one oh four Elm street
Tune to 102.7	tune to one oh two point seven
Pi is about 3.14	pi is about three point one four

de-DE

Human-labeled transcriptions for German audio must be UTF-8 encoded with a byte-order marker.

Text normalization for German

Text normalization is the transformation of words into a consistent format used when training a model. Some normalization rules are applied to text automatically, however, we recommend using these guidelines as you prepare your human-labeled transcription data:

- Write decimal points as "," and not ".".
- Write time separators as ":" and not "." (for example: 12:00 Uhr).
- Abbreviations such as "ca." aren't replaced. We recommend that you use the full spoken form.
- The four main mathematical operators (+, -, *, and /) are removed. We recommend replacing them with the written form: "plus," "minus," "mal," and "geteilt."
- Comparison operators are removed (=, <, and >). We recommend replacing them with "gleich," "kleiner als," and "grösser als."
- Write fractions, such as 3/4, in written form (for example: "drei viertel" instead of 3/4).
- Replace the "€" symbol with its written form "Euro."

Here are a few examples of normalization that you should perform on the transcription:

 [Expand table](#)

Original text	Text after user normalization	Text after system normalization
Es ist 12.23 Uhr	Es ist 12:23 Uhr	es ist zwölf uhr drei und zwanzig uhr
{12.45}	{12,45}	zwölf komma vier fünf
2 + 3 - 4	2 plus 3 minus 4	zwei plus drei minus vier

The following normalization rules are automatically applied to transcriptions:

- Use lowercase letters for all text.
- Remove all punctuation, including various types of quotation marks ("test", 'test', "test„ and «test» are OK).
- Discard rows with any special characters from this set: € ☒ ¥ ! § © ª ¬ ® ° ± ² μ × ÿ Ø¬¬.
- Expand numbers to spoken form, including dollar or Euro amounts.
- Accept umlauts only for a, o, and u. Others are replaced by "th" or discarded.

Here are a few examples of normalization automatically performed on the transcription:

[Expand table](#)

Original text	Text after normalization
Frankfurter Ring	frankfurter ring
¡Eine Frage!	eine frage
Wir, haben	wir haben

ja-JP

In Japanese (ja-JP), there's a maximum length of 90 characters for each sentence. Lines with longer sentences are discarded. To add longer text, insert a period in between.

zh-CN

Human-labeled transcriptions for Mandarin Chinese audio must be UTF-8 encoded with a byte-order marker. Avoid the use of half-width punctuation characters. These characters can be included inadvertently when you prepare the data in a word-processing program or scrape data from web pages. If these characters are present, make sure to update them with the appropriate full-width substitution.

Here are a few examples:

[Expand table](#)

Characters to avoid	Substitution	Notes
"你好"	"你好"	The opening and closing quotations marks are substituted with appropriate characters.
需要什么帮助?	需要什么帮助?	The question mark is substituted with the appropriate character.

Text normalization for Mandarin Chinese

Text normalization is the transformation of words into a consistent format used when training a model. Some normalization rules are applied to text automatically, however, we recommend using these guidelines as you prepare your human-labeled transcription data:

- Write out abbreviations in words.
- Write out numeric strings in spoken form.

Here are a few examples of normalization that you should perform on the transcription:

[Expand table](#)

Original text	Text after normalization
我今年 21	我今年二十一
3 号楼 504	三号楼五零四

The following normalization rules are automatically applied to transcriptions:

- Remove all punctuation.
- Expand numbers to spoken form.
- Convert full-width letters to half-width letters.
- Using uppercase letters for all English words.

Here are some examples of automatic transcription normalization:

[Expand table](#)

Original text	Text after normalization
3.1415	三点一四一五
¥ 3.5	三元五角
w f y z	W F Y Z
1992 年 8 月 8 日	一九九二年八月八日
你吃饭了吗?	你吃饭了吗
下午 5:00 的航班	下午五点的航班
我今年 21 岁	我今年二十一岁

Next Steps

- [Test model quantitatively](#)
- [Test recognition quality](#)
- [Train your model](#)

Structured text phonetic pronunciation data

08/07/2025

You can specify the phonetic pronunciation of words using the Universal Phone Set (UPS) in a [structured text data](#) file. The UPS is a machine-readable phone set that is based on the International Phonetic Set Alphabet (IPA). The IPA is a standard used by linguists world-wide.

UPS pronunciations consist of a string of UPS phonemes, each separated by whitespace. UPS phoneme labels are all defined using ASCII character strings.

For steps on implementing UPS, see [Structured text phonetic pronunciation](#). Structured text phonetic pronunciation data is separate from [pronunciation data](#), and they can't be used together. The first one is "sounds-like" or spoken-form data, and is input as a separate file, and trains the model what the spoken form sounds like

[Structured text phonetic pronunciation data](#) is specified per syllable in a markdown file. Separately, [pronunciation data](#) is input on its own, and trains the model what the spoken form sounds like. You can either use a pronunciation data file on its own, or you can add pronunciation within a structured text data file. The Speech service doesn't support training a model with both of those datasets as input.

See the sections in this article for the Universal Phone Set for each locale.

en-US

Consonants

[] [Expand table](#)

UPS Phonemes	IPA	Example
B	b	big
CH	tʃ / ʈʃ	chin
D	d	dig
DH	ð	then
F	f	fork

UPS Phonemes	IPA	Example
G	g	gut
H	h	help
JH	dʒ / dʒ	joy
K	k	cut
L	l	lid
M	m	mat
N	n	no
NG	ŋ	sing
P	p	put
R	ɹ	red
S	s	sit
SH	ʃ	she
T	t	talk
TH	θ	thin
V	v	vat
W	w	with
J	j	yard
Z	z	zap
ZH	ʒ	pleasure

Vowels

[\[+\] Expand table](#)

UPS Phonemes	IPA	Example
AA	a	father
AE	æ	cat

UPS Phonemes	IPA	Example
AH	ʌ	cut
AO	ɔ	dog
AOX	ɔ.ə	four
AU	a.ʊ	foul
AX	ə	ago
AX R	ə̯	minor
AI	a.ɪ	bite
EH	ɛ	pet
EHX	ɛ.ə	stairs
ER R	ɜ̯	urban
EI	e.ɪ	ate
IH	ɪ	fill
I	i	feel
O	o	go
OI	ɔ.ɪ	toy
OWX	o.ə	boa
Q	ɒ	hot
UH	ʊ	book
U	u	too, blue
UWX	u.ə	lure

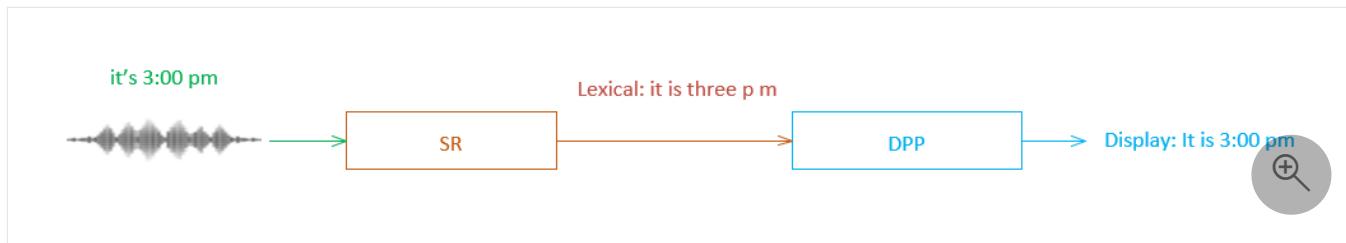
Next steps

- Upload your data
- Test recognition quality
- Train your model

How to prepare display text format training data for custom speech

05/19/2025

Azure AI Speech service can be viewed as two components: speech recognition and display text formatting. Speech recognition transcribes audio to lexical text, and then the lexical text is transformed to display text.



These are the locales that support the display text format feature: da-DK, de-DE, en-AU, en-CA, en-GB, en-HK, en-IE, en-IN, en-NG, en-NZ, en-PH, en-SG, en-US, es-ES, es-MX, fi-FI, fr-CA, fr-FR, hi-IN, it-IT, ja-JP, ko-KR, nb-NO, nl-NL, pl-PL, pt-BR, pt-PT, sv-SE, tr-TR, zh-CN, zh-HK.

Default display text formatting

The display text pipeline is composed by a sequence of display format builders. Each builder corresponds to a display format task such as ITN, capitalization, and profanity filtering.

- **Inverse Text Normalization (ITN)** - To convert the text of spoken form numbers to display form. For example: "I spend twenty dollars" -> "I spend \$20"
- **Capitalization** - To upper case entity names, acronyms, or the first letter of a sentence. For example: "she is from microsoft" -> "She is from Microsoft"
- **Profanity filtering** - Masking or removal of profanity words from a sentence. For example, assuming "abcd" is a profanity word, then the word is masked by profanity masking: "I never say abcd" -> "I never say ****"

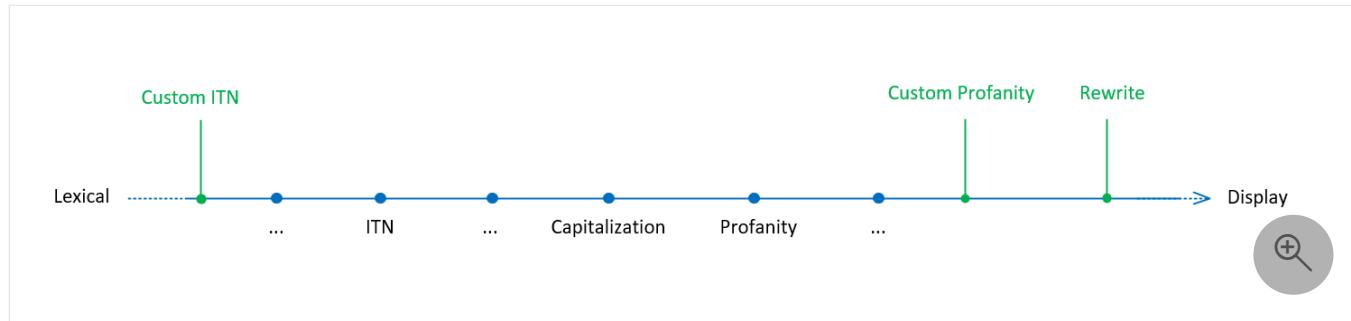
Microsoft maintains the base builders of the display text pipeline for the general purpose display processing tasks. You get the base builders by default when you use the Speech service. For more information about out-of-the-box formatting, see [Display text format](#).

Custom display text formatting

Besides the base builders maintained by Microsoft, you can define custom display text formatting rules to customize the display text formatting pipeline for your specific scenarios. The custom display text formatting rules are defined in a custom display text formatting file.

- [Custom ITN](#) - Extend the functionalities of base ITN, by applying a rule based custom ITN model from customer.
- [Custom rewrite](#) - Rewrite one phrase to another based on a rule based model from customer.
- [Custom profanity filtering](#) - Perform profanity handling based on the profanity word list from customer.

The order of the display text formatting pipeline is illustrated in this diagram.



Custom ITN

The philosophy of pattern-based custom ITN is that you can specify the final output that you want to see. The Speech service figures out how the words might be spoken and map the predicted spoken expressions to the specified output format.

A custom ITN model is built from a set of ITN rules. An ITN rule is a regular expression like pattern string, which describes:

- A matching pattern of the input string
- The desired format of the output string

The default ITN rules provided by Microsoft are applied first. The output of the default ITN model is used as the input of the custom ITN model. The matching algorithm inside the custom ITN model is case-insensitive.

There are four categories of pattern matching with custom ITN rules.

- [Patterns with literals](#)
- [Patterns with wildcards](#)
- [Patterns with Regex-style Notation](#)
- [Patterns with explicit replacement](#)

Patterns with literals

For example, a developer might have an item (such as a product) named with the alphanumeric form `J0:500`. The Speech service figures out that users might say the letter part as `J 0`, or they might say `joe`, and the number part as `five hundred` or `five zero zero` or `five oh oh` or `five double zero`, and then build a model that maps all of these possibilities back to `J0:500` (including inserting the colon).

Patterns can be applied in parallel by specifying one rule per line in the display text formatting file. Here's an example of a display text formatting file that specifies two rules:

```
text
J0:500
MM:760
```

Patterns with wildcards

You can refer to a whole series of alphanumeric items (such as `J0:500`, `J0:600`, `J0:700`) without having to spell out all possibilities in several ways.

Character ranges can be specified with the notation `[...]`, so `J0:[5-7]00` is equivalent to writing out three patterns.

There's also a set of wildcard items that can be used. One of these is `\d`, which means any digit. So `J0:\d00` covers `J0:000`, `J0:100`, and others up to `J0:900`.

Like a regular expression, there are multiple predefined character classes for an ITN rule:

- `\d` - match a digit from '0' to '9', and output it directly
- `\l` - match a letter (case-insensitive) and transduce it to lower case
- `\u` - match a letter (case-insensitive) and transduce it to upper case
- `\a` - match a letter (case-insensitive) and output it directly

There are also escape expressions for referring to characters that otherwise have special syntactic meaning:

- `\\\` - match and output the char `\`
- `\(` and `\)`
- `\{` and `\}`
- `\|`
- `\+` and `\?` and `*`

Patterns with regex-style notation

To enhance the flexibility of pattern writing, regular expression-like constructions of phrases with alternatives and Kleene-closure are supported.

- A phrase is indicated with parentheses, like `(...)` - The parentheses don't literally count as characters to be matched.
- You can indicate alternatives within a phrase with the `|` character such as `(AB|CDE)`.
- You can suffix a phrase with `?` to indicate that it's optional, `+` to indicate that it can be repeated, or `*` to indicate both. You can only suffix phrases with these characters and not individual characters (which is more restrictive than most regular expression implementations).

A pattern such as `(AB|CD)-(\d)+'` would represent constructs like "AB-9" or "CD-22" and be expanded to spoken words like `A B nine` and `C D twenty two` (or `C D two two`).

Patterns with explicit replacement

The general philosophy is "you show us what the output should look like, and the Speech service figures out how people say it." But this doesn't always work because some scenarios might have quirky unpredictable ways of saying things, or the Speech service background rules might have gaps. For example, there can be colloquial pronunciations for initials and acronyms - `ZPI` might be spoken as `zippy`. In this case, a pattern like `ZPI-\d\d` is unlikely to work if a user says `zippy twenty two`. For this sort of situation, there's a display text format notation `{spoken>written}`. This particular case could be written out `{zippy>ZPI}-\d\d`.

This can be useful for handling things that the Speech mapping rules but don't yet support. For example you might write a pattern `\d0-\d0` expecting the system to understand that `-` can mean a range, and should be pronounced `to`, as in `twenty to thirty`. But perhaps it doesn't. So you can write a more explicit pattern like `\d0{to>-}\d0` and tell it how you expect the dash to be read.

You can also leave out the `>` and following written form to indicate words that should be recognized but ignored. So a pattern like `{write} (\u.)+` recognizes `write A B C` and output `A.B.C` --dropping the `write` part.

Custom ITN Examples

Group digits

To group 6 digits into two groups and add a `'-` character between them:

ITN rule: \d\d\d-\d\d\d Sample: "cadence one oh five one fifteen" -> "cadence 105-115"

Format a film name

Space: 1999 is a famous film, to support it:

ITN rule: Space: 1999 Sample: "watching space nineteen ninety nine" -> "watching
Space: 1999"

Pattern with Replacement

ITN rule: \d[05]{ to >-}\d[05] Sample: fifteen to twenty -> 15-20

Custom rewrite

General speaking, for an input string, rewrite model tries to replace the `original phrase` in the input string with the corresponding `new phrase` for each rewrite rule. A rewrite model is a collection of rewrite rules.

- A rewrite rule is a pair of two phrases: the original phrase and a new phrase.
- The two phrases are separated by a TAB character. For example, `original phrase`{TAB}`new phrase`.
- The original phrase is matched (case-insensitive) and replaced with the new phrase (case-sensitive). [Grammar punctuation characters](#) in the original phrase are ignored during match.
- If any rewrite rules conflict, the one with the longer `original phrase` is used as the match.

The rewrite model supports grammar capitalization by default, which capitalizes the first letter of a sentence for `en-US` like locales. It's turned off if the capitalization feature of display text formatting is turned off in a speech recognition request.

Grammar punctuation

Grammar punctuation characters are used to separate a sentence or phrase, and clarify how a sentence or phrase should be read.

. , ? ! : ; ? . , : i | ? ,

Here are the grammar punctuation rules:

- The supported punctuation characters are for grammar punctuation if they're followed by space or at the beginning or end of a sentence or phrase. For example, the `.` in `x. y` (with a space between `.` and `y`) is a grammar punctuation.
- Punctuation characters that are in the middle of a word (except `zh-cn` and `ja-jp`) aren't grammar punctuation. In that case, they're ordinary characters. For example, the `.` in `x.y` isn't a grammar punctuation.
- For `zh-cn` and `ja-jp` (nonspacing locales), punctuation characters are always used as grammar punctuation even if they are between characters. For example, the `.` in `中.文` is a grammar punctuation.

Custom rewrite examples

Spelling correction

The name `COVID-19` might be recognized as `covered 19`. To make sure that `COVID-19 is a virus` is displayed instead of `covered 19 is a virus`, use the following rewrite rule:

```
text
#rewrite
covered 19{TAB}COVID-19
```

Name capitalization

Gottfried Wilhelm Leibniz was a German mathematician. To make sure that `Gottfried Wilhelm Leibniz` is capitalized, use the following rewrite rule:

```
text
#rewrite
gottfried leibniz{TAB}Gottfried Leibniz
```

Custom profanity

A custom profanity model acts the same as the base profanity model, except it uses a custom profanity phrase list. In addition, the custom profanity model tries to match (case insensitive) all the profanity phrases defined in the display text formatting file.

- The profanity phrases are matched (case-insensitive).
- If any profanity phrases rules conflict, the longest phrase is used as the match.

- These punctuation characters aren't supported in a profanity phrase: . , ? ! : ; ?
., ! ? ; .
- For zh-CN and ja-JP locales, English profanity phrases aren't supported. English profanity words are supported. Profanity phrases for zh-CN and ja-JP locales are supported.

The profanity is removed or masked depending on your speech recognition request settings.

Once profanity is added in the display text format rule file and the custom model is trained, it's used for the default output in batch speech to text and real-time speech to text.

Custom profanity examples

Here are some examples of how to mask profanity words and phrases in the display text formatting file.

Mask single profanity word example

Assume xyz is a profanity word. To add it:

```
text
```

```
#profanity  
xyz
```

Here's a test sample: Turned on profanity masking to mask xyz -> Turned on profanity masking to mask ***

Mask profanity phrase

Assume abc lmn is a profanity phrase. To add it:

```
text
```

```
#profanity  
abc lmn
```

Here's a test sample: Turned on profanity masking to mask abc lmn -> Turned on profanity masking to mask *** ***

Next Steps

- Test model quantitatively
- Test recognition quality
- Train your model

Custom speech model lifecycle

07/31/2025

You can use a custom speech model for some time after you deploy it to your custom endpoint. But when new base models are made available, the older models are expired. You must periodically recreate and train your custom model from the latest base model to take advantage of the improved accuracy and quality.

Here are some key terms related to the model lifecycle:

- **Training:** Taking a base model and customizing it to your domain/scenario by using text data and/or audio data. In some contexts such as the REST API properties, training is also referred to as **adaptation**.
- **Transcription:** Using a model and performing speech recognition (decoding audio into text).
- **Endpoint:** A specific deployment of either a base model or a custom model that only you can access.

! Note

Endpoints used by `F0` Speech resources are deleted after seven days.

Expiration timeline

Here are timelines for model adaptation and transcription expiration:

- Training is available for one year after the quarter when Microsoft created the base model.
- Transcription with a base model is available for two years after the quarter when Microsoft created the base model.
- Transcription with a custom model is available for two years after the quarter when you created the custom model.

In this context, quarters end on January 15, April 15, July 15, and October 15.

What to do when a model expires

When a custom model or base model expires, it's no longer available for transcription. You can change the model that is used by your custom speech endpoint without downtime.

Transcription route	Expired model result	Recommendation
Custom endpoint	Speech recognition requests fall back to the most recent base model for the same locale . You get results, but recognition might not accurately transcribe your domain data.	Update the endpoint's model as described in the Deploy a custom speech model guide.
Batch transcription	Batch transcription requests for expired models fail with a 4xx error.	In each Transcriptions - Submit REST API request body, set the <code>model</code> property to a base model or custom model that isn't expired. Otherwise don't include the <code>model</code> property to always use the latest base model.

Get base model expiration dates

 Tip

Bring your custom speech models from [Speech Studio](#) ↗ to the [Azure AI Foundry portal](#) ↗. In Azure AI Foundry portal, you can pick up where you left off by connecting to your existing Speech resource. For more information about connecting to an existing Speech resource, see [Connect to an existing Speech resource](#).

The last date that you could use the base model for training was shown when you created the custom model. For more information, see [Train a custom speech model](#).

Follow these instructions to get the transcription expiration date for a base model:

1. Sign in to the [Azure AI Foundry portal](#) ↗.
2. Select **Fine-tuning** from the left pane.
3. Select **AI Service fine-tuning**.
4. Select the custom model that you want to check from the **Model name** column.
5. Select **Deploy models**.
6. The expiration date for the model is shown in the **Expiration** column. This date is the last date that you can use the model for transcription.

Get custom model expiration dates

Follow these instructions to get the transcription expiration date for a custom model:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane.
3. Select **AI Service fine-tuning**.
4. Select the custom model that you want to check from the **Model name** column.
5. Select **Deploy models**.
6. The expiration date for the model is shown in the **Expiration** column. This date is the last date that you can use the model for transcription.

Related content

- [Train a model](#)
- [Custom speech overview](#)

Use pronunciation assessment

In this article, you learn how to evaluate pronunciation with speech to text through the Speech SDK. Pronunciation assessment evaluates speech pronunciation and gives speakers feedback on the accuracy and fluency of spoken audio.

! Note

Pronunciation assessment uses a specific version of the speech-to-text model, different from the standard speech to text model, to ensure consistent and accurate pronunciation assessment.

Use pronunciation assessment in streaming mode

Pronunciation assessment supports uninterrupted streaming mode. The recording time can be unlimited through the Speech SDK. As long as you don't stop recording, the evaluation process doesn't finish and you can pause and resume evaluation conveniently.

For information about availability of pronunciation assessment, see [supported languages](#) and [available regions](#).

As a baseline, usage of pronunciation assessment costs the same as speech to text for Standard or commitment tier [pricing](#). If you [purchase a commitment tier](#) for speech to text, the spend for pronunciation assessment goes towards meeting the commitment. For more information, see [Pricing](#).

For how to use Pronunciation Assessment in streaming mode in your own application, see [sample code](#).

Continuous recognition

If your audio file exceeds 30 seconds, use continuous mode for processing. In continuous mode, the `EnableMiscue` option is not supported. To obtain `Omission` and `Insertion` tags, you need to compare the recognized results with the reference text. You can find a sample implementation for continuous mode on [GitHub](#) under the function

`PronunciationAssessmentContinuousWithFile`.

Set configuration parameters

In the `SpeechRecognizer`, you can specify the language to learn or practice improving pronunciation. The default locale is `en-US`. To learn how to specify the learning language for pronunciation assessment in your own application, you can use the following sample code.

C#

```
var recognizer = new SpeechRecognizer(speechConfig, "en-US", audioConfig);
```

💡 Tip

If you aren't sure which locale to set for a language that has multiple locales, try each locale separately. For instance, for Spanish, try `es-ES` and `es-MX`. Determine which locale scores higher for your scenario.

You must create a `PronunciationAssessmentConfig` object. You can set `EnableProsodyAssessment` to enable prosody assessment. For more information, see [configuration methods](#).

C#

```
var pronunciationAssessmentConfig = new PronunciationAssessmentConfig(
    referenceText: "",
    gradingSystem: GradingSystem.HundredMark,
    granularity: Granularity.Phoneme,
    enableMiscue: false);
pronunciationAssessmentConfig.EnableProsodyAssessment();
```

This table lists some of the key configuration parameters for pronunciation assessment.

[] [Expand table](#)

Parameter	Description
<code>ReferenceText</code>	The text that the pronunciation is evaluated against. The <code>ReferenceText</code> parameter is optional. Set the reference text if you want to run a scripted assessment for the reading language learning scenario. Don't set the reference text if you want to run an unscripted assessment . For pricing differences between scripted and unscripted assessment, see Pricing .
<code>GradingSystem</code>	The point system for score calibration. <code>FivePoint</code> gives a 0-5 floating point score. <code>HundredMark</code> gives a 0-100 floating point score. Default: <code>FivePoint</code> .
<code>Granularity</code>	Determines the lowest level of evaluation granularity. Returns scores for levels greater than or equal to the minimal value. Accepted values are <code>Phoneme</code> , which shows the score on the full text, word, syllable, and phoneme level, <code>Word</code> , which shows the score on the

Parameter	Description
	full text and word level, or <code>FullText</code> , which shows the score on the full text level only. The provided full reference text can be a word, sentence, or paragraph. It depends on your input reference text. Default: <code>Phoneme</code> .
<code>EnableMispell</code>	Enables mispell calculation when the pronounced words are compared to the reference text. Enabling mispell is optional. If this value is <code>True</code> , the <code>ErrorType</code> result value can be set to <code>Omission</code> or <code>Insertion</code> based on the comparison. Values are <code>False</code> and <code>True</code> . Default: <code>False</code> . To enable mispell calculation, set the <code>EnableMispell</code> to <code>True</code> . You can refer to the code snippet above the table.
<code>ScenarioId</code>	A GUID for a customized point system.

Configuration methods

This table lists some of the optional methods you can set for the `PronunciationAssessmentConfig` object.

 **Note**

Prosody assessment is only available in the [en-US](#) locale.

To explore the prosody assessment, upgrade to the SDK version 1.35.0 or later.

 [Expand table](#)

Method	Description
<code>EnableProsodyAssessment</code>	Enables prosody assessment for your pronunciation evaluation. This feature assesses aspects like stress, intonation, speaking speed, and rhythm. This feature provides insights into the naturalness and expressiveness of your speech. Enabling prosody assessment is optional. If this method is called, the <code>ProsodyScore</code> result value is returned.

Get pronunciation assessment results

When speech is recognized, you can request the pronunciation assessment results as SDK objects or a JSON string.

C#

```

using (var speechRecognizer = new SpeechRecognizer(
    speechConfig,
    audioConfig))
{
    // (Optional) get the session ID
    speechRecognizer.SessionStarted += (s, e) => {
        Console.WriteLine($"SESSION ID: {e.SessionId}");
    };
    pronunciationAssessmentConfig.ApplyTo(speechRecognizer);
    var speechRecognitionResult = await speechRecognizer.RecognizeOnceAsync();

    // The pronunciation assessment result as a Speech SDK object
    var pronunciationAssessmentResult =
        PronunciationAssessmentResult.FromResult(speechRecognitionResult);

    // The pronunciation assessment result as a JSON string
    var pronunciationAssessmentResultJson =
        speechRecognitionResult.Properties.GetProperty(PropertyId.SpeechServiceResponse_Json
    Result);
}

```

Result parameters

Depending on whether you're using [scripted](#) or [unscripted](#) assessment, you can get different pronunciation assessment results. Scripted assessment is for the reading language learning scenario. Unscripted assessment is for the speaking language learning scenario.

 **Note**

For pricing differences between scripted and unscripted assessment, see [Pricing](#).

Scripted assessment results

This table lists some of the key pronunciation assessment results for the scripted assessment, or reading scenario.

 [Expand table](#)

Parameter	Description	Granularity
AccuracyScore	Pronunciation accuracy of the speech. Accuracy indicates how closely the phonemes match a native speaker's pronunciation. Syllable, word, and full text accuracy scores are aggregated from the phoneme-level accuracy score, and refined with assessment objectives.	Phoneme level, Syllable level (en-US only),

Parameter	Description	Granularity
		Word level, Full Text level
FluencyScore	Fluency of the given speech. Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words.	Full Text level
CompletenessScore	Completeness of the speech, calculated by the ratio of pronounced words to the input reference text.	Full Text level
ProsodyScore	Prosody of the given speech. Prosody indicates how natural the given speech is, including stress, intonation, speaking speed, and rhythm.	Full Text level
PronScore	Overall score of the pronunciation quality of the given speech. PronScore is calculated from AccuracyScore, FluencyScore, CompletenessScore, and ProsodyScore with weight, provided that ProsodyScore and CompletenessScore are available. If either of them isn't available, PronScore won't consider that score.	Full Text level
ErrorType	This value indicates the error type compared to the reference text. Options include whether a word is omitted, inserted, or improperly inserted with a break. It also indicates a missing break at punctuation. It also indicates whether a word is badly pronounced, or monotonically rising, falling, or flat on the utterance. Possible values are None for no error on this word, Omission, Insertion, Mispronunciation, UnexpectedBreak, MissingBreak, and Monotone. The error type can be Mispronunciation when the pronunciation AccuracyScore for a word is below 60.	Word level

Unscripted assessment results

This table lists some of the key pronunciation assessment results for the unscripted assessment, or speaking scenario.

Note

Prosody assessment is only available in the [en-US](#) locale. For unscripted assessment, the speech-to-text (STT) model used is different from Azure STT. If you need assessment based on highly accurate recognized text, we recommend first calling Azure STT to obtain the reference text, and then performing scripted assessment.

 [Expand table](#)

Response parameter	Description	Granularity
AccuracyScore	Pronunciation accuracy of the speech. Accuracy indicates how closely the phonemes match a native speaker's pronunciation. Syllable, word, and full text accuracy scores are aggregated from phoneme-level accuracy score, and refined with assessment objectives.	Phoneme level, Syllable level (en-US only), Word level, Full Text level
FluencyScore	Fluency of the given speech. Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words.	Full Text level
ProsodyScore	Prosody of the given speech. Prosody indicates how natural the given speech is, including stress, intonation, speaking speed, and rhythm.	Full Text level
PronScore	Overall score of the pronunciation quality of the given speech. PronScore is calculated from AccuracyScore, FluencyScore, and ProsodyScore with weight, provided that ProsodyScore is available. If ProsodyScore isn't available, PronScore won't consider that score.	Full Text level
ErrorType	A word is badly pronounced, improperly inserted with a break, or missing a break at punctuation. It also indicates whether a pronunciation is monotonically rising, falling, or flat on the utterance. Possible values are None for no error on this word, Mispronunciation, UnexpectedBreak, MissingBreak, and Monotone.	Word level

The following table describes the prosody assessment results in more detail:

[Expand table](#)

Field	Description
ProsodyScore	Prosody score of the entire utterance.
Feedback	Feedback on the word level, including Break and Intonation.
Break	
ErrorTypes	Error types related to breaks, including UnexpectedBreak and MissingBreak. The current version doesn't provide the break error type. You need to set thresholds on the fields UnexpectedBreak - Confidence and MissingBreak - confidence to decide whether there's an unexpected break or missing break before the word.
UnexpectedBreak	Indicates an unexpected break before the word.
MissingBreak	Indicates a missing break before the word.
Thresholds	Suggested thresholds on both confidence scores are 0.75. That means, if the value of UnexpectedBreak - Confidence is larger than 0.75, it has an unexpected break. If the

Field	Description
	value of <code>MissingBreak - confidence</code> is larger than 0.75, it has a missing break. While 0.75 is a value we recommend, it's better to adjust the thresholds based on your own scenario. If you want to have variable detection sensitivity on these two breaks, you can assign different thresholds to the <code>UnexpectedBreak - Confidence</code> and <code>MissingBreak - Confidence</code> fields.
<code>Intonation</code>	Indicates intonation in speech.
<code>ErrorTypes</code>	Error types related to intonation, currently supporting only Monotone. If the <code>Monotone</code> exists in the field <code>ErrorTypes</code> , the utterance is detected to be monotonic. Monotone is detected on the whole utterance, but the tag is assigned to all the words. All the words in the same utterance share the same monotone detection information.
<code>Monotone</code>	Indicates monotonic speech.
<code>Thresholds</code> <code>(Monotone</code> <code>Confidence)</code>	The fields <code>Monotone - SyllablePitchDeltaConfidence</code> are reserved for user-customized monotone detection. If you're unsatisfied with the provided monotone decision, adjust the thresholds on these fields to customize the detection according to your preferences.

JSON result example

The [scripted](#) pronunciation assessment results for the spoken word "hello" are shown as a JSON string in the following example.

- The phoneme [alphabet](#) is IPA.
- The [syllables](#) are returned alongside phonemes for the same word.
- You can use the `Offset` and `Duration` values to align syllables with their corresponding phonemes. For example, the starting offset (11700000) of the second syllable `lou` aligns with the third phoneme, `l`. The offset represents the time at which the recognized speech begins in the audio stream. The value is measured in 100-nanosecond units. To learn more about `Offset` and `Duration`, see [response properties](#).
- There are five `NBestPhonemes` that correspond to the number of [spoken phonemes](#) requested.
- Within `Phonemes`, the most likely [spoken phonemes](#) was `ə` instead of the expected phoneme `ɛ`. The expected phoneme `ɛ` only received a confidence score of 47. Other potential matches received confidence scores of 52, 17, and 2.

JSON

```
{
  "Id": "bbb42ea51bdb46d19a1d685e635fe173",
  "RecognitionStatus": 0,
```

```
"Offset": 7500000,
"Duration": 13800000,
"DisplayText": "Hello.",
"NBest": [
  {
    "Confidence": 0.975003,
    "Lexical": "hello",
    "ITN": "hello",
    "MaskedITN": "hello",
    "Display": "Hello.",
    "PronunciationAssessment": {
      "AccuracyScore": 100,
      "FluencyScore": 100,
      "CompletenessScore": 100,
      "PronScore": 100
    },
    "Words": [
      {
        "Word": "hello",
        "Offset": 7500000,
        "Duration": 13800000,
        "PronunciationAssessment": {
          "AccuracyScore": 99.0,
          "ErrorType": "None"
        },
        "Syllables": [
          {
            "Syllable": "hε",
            "PronunciationAssessment": {
              "AccuracyScore": 91.0
            },
            "Offset": 7500000,
            "Duration": 4100000
          },
          {
            "Syllable": "lou",
            "PronunciationAssessment": {
              "AccuracyScore": 100.0
            },
            "Offset": 11700000,
            "Duration": 9600000
          }
        ],
        "Phonemes": [
          {
            "Phoneme": "h",
            "PronunciationAssessment": {
              "AccuracyScore": 98.0,
              "NBestPhonemes": [
                {
                  "Phoneme": "h",
                  "Score": 100.0
                },
                {
                  "Phoneme": "oʊ",
                  "Score": 99.0
                }
              ]
            }
          }
        ]
      }
    ]
  }
]
```

```
        "Score": 52.0
    },
    {
        "Phoneme": "ə",
        "Score": 35.0
    },
    {
        "Phoneme": "k",
        "Score": 23.0
    },
    {
        "Phoneme": "æ",
        "Score": 20.0
    }
]
},
"Offset": 7500000,
"Duration": 3500000
},
{
    "Phoneme": "ε",
    "PronunciationAssessment": {
        "AccuracyScore": 47.0,
        "NBestPhonemes": [
            {
                "Phoneme": "ə",
                "Score": 100.0
            },
            {
                "Phoneme": "l",
                "Score": 52.0
            },
            {
                "Phoneme": "ε",
                "Score": 47.0
            },
            {
                "Phoneme": "h",
                "Score": 17.0
            },
            {
                "Phoneme": "æ",
                "Score": 2.0
            }
        ]
    },
    "Offset": 11100000,
    "Duration": 500000
},
{
    "Phoneme": "l",
    "PronunciationAssessment": {
        "AccuracyScore": 100.0,
        "NBestPhonemes": [
            {

```

```
        "Phoneme": "l",
        "Score": 100.0
    },
    {
        "Phoneme": "ou",
        "Score": 46.0
    },
    {
        "Phoneme": "ə",
        "Score": 5.0
    },
    {
        "Phoneme": "ɛ",
        "Score": 3.0
    },
    {
        "Phoneme": "u",
        "Score": 1.0
    }
]
},
{
    "Offset": 11700000,
    "Duration": 1100000
},
{
    "Phoneme": "ou",
    "PronunciationAssessment": {
        "AccuracyScore": 100.0,
        "NBestPhonemes": [
            {
                "Phoneme": "ou",
                "Score": 100.0
            },
            {
                "Phoneme": "d",
                "Score": 29.0
            },
            {
                "Phoneme": "t",
                "Score": 24.0
            },
            {
                "Phoneme": "n",
                "Score": 22.0
            },
            {
                "Phoneme": "l",
                "Score": 18.0
            }
        ]
    },
    "Offset": 12900000,
    "Duration": 8400000
}
]
```

```
        ]  
    }  
]  
}
```

You can get pronunciation assessment scores for:

- Full text
- Words
- Syllable groups
- Phonemes in [SAPI](#) or [IPA](#) format

Supported features per locale

The following table summarizes which features that locales support. For more specifics, see the following sections. If the locales you require aren't listed in the following table for the supported feature, fill out this [intake form](#) for further assistance.

[\[\] Expand table](#)

Phoneme alphabet	IPA	SAPI
Phoneme name	en-US	en-US, zh-CN
Syllable group	en-US	en-US
Spoken phoneme	en-US	en-US

Syllable groups

Pronunciation assessment can provide syllable-level assessment results. A word is typically pronounced syllable by syllable rather than phoneme by phoneme. Grouping in syllables is more legible and aligned with speaking habits.

Pronunciation assessment supports syllable groups only in `en-us` with IPA and with SAPI.

The following table compares example phonemes with the corresponding syllables.

[\[\] Expand table](#)

Sample word	Phonemes	Syllables
technological	teknələdʒɪkl	tek-nə-la-dʒɪkl

Sample word	Phonemes	Syllables
hello	həloʊ	hə-loʊ
luck	lʌk	lʌk
photosynthesis	fəʊtəsɪnθəsɪs	fou-tə-sin-θə-sis

To request syllable-level results along with phonemes, set the granularity [configuration parameter](#) to `Phoneme`.

Phoneme alphabet format

Pronunciation assessment supports phoneme name in `en-US` with IPA and in `en-US` and `zh-CN` with SAPI.

For locales that support phoneme name, the phoneme name is provided together with the score. Phoneme names help identify which phonemes were pronounced accurately or inaccurately. For other locales, you can only get the phoneme score.

The following table compares example SAPI phonemes with the corresponding IPA phonemes.

[Expand table](#)

Sample word	SAPI Phonemes	IPA phonemes
hello	h eh l ow	h ε l oʊ
luck	l ah k	l ʌ k
photosynthesis	f ow t ax s ih n th ax s ih s	f ou t ə s i n θ ə s i s

To request IPA phonemes, set the phoneme alphabet to `IPA`. If you don't specify the alphabet, the phonemes are in SAPI format by default.

C#

```
pronunciationAssessmentConfig.PhonemeAlphabet = "IPA";
```

Assess spoken phonemes

With spoken phonemes, you can get confidence scores that indicate how likely the spoken phonemes matched the expected phonemes.

Pronunciation assessment supports spoken phonemes in `en-US` with IPA and with SAPI.

For example, to obtain the complete spoken sound for the word `Hello`, you can concatenate the first spoken phoneme for each expected phoneme with the highest confidence score. In the following assessment result, when you speak the word `hello`, the expected IPA phonemes are `h ε 1 oʊ`. However, the actual spoken phonemes are `h ə 1 oʊ`. You have five possible candidates for each expected phoneme in this example. The assessment result shows that the most likely spoken phoneme was `ə` instead of the expected phoneme `ε`. The expected phoneme `ε` only received a confidence score of 47. Other potential matches received confidence scores of 52, 17, and 2.

JSON

```
{  
    "Id": "bbb42ea51bdb46d19a1d685e635fe173",  
    "RecognitionStatus": 0,  
    "Offset": 7500000,  
    "Duration": 13800000,  
    "DisplayText": "Hello.",  
    "NBest": [  
        {  
            "Confidence": 0.975003,  
            "Lexical": "hello",  
            "ITN": "hello",  
            "MaskedITN": "hello",  
            "Display": "Hello.",  
            "PronunciationAssessment": {  
                "AccuracyScore": 100,  
                "FluencyScore": 100,  
                "CompletenessScore": 100,  
                "PronScore": 100  
            },  
            "Words": [  
                {  
                    "Word": "hello",  
                    "Offset": 7500000,  
                    "Duration": 13800000,  
                    "PronunciationAssessment": {  
                        "AccuracyScore": 99.0,  
                        "ErrorType": "None"  
                    },  
                    "Syllables": [  
                        {  
                            "Syllable": "hε",  
                            "PronunciationAssessment": {  
                                "AccuracyScore": 91.0  
                            },  
                            "Offset": 7500000,  
                            "Duration": 4100000  
                        },  
                        {  
                            "Syllable": "ləʊ",  
                            "PronunciationAssessment": {  
                                "AccuracyScore": 47.0  
                            },  
                            "Offset": 8100000,  
                            "Duration": 3700000  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
}
```

```
        "AccuracyScore": 100.0
    },
    "Offset": 11700000,
    "Duration": 9600000
}
],
"Phonemes": [
{
    "Phoneme": "h",
    "PronunciationAssessment": {
        "AccuracyScore": 98.0,
        "NBestPhonemes": [
            {
                "Phoneme": "h",
                "Score": 100.0
            },
            {
                "Phoneme": "o\u026a",
                "Score": 52.0
            },
            {
                "Phoneme": "\u028a",
                "Score": 35.0
            },
            {
                "Phoneme": "k",
                "Score": 23.0
            },
            {
                "Phoneme": "\u028e",
                "Score": 20.0
            }
        ]
    },
    "Offset": 7500000,
    "Duration": 3500000
},
{
    "Phoneme": "\u028c",
    "PronunciationAssessment": {
        "AccuracyScore": 47.0,
        "NBestPhonemes": [
            {
                "Phoneme": "\u028a",
                "Score": 100.0
            },
            {
                "Phoneme": "l",
                "Score": 52.0
            },
            {
                "Phoneme": "\u028c",
                "Score": 47.0
            }
        ]
    }
}
```

```
        "Phoneme": "h",
        "Score": 17.0
    },
    {
        "Phoneme": "æ",
        "Score": 2.0
    }
]
},
"Offset": 11100000,
"Duration": 500000
},
{
    "Phoneme": "l",
    "PronunciationAssessment": {
        "AccuracyScore": 100.0,
        "NBestPhonemes": [
            {
                "Phoneme": "l",
                "Score": 100.0
            },
            {
                "Phoneme": "ou",
                "Score": 46.0
            },
            {
                "Phoneme": "ə",
                "Score": 5.0
            },
            {
                "Phoneme": "ɛ",
                "Score": 3.0
            },
            {
                "Phoneme": "u",
                "Score": 1.0
            }
        ]
    },
    "Offset": 11700000,
    "Duration": 1100000
},
{
    "Phoneme": "ou",
    "PronunciationAssessment": {
        "AccuracyScore": 100.0,
        "NBestPhonemes": [
            {
                "Phoneme": "ou",
                "Score": 100.0
            },
            {
                "Phoneme": "d",
                "Score": 29.0
            }
        ]
    }
}
```

```

        {
            "Phoneme": "t",
            "Score": 24.0
        },
        {
            "Phoneme": "n",
            "Score": 22.0
        },
        {
            "Phoneme": "l",
            "Score": 18.0
        }
    ]
},
"Offset": 12900000,
"Duration": 8400000
}
]
}
]
}
}

```

To indicate whether, and how many potential spoken phonemes to get confidence scores for, set the `NBestPhonemeCount` parameter to an integer value such as `5`.

C#

```
pronunciationAssessmentConfig.NBestPhonemeCount = 5;
```

Pronunciation score calculation

Pronunciation scores are calculated by weighting accuracy, prosody, fluency, and completeness scores based on specific formulas for reading and speaking scenarios.

When sorting the scores of accuracy, prosody, fluency, and completeness from low to high (if each score is available) and representing the lowest score to the highest score as s_0 to s_3 , the pronunciation score is calculated as follows:

For reading scenario:

- With prosody score: $\text{PronScore} = 0.4 * s_0 + 0.2 * s_1 + 0.2 * s_2 + 0.2 * s_3$
- Without prosody score: $\text{PronScore} = 0.6 * s_0 + 0.2 * s_1 + 0.2 * s_2$

For the speaking scenario (the completeness score isn't applicable):

- With prosody score: $\text{PronScore} = 0.6 * s_0 + 0.2 * s_1 + 0.2 * s_2$

- Without prosody score: PronScore = 0.6 * s0 + 0.4 * s1

This formula provides a weighted calculation based on the importance of each score, ensuring a comprehensive evaluation of pronunciation.

Content assessment

Important

Content assessment (preview) is retired from Speech SDK versions 1.46.0 and later. As an alternative, you can use Azure OpenAI in Azure AI Foundry Models to get content assessment results as described in this section.

For some recognized speech, you might also want to get content assessment results for vocabulary, grammar, and topic relevance. You can use a chat model such as Azure OpenAI `gpt-4o` to get the content assessment results. For more information about using chat models, see [Azure OpenAI models](#) and the Azure AI Model Inference API [chat completions reference documentation](#).

The user and system messages are used to set the context for the chat model. In the following example, the user message contains the essay to be assessed, and the system message provides instructions on how to evaluate the essay.

JSON

```
{
  "messages": [
    {
      "role": "system",
      "content": "You are an English teacher and please help to grade a student's essay from vocabulary and grammar and topic relevance on how well the essay aligns with the title, and output format as: {\\"vocabulary\\": *.*(0-100), \\"grammar\\": *.*(0-100), \\"topic\\": *.*(0-100)}."
    },
    {
      "role": "user",
      "content": "Example1: this essay: \"sampleSentence1\" has vocabulary and grammar scores of ** and **, respectively. Example2: this essay: \"sampleSentence2\" has vocabulary and grammar scores of ** and **, respectively. Example3: this essay: \"sampleSentence3\" has vocabulary and grammar scores of ** and **, respectively. The essay for you to score is \"sendText\", and the title is \"topic\". The transcript is from speech recognition so that please first add punctuations when needed, remove duplicates and unnecessary un uh from oral speech, then find all the misuse of words and grammar errors in this essay, find advanced words and grammar usages, and finally give scores based on this information. Please only respond as this format {\\"vocabulary\\": *.*(0-100), \\"grammar\\": *.*(0-100)}, \\"topic\\": *.*(0-100)"}
  ]
}
```

```
100)}. [THE TRANSCRIPT FROM SPEECH RECOGNITION IS REDACTED FOR BREVITY]"  
    }  
]  
}
```

Related content

- Learn about quality [benchmark](#).
- Try [pronunciation assessment in the studio](#).
- Check out an easy-to-deploy Pronunciation Assessment [demo](#).
- Watch the [video demo](#) of pronunciation assessment.

Last updated on 10/23/2025

Improve recognition accuracy with phrase list

A phrase list is a list of words or phrases provided ahead of time to help improve their recognition. Adding a phrase to a phrase list increases its importance, thus making it more likely to be recognized. You can add phrase list in real-time transcription and fast transcription.

Examples of phrases include:

- Names
- Geographical locations
- Homonyms
- Words or acronyms unique to your industry or organization

Phrase lists are simple and lightweight:

- **Just-in-time:** A phrase list is provided just before starting the speech recognition, eliminating the need to train a custom model.
- **Lightweight:** You don't need a large data set. Provide a word or phrase to boost its recognition.

For supported phrase list locales, see [Language and voice support for the Speech service](#).

You can use phrase lists with the [Speech Studio](#), [Speech SDK](#), or [Speech Command Line Interface \(CLI\)](#). It's supported with [Real-time transcription](#) and [Fast transcription API](#). The [Batch transcription API](#) doesn't support phrase lists.

You can use phrase lists with both standard and [custom speech](#). There are some situations where training a custom model that includes phrases is likely the best option to improve accuracy. For example, in the following cases you would use custom speech:

- If you need to use a large list of phrases. A phrase list shouldn't have more than 500 phrases.
- If you need a phrase list for languages that aren't currently supported.

Phrase list weight

When using the Speech SDK with Real-time transcription, you can control the weight of phrase list phrases relative to the default dictionary. This setting determines how much influence the phrase list has on speech-to-text results.

The phrase list weight can be set within a range of `0.0` to `2.0`:

- 0.0: Disables the phrase list
- 1.0: Default weight (standard influence)
- 2.0: Maximum weight (highest influence)

A higher weight increases the likelihood that phrases from your list are recognized over alternatives in the default dictionary. This setting applies to the complete list.

Try it in Speech Studio

You can use [Speech Studio](#) to test how phrase list would help improve recognition for your audio. To implement a phrase list with your application in production, you use the Speech SDK or Speech CLI.

For example, let's say that you want the Speech service to recognize this sentence: "Hi Rehaan, I'm Jessie from Contoso bank."

You might find that a phrase is incorrectly recognized as: "Hi **everyone**, I'm **Jesse** from **can't do so bank**."

In the previous scenario, you would want to add "Rehaan", "Jessie", and "Contoso" to your phrase list. Then the names should be recognized correctly.

Now try Speech Studio to see how phrase list can improve recognition accuracy.

(!) Note

You can be prompted to select your Azure subscription and Speech resource, and then acknowledge billing for your region.

1. Go to **Real-time Speech to text** in [Speech Studio](#).
2. You test speech recognition by uploading an audio file or recording audio with a microphone. For example, select **record audio with a microphone** and then say "Hi Rehaan, I'm Jessie from Contoso bank." Then select the red button to stop recording.
3. You should see the transcription result in the **Test results** text box. If "Rehaan", "Jessie", or "Contoso" were recognized incorrectly, you can add the terms to a phrase list in the next step.
4. Select **Show advanced options** and turn on **Phrase list**.
5. Enter "Contoso;Jessie;Rehaan" in the phrase list text box. Multiple phrases need to be separated by a semicolon.

The screenshot shows the Azure Speech-to-text configuration interface. At the top, there's a 'Choose a language' dropdown set to 'English (United States)' with an 'Auto detect' toggle switch. Below it is a 'Choose a custom endpoint' dropdown set to '[None]'. To the right is an 'Output format' dropdown set to 'Detailed'. Underneath these are two sections: 'Phrase list' (set to 'On') and a text input field containing 'Contoso;Jessie;Rehaan', which has a magnifying glass icon next to it.

6. Use the microphone to test recognition again. Otherwise you can select the retry arrow next to your audio file to rerun your audio. The terms "Rehaan", "Jessie", or "Contoso" should be recognized.

Implement phrase list in real-time transcription

With the [Speech SDK](#) you can add phrases individually and then run speech recognition.

C#

```
var phraseList = PhraseListGrammar.FromRecognizer(recognizer);
phraseList.AddPhrase("Contoso");
phraseList.AddPhrase("Jessie");
phraseList.AddPhrase("Rehaan");
phraselist.SetWeight(weight);
```

Allowed characters include locale-specific letters and digits, white space characters, and special characters such as +, -, \$, :, (,), {, }, _, ., ?, @, \, ', &, #, %, ^, *, `, <, >, ;, /. Other special characters are removed internally from the phrase.

Implement phrase list in fast transcription

You can add a list of phrases in fast transcription via [Speech-to-text REST API](#)

Azure CLI

```
curl --location
'https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/transcriptions:t
ranscribe?api-version=2025-10-15' \
--header 'Ocp-Apim-Subscription-Key: YourSpeechResourceKey' \
--form 'audio=@"YourAudioFile"' \
--form 'definition={
  "locales": ["en-US"],
  "phraseList": {
    "phrases": ["Contoso", "Jessie", "Rehaan"]}
```

}',

Next steps

Learn more about options to improve recognition accuracy.

[Custom speech](#)

Last updated on 11/05/2025

Display text formatting with speech to text

08/07/2025

Speech to text offers an array of formatting features to ensure that the transcribed text is clear and legible. See the sections below for an overview of how each feature is used to improve the overall clarity of the final text output.

ITN

Inverse Text Normalization (ITN) is a process that converts verbalized forms into their corresponding symbolic written forms. For example, the spoken word "four" is converted to the written form "4". The speech to text service completes this process and it's not configurable. Some of the supported text formats include dates, times, decimals, currencies, addresses, emails, and phone numbers. You can speak naturally, and the service formats text as expected. The following table shows the ITN rules that are applied to the text output.

[] Expand table

Recognized speech	Display text
that will cost nine hundred dollars	That will cost \$900.
my phone number is one eight hundred, four five six, eight nine ten	My phone number is 1-800-456-8910.
the time is six forty five p m	The time is 6:45 PM.
I live on thirty five lexington avenue	I live on 35 Lexington Ave.
the answer is six point five	The answer is 6.5.
send it to support at help dot com	Send it to support@help.com.

Capitalization

Speech to text models recognize words that should be capitalized to improve readability, accuracy, and grammar. For example, the Speech service automatically capitalizes proper nouns and words at the beginning of a sentence. Some examples are shown in this table.

[] Expand table

Recognized speech	Display text
i got an x l t shirt	I got an XL t-shirt.
my name is jennifer smith	My name is Jennifer Smith.
i want to visit new york city	I want to visit New York City.

Disfluency removal

When speaking, it's common for someone to stutter, duplicate words, and say filler words like "uhm" or "uh". Speech to text can recognize such disfluencies and remove them from the display text. Disfluency removal is great for transcribing live unscripted speeches to read them back later. Some examples are shown in this table.

[\[+\] Expand table](#)

Recognized speech	Display text
i uh said that we can go to the uhmm movies	I said that we can go to the movies.
its its not that big of uhm a deal	It's not that big of a deal.
umm i think tomorrow should work	I think tomorrow should work.

Punctuation

Speech to text automatically punctuates your text to improve clarity. Punctuation is helpful for reading back call or conversation transcriptions. Some examples are shown in this table.

[\[+\] Expand table](#)

Recognized speech	Display text
how are you	How are you?
we can go to the mall park or beach	We can go to the mall, park, or beach.

When you're using speech to text with continuous recognition, you can configure the Speech service to recognize explicit punctuation marks. Then you can speak punctuation aloud in order to make your text more legible. This is especially useful in a situation where you want to use complex punctuation without having to merge it later. Some examples are shown in this table.

Recognized speech	Display text
they entered the room dot dot dot	They entered the room...
i heart emoji you period	I <3 you.
the options are apple forward slash banana forward slash orange period	The options are apple/banana/orange.
are you sure question mark	Are you sure?

Use the Speech SDK to enable dictation mode when you're using speech to text with continuous recognition. This mode causes the speech configuration instance to interpret word descriptions of sentence structures such as punctuation.

```
speechConfig.EnableDictation();
```

Profanity filter

You can specify whether to mask, remove, or show profanity in the final transcribed text. Masking replaces profane words with asterisk (*) characters so that you can keep the original sentiment of your text while making it more appropriate for certain situations

 Note

Microsoft also reserves the right to mask or remove any word that is deemed inappropriate. Such words will not be returned by the Speech service, whether or not you enabled profanity filtering.

The profanity filter options are:

- **Masked**: Replaces letters in profane words with asterisk (*) characters. Masked is the default option.
- **Raw**: Include the profane words verbatim.
- **Removed**: Removes profane words.

For example, to remove profane words from the speech recognition result, set the profanity filter to **Removed** as shown here:

```
speechConfig.SetProfanity(ProfanityOption.Removed);
```

Profanity filter is applied to the result `Text` and `MaskedNormalizedForm` properties. Profanity filter isn't applied to the result `LexicalForm` and `NormalizedForm` properties. Neither is the filter applied to the word level results.

Related content

- [Speech to text quickstart](#)
- [Get speech recognition results](#)

What is the Whisper model?

The Whisper model is a speech to text model from OpenAI that you can use to transcribe or translate audio files. The model is trained on a large dataset of English audio and text.

- The model is optimized for transcribing audio files that contain speech in English.
- The model can also be used to translate audio files that contain speech in other languages. The output of the transcription is English text.

Whisper models are available via the Azure OpenAI in Azure AI Foundry Models or via Azure AI Speech. The features differ for those offerings. In [Azure AI Speech \(batch transcription\)](#), Whisper is just one of several models that you can use for speech to text.

You might ask:

- Is the Whisper Model a good choice for my scenario, or is an Azure AI Speech model better? What are the API comparisons between the two types of models?
- If I want to use the Whisper Model, should I use it via the Azure OpenAI or via Azure AI Speech ? What are the scenarios that guide me to use one or the other?

Whisper model or Azure AI Speech models

Either the Whisper model or the Azure AI Speech models are appropriate depending on your scenarios. If you decide to use Azure AI Speech, you can choose from several models, including the Whisper model. The following table compares options with recommendations about where to start.

[] Expand table

Scenario	Whisper model	Azure AI Speech models
Real-time transcriptions, captions, and subtitles for audio and video.	Not available	Recommended
Transcriptions, captions, and subtitles for prerecorded audio and video.	The Whisper model via Azure OpenAI is recommended for fast processing of individual audio files. The Whisper model via Azure AI Speech (batch transcription) is recommended for batch processing of large files. For more information, see Whisper model via Azure AI Speech batch transcription or via Azure OpenAI?	Recommended for batch processing of large files, diarization, and word level timestamps.

Scenario	Whisper model	Azure AI Speech models
Transcript of phone call recordings and analytics such as call summary, sentiment, key topics, and custom insights.	Available	Recommended
Real-time transcription and analytics to assist call center agents with customer questions.	Not available	Recommended
Transcript of meeting recordings and analytics such as meeting summary, meeting chapters, and action item extraction.	Available	Recommended
Real-time text entry and document generation through voice dictation.	Not available	Recommended
Contact center voice agent: Call routing and interactive voice response for call centers.	Available	Recommended
Voice assistant: Application specific voice assistant for a set-top box, mobile app, in-car, and other scenarios.	Available	Recommended
Pronunciation assessment: Assess the pronunciation of a speaker's voice.	Not available	Recommended
Translate live audio from one language to another.	Not available	Recommended via the speech translation API .
Translate prerecorded audio from other languages into English.	Recommended	Also available via the speech translation API .
Translate prerecorded audio into languages other than English.	Not available	Recommended via the speech translation API .

Whisper model via Azure AI Speech or via Azure OpenAI?

If you decide to use the Whisper model, you have two options. You can choose whether to use the Whisper Model via [Azure OpenAI](#) or via [Azure AI Speech \(batch transcription\)](#). In either case, the readability of the transcribed text is the same.

Whisper Model via Azure OpenAI might be best for:

- Quickly transcribing audio files one at a time.
- Translate audio from other languages into English. You can input mixed language audio and the output is in English.
- Provide a prompt to the model to guide the output.
- Supported file formats: mp3, mp4, mpweg, mpg, m4a, wav, and webm.
- Only ASCII character supported for filename.

Whisper Model via Azure AI Speech batch transcription might be best for:

- Transcribing files larger than 25MB (up to 1GB). The file size limit for the Azure OpenAI Whisper model is 25 MB.
- Transcribing large batches of audio files.
- Diarization to distinguish between the different speakers participating in the conversation. The Speech service provides information about which speaker was speaking a particular part of transcribed speech. The Whisper model via Azure OpenAI doesn't support diarization.
- Word-level timestamps
- Supported file formats: mp3, wav, and ogg.

Regional support is another consideration.

- For the current list of regions where the Whisper model is available, see the [Speech service regions table](#).

Related content

- [Use Whisper models via the Azure AI Speech batch transcription API](#)
- [Try the speech to text quickstart for Whisper via Azure OpenAI](#)
- [Try the real-time speech to text quickstart via Azure AI Speech](#)

Speech to text FAQ

This article answers commonly asked questions about the speech to text capability. If you can't find answers to your questions here, check out [other support options](#).

General

What is the difference between a base model and a custom speech to text model?

A baseline speech to text model is trained with Microsoft-owned data and is already deployed in the cloud. You can create and use a custom model to better fit an environment that has specific ambient noise or language. Factory floors, cars, or noisy streets would require an adapted acoustic model. Topics such as biology, physics, radiology, product names, and custom acronyms would require an adapted language model. If you want to train a custom model, you should start with related text to improve the recognition of special terms and phrases.

Where do I start if I want to use a base model?

First, get an API key and region in the [Azure portal](#). If you want to make REST calls to a predeployed base model, see the [REST APIs](#) documentation. If you want to use WebSockets, [download the Speech SDK](#).

Do I always need to build a custom speech model?

No. If your application uses generic, day-to-day language, you don't need to customize a model. If your application is used in an environment where there's little or no background noise, you don't need to customize a model.

You can deploy baseline and customized models in the portal and then run accuracy tests against them. You can use this feature to measure the accuracy of a base model versus a custom model.

How do I know when the processing for my dataset or model is complete?

Currently, the only way to know is to view the status of the model or dataset in the table. When the processing is complete, the status is *Succeeded*.

Can I create more than one model?

There's no limit to the number of models you can have in your collection.

I realized that I made a mistake. How do I cancel a data import or model creation that's in progress?

Currently, you can't roll back an acoustic or language adaptation process. You can delete imported data and models when they're in a terminal state.

I get several results for each phrase with the detailed output format. Which one should I use?

Always take the first result, even if another result ("N-Best") might have a higher confidence value. Speech service considers the first result to be the best. The result can also be an empty string if no speech was recognized.

The other results are likely worse and might not have full capitalization and punctuation applied. These results are most useful in special scenarios, such as giving users the option to pick corrections from a list or handling incorrectly recognized commands.

Why are there multiple base models?

You can choose from more than one base model in Speech service. Each model name contains the date when it was added. When you start training a custom model, use the most recent model to get the best accuracy. Older base models are still available for some time after a new model is made available. You can continue using the model that you worked with until it's retired (see [Model and endpoint lifecycle](#)). We still recommend that you switch to the latest base model for better accuracy.

Can I update my existing model (model stacking)?

You can't update an existing model. As a solution, combine the old dataset with the new dataset and readapt.

The old dataset and the new dataset must be combined in a single .zip file (for acoustic data) or in a .txt file (for language data). When the adaptation is finished, redeploy the new, updated model to obtain a new endpoint.

When a new version of a base model is available, is my deployment automatically updated?

Deployments are *not* automatically updated.

If you adapted and deployed a model, the existing deployment remains as is. You can decommission the deployed model, readapt it by using the newer version of the base model, and redeploy it for better accuracy.

Both base models and custom models are retired after some time (see [Model and endpoint lifecycle](#)).

Can I download my model and run it locally?

You can run a custom model locally in a [Docker container](#).

Can I copy or move my datasets, models, and deployments to another region or subscription?

You can use the [Models_Copy REST API](#) to copy a custom model to another region or subscription. Datasets and deployments can't be copied. You can import a dataset again in another subscription and create endpoints there by using the model copies.

Are my requests logged?

By default, requests aren't logged (neither audio nor transcription). If necessary, you can select the **Log content from this endpoint** option when you [create a custom endpoint](#). You can also enable audio logging in the [Speech SDK](#) on a per-request basis, without having to create a custom endpoint. In both cases, audio and recognition results of requests will be stored in secure storage. Subscriptions that use Microsoft-owned storage are available for 30 days.

You can export the logged files on the deployment page in Speech Studio if you use a custom endpoint with **Log content from this endpoint** enabled. If audio logging is enabled via the SDK, call the API to access the files. You can also use API to [delete the logs](#) anytime.

Are my requests throttled?

For information, see [Speech service quotas and limits](#).

How am I charged for dual channel audio?

If you submit each channel separately in their own file, you're charged for the audio duration of each file. If you submit a single file with the channels multiplexed together, you're charged for the duration of the single file. For more information about pricing, see the [Azure AI services pricing page](#).

 **Important**

If you have further privacy concerns that prevent you from using the custom speech service, contact one of the support channels.

Increasing concurrency

For information, see [Speech service quotas and limits](#).

Importing data

What is the limit to the size of a dataset, and why is it the limit?

The limit is because of the restriction on the size of files for HTTP upload. For the actual limit, see [Speech service quotas and limits](#). You can split your data into multiple datasets and select all of them to train the model.

Can I zip (compress) my text files so that I can upload a larger text file?

No. Currently, only uncompressed text files are allowed.

The data report says there were failed utterances. What is the issue?

A failure to upload 100 percent of the utterances in a file isn't a problem. If most of the utterances in an acoustic or language dataset (for example, more than 95 percent) are successfully imported, the dataset can be usable. However, we still recommend that you try to understand why the utterances failed and then fix the problem. Most common problems, such as formatting errors, are easy to fix.

Creating an acoustic model

How much acoustic data do I need?

We recommend starting with from 30 minutes to 1 hour of acoustic data.

What data should I collect?

Collect data that's as close to the application scenario and use case as possible. The data collection should match the target application and users in terms of device or devices, environments, and types of speakers. In general, you should collect data from as broad a range of speakers as possible.

How should I collect acoustic data?

You can create a standalone data collection application or use off-the-shelf audio recording software. You can also create a version of your application that logs the audio data and then uses the data.

Do I need to transcribe adaptation data myself?

Yes. You can transcribe it yourself or use a professional transcription service. Some users prefer professional transcribers, and others use crowdsourcing or transcribe the data themselves.

How long does it take to train a custom model with audio data?

Training a model with audio data can be a lengthy process. Depending on the amount of data, it can take several days to create a custom model. If it can't be finished within one week, the service might abort the training operation and report the model as failed.

In general, Speech service processes approximately 10 hours of audio data per day in regions that have dedicated hardware. Training with text only is faster and ordinarily finishes within minutes.

Use one of the regions where dedicated hardware is available for training. The Speech service uses up to 100 hours of audio for training in these regions.

Accuracy testing

What is word error rate (WER), and how is it computed?

WER is the evaluation metric for speech recognition. WER is calculated as the total number of errors (insertions, deletions, and substitutions), divided by the total number of words in the reference transcription. For more information, see [Test model quantitatively](#).

How do I determine whether the results of an accuracy test are good?

The results show a comparison between the base model and the model you customized. To make customization worthwhile, you should aim to beat the base model.

How do I determine the WER of a base model so I can see whether it improved?

The offline test results show the baseline accuracy of the custom model and the improvement over baseline.

Creating a language model

How much text data do I need to upload?

It depends on how different the vocabulary and phrases used in your application are from the starting language models. For all new words, it's useful to provide as many examples as possible of the usage of those words. For common phrases that are used in your application, including phrases in the language data, providing many examples is useful because it tells the system to listen for these terms also. It's common to have at least 100 and, ordinarily, several hundred or more utterances in the language dataset. Also, if some types of queries are expected to be more common than others, you can insert multiple copies of the common queries in the dataset.

Can I simply upload a list of words?

Uploading a list of words adds them to the vocabulary, but it doesn't teach the system how the words are ordinarily used. By providing full or partial utterances (sentences or phrases of things that users are likely to say), the language model can learn the new words and how they're used. The custom language model is good not only for adding new words to the system, but also for

adjusting the likelihood of known words for your application. Providing full utterances helps the system learn better.

Pronunciation assessment

Why does the recognized text differ from the reference text?

The recognized text is generated based on the audio input, the reference text, the `EnableMispell` configuration and the assessment mode.

In **scripted assessment**, there are two modes, single-shot and continuous, and the behavior differs slightly. In single-shot mode, if `EnableMispell` is set to `false`, the system forces the recognized text to match the reference text. When `EnableMispell` is `true`, only the words present in the reference text are considered as recognized results from the audio input. Continuous mode does not support the `EnableMispell` option and behaves similarly to single-shot mode with `EnableMispell` set to `true`. Differences between recognized and reference text might occur due to factors such as pronunciation variations, background noise, or limitations in the speech recognition model.

In **unscripted assessment**, the recognized text is generated solely from the audio input without any reference text, which can lead to discrepancies between the recognized text and the intended content. In these cases, the recognized text reflects what the system interprets from the audio and may not always align with the expected message. If you notice significant differences, review the audio quality and speaker clarity, or consider using Azure Speech-to-Text to transcribe the audio first. You can then use that transcription as the reference text for a more accurate assessment.

Next steps

- [Speech to text quickstart](#)
- [What's new](#)

Text to speech documentation

Text to speech from the Speech service enables your applications, tools, or devices to convert text into human-like synthesized speech.

About text to speech

OVERVIEW

[What is text to speech?](#)

[Use the Speech CLI for text to speech with no code](#)

QUICKSTART

[Get started with text to speech](#)

Develop with text to speech

HOW-TO GUIDE

[Batch synthesis for long-form text](#)

[Get started with custom voice](#)

[Create and use custom voice models](#)

[Create audio content in Speech Studio](#)

CONCEPT

[What is custom voice?](#)

[Improve synthesis with SSML](#)

Reference

REFERENCE

[Neural voice support](#)

[SSML phonetic sets](#)

[Text to speech pricing ↗](#)

Help and feedback



[Support and help options](#)

What is text to speech?

08/07/2025

In this overview, you learn about the benefits and capabilities of the text to speech feature of the Speech service, which is part of Azure AI services.

Text to speech enables your applications, tools, or devices to convert text into human-like synthesized speech. The text to speech capability is also known as speech synthesis. Use human-like standard voices out of the box, or create a custom voice that's unique to your product or brand. For a full list of supported voices, languages, and locales, see [Language and voice support for the Speech service](#).

Core features

Text to speech includes the following features:

 Expand table

Feature	Summary	Demo
Standard voice (called <i>Neural</i> on the pricing page)	Highly natural out-of-the-box voices. Create an Azure subscription and Speech resource, and then use the Speech SDK or visit the Speech Studio portal and select standard voices to get started. Check the pricing details .	Check the Voice Gallery and determine the right voice for your business needs.
Custom voice	Easy-to-use self-service for creating a natural brand voice, with limited access for responsible use. Create an Azure subscription and Azure AI Foundry resource and then apply to use custom voice . After you're granted access, go to the professional voice fine-tuning documentation to get started. Check the pricing details .	Check the voice samples .

More about neural text to speech features

Text to speech uses deep neural networks to make the voices of computers nearly indistinguishable from the recordings of people. With the clear articulation of words, neural text to speech significantly reduces listening fatigue when users interact with AI systems.

The patterns of stress and intonation in spoken language are called *prosody*. Traditional text to speech systems break down prosody into separate linguistic analysis and acoustic prediction steps governed by independent models. That can result in muffled, buzzy voice synthesis.

Here's more information about neural text to speech features in the Speech service, and how they overcome the limits of traditional text to speech systems:

- **Real-time speech synthesis:** Use the [Speech SDK](#) or [REST API](#) to convert text to speech by using [standard voices](#) or [custom voices](#).
- **Asynchronous synthesis of long audio:** Use the [batch synthesis API](#) to asynchronously synthesize text to speech files longer than 10 minutes (for example, audio books or lectures). Unlike synthesis performed via the Speech SDK or Speech to text REST API, responses aren't returned in real-time. The expectation is that requests are sent asynchronously, responses are polled for, and synthesized audio is downloaded when the service makes it available.
- **Standard voices:** Azure AI Speech uses deep neural networks to overcome the limits of traditional speech synthesis regarding stress and intonation in spoken language. Prosody prediction and voice synthesis happen simultaneously, which results in more fluid and natural-sounding outputs. Each standard voice model is available at 24 kHz and high-fidelity 48 kHz. You can use neural voices to:
 - Make interactions with chatbots and voice assistants more natural and engaging.
 - Convert digital texts such as e-books into audiobooks.
 - Enhance in-car navigation systems.

For a full list of standard Azure AI Speech neural voices, see [Language and voice support for the Speech service](#).

- **Improve text to speech output with SSML:** Speech Synthesis Markup Language (SSML) is an XML-based markup language used to customize text to speech outputs. With SSML, you can adjust pitch, add pauses, improve pronunciation, change speaking rate, adjust volume, and attribute multiple voices to a single document.

You can use SSML to define your own lexicons or switch to different speaking styles. With the [multilingual voices](#), you can also adjust the speaking languages via SSML. To improve the voice output for your scenario, see [Improve synthesis with Speech Synthesis Markup Language](#) and [Speech synthesis with the Audio Content Creation tool](#).

- **Visemes:** Visemes are the key poses in observed speech, including the position of the lips, jaw, and tongue in producing a particular phoneme. Visemes have a strong correlation with voices and phonemes.

By using viseme events in Speech SDK, you can generate facial animation data. This data can be used to animate faces in lip-reading communication, education, entertainment, and customer service. Viseme is currently supported only for the `en-US` (US English) [neural voices](#).

Note

In addition to Azure AI Speech neural (non HD) voices, you can also use [Azure AI Speech high definition \(HD\) voices](#) and [Azure OpenAI neural \(HD and non HD\) voices](#). The HD voices provide a higher quality for more versatile scenarios.

Some voices don't support all [Speech Synthesis Markup Language \(SSML\)](#) tags. This includes neural text to speech HD voices, personal voices, and embedded voices.

- For Azure AI Speech high definition (HD) voices, check the SSML support [here](#).
- For personal voice, you can find the SSML support [here](#).
- For embedded voices, check the SSML support [here](#).

Get started

To get started with text to speech, see the [quickstart](#). Text to speech is available via the [Speech SDK](#), the [REST API](#), and the [Speech CLI](#).

Tip

To convert text to speech with a no-code approach, try the [Audio Content Creation](#) tool in [Speech Studio](#) ↗.

Sample code

Sample code for text to speech is available on GitHub. These samples cover text to speech conversion in most popular programming languages:

- [Text to speech samples \(SDK\)](#) ↗
- [Text to speech samples \(REST\)](#) ↗

Custom voice

In addition to standard voices, you can create custom voices that are unique to your product or brand. Custom voice is an umbrella term that includes professional voice fine-tuning and personal voice. All it takes to get started is a handful of audio files and the associated transcriptions. For more information, see the [professional voice fine-tuning documentation](#).

Pricing note

Billable characters

When you use the text to speech feature, you're billed for each character that's converted to speech, including punctuation. Although the SSML document itself isn't billable, optional elements that are used to adjust how the text is converted to speech, like phonemes and pitch, are counted as billable characters. Here's a list of what's billable:

- Text passed to the text to speech feature in the SSML body of the request
- All markup within the text field of the request body in the SSML format, except for `<speak>` and `<voice>` tags
- Letters, punctuation, spaces, tabs, markup, and all white-space characters
- Every code point defined in Unicode

For detailed information, see [Speech service pricing ↗](#).

Important

Each Chinese character is counted as two characters for billing, including kanji used in Japanese, hanja used in Korean, or hanzi used in other languages.

Model training and hosting time for custom voice

Custom voice training and hosting are both calculated by hour and billed per second. For the billing unit price, see [Speech service pricing ↗](#).

Professional voice fine-tuning time is measured by "compute hour" (a unit to measure machine running time). Typically, when training a voice model, two computing tasks are running in parallel. So, the calculated compute hours are longer than the actual training time. For professional voice fine-tuning, it usually takes 20 to 40 compute hours to train a single-style voice, and around 90 compute hours to train a multi-style voice. The professional voice fine-tuning time is billed with a cap of 96 compute hours. So in the case that a voice model is trained in 98 compute hours, you'll only be charged with 96 compute hours.

Custom voice endpoint hosting is measured by the actual time (hour). The hosting time (hours) for each endpoint is calculated at 00:00 UTC every day for the previous 24 hours. For example, if the endpoint has been active for 24 hours on day one, it's billed for 24 hours at 00:00 UTC the second day. If the endpoint is newly created or suspended during the day, it's billed for its accumulated running time until 00:00 UTC the second day. If the endpoint isn't currently hosted, it isn't billed. In addition to the daily calculation at 00:00 UTC each day, the billing is

also triggered immediately when an endpoint is deleted or suspended. For example, for an endpoint created at 08:00 UTC on December 1, the hosting hour will be calculated to 16 hours at 00:00 UTC on December 2 and 24 hours at 00:00 UTC on December 3. If the user suspends hosting the endpoint at 16:30 UTC on December 3, the duration (16.5 hours) from 00:00 to 16:30 UTC on December 3 will be calculated for billing.

Personal voice

When you use the personal voice feature, you're billed for both profile storage and synthesis.

- **Profile storage:** After a personal voice profile is created, it will be billed until it's removed from the system. The billing unit is per voice per day. If voice storage lasts for less than 24 hours, it's still billed as one full day.
- **Synthesis:** Billed per character. For details on billable characters, see the above [billable characters](#).

Text to speech avatar

When you use the text-to-speech avatar feature, charges are billed per second based on the length of video output. However, for the real-time avatar, charges are billed per second based on the time when the avatar is active, regardless of whether it's speaking or remaining silent. To optimize costs for real-time avatar usage, refer to the "Use Local Video for Idle" tips provided in the [avatar chat sample code ↗](#).

Custom text to speech avatar training time is measured by "compute hour" (machine running time) and billed per second. Training duration varies depending on how much data you use. It normally takes 20-40 compute hours on average to train a custom avatar. The avatar training time is billed with a cap of 96 compute hours. So in the case that an avatar model is trained in 98 compute hours, you're only charged for 96 compute hours.

Avatar hosting is billed per second per endpoint. You can suspend your endpoint to save costs. If you want to suspend your endpoint, you can delete it directly. To use it again, redeploy the endpoint.

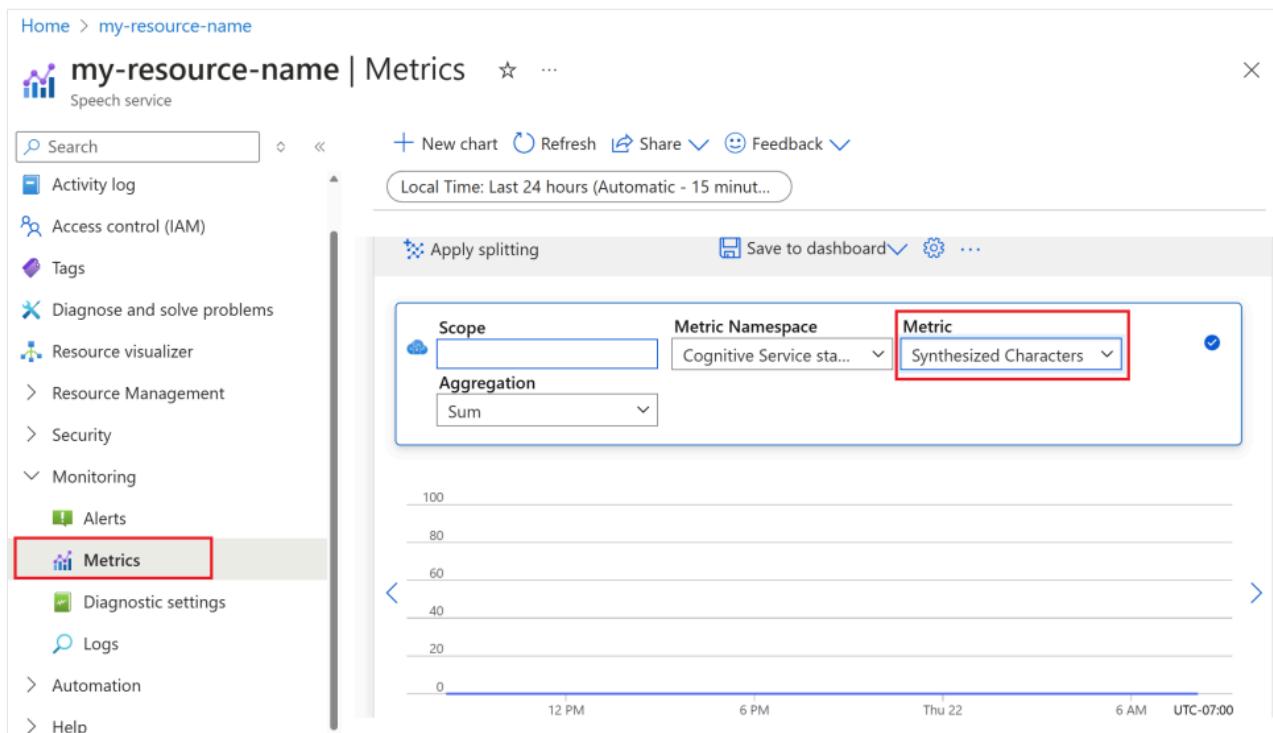
Monitor Azure text to speech metrics

Monitoring key metrics associated with text to speech services is crucial for managing resource usage and controlling costs. This section guides you on how to find usage information in the Azure portal and provide detailed definitions of the key metrics. For more information on Azure monitor metrics, see [Azure Monitor Metrics overview](#).

How to find usage information in the Azure portal

To effectively manage your Azure resources, it's essential to access and review usage information regularly. Here's how to find the usage information:

1. Go to the [Azure portal](#) and sign in with your Azure account.
2. Navigate to **Resources** and select your resource you wish to monitor.
3. Select **Metrics** under **Monitoring** from the left-hand menu.



4. Customize metric views.

You can filter data by resource type, metric type, time range, and other parameters to create custom views that align with your monitoring needs. Additionally, you can save the metric view to dashboards by selecting **Save to dashboard** for easy access to frequently used metrics.

5. Set up alerts.

To manage usage more effectively, set up alerts by navigating to the **Alerts** tab under **Monitoring** from the left-hand menu. Alerts can notify you when your usage reaches specific thresholds, helping to prevent unexpected costs.

Definition of metrics

Here's a table summarizing the key metrics for Azure text to speech.

Metric name	Description
Synthesized Characters	Tracks the number of characters converted into speech, including standard voice and custom voice. For details on billable characters, see Billable characters .
Video Seconds Synthesized	Measures the total duration of video synthesized, including batch avatar synthesis, real-time avatar synthesis, and custom avatar synthesis.
Avatar Model Hosting Seconds	Tracks the total time in seconds that your custom avatar model is hosted.
Voice Model Hosting Hours	Tracks the total time in hours that your custom voice model is hosted.
Voice Model Training Minutes	Measures the total time in minutes for training your custom voice model.

Reference docs

- [Speech SDK](#)
- [REST API: Text to speech](#)

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who are affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

- [Transparency note and use cases for custom voice](#)
- [Characteristics and limitations for using custom voice](#)
- [Limited access to custom voice](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)
- [Disclosure for voice talent](#)
- [Disclosure design guidelines](#)
- [Disclosure design patterns](#)
- [Code of Conduct for Text to speech integrations](#)
- [Data, privacy, and security for custom voice](#)

Next steps

- [Text to speech quickstart](#)

- Get the Speech SDK

Quickstart: Convert text to speech

07/17/2025

[Reference documentation](#) | [Package \(NuGet\)](#) | [Additional samples on GitHub](#)

With Azure AI Speech, you can run an application that synthesizes a human-like voice to read text. You can change the voice, enter text to be spoken, and listen to the output on your computer's speaker.

💡 Tip

You can try text to speech in the [Speech Studio Voice Gallery](#) without signing up or writing any code.

💡 Tip

Try out the [Azure AI Speech Toolkit](#) to easily build and run samples on Visual Studio Code.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#).
- ✓ [Create an AI Services resource for Speech](#) in the Azure portal.
- ✓ Get the Speech resource key and endpoint. After your Speech resource is deployed, select [Go to resource](#) to view and manage keys.

Set up the environment

The Speech SDK is available as a [NuGet package](#) that implements .NET Standard 2.0. Install the Speech SDK later in this guide by using the console. For detailed installation instructions, see [Install the Speech SDK](#).

Set environment variables

You need to authenticate your application to access Azure AI services. This article shows you how to use environment variables to store your credentials. You can then access the environment variables from your code to authenticate your application. For production, use a more secure way to store and access your credentials.

Important

We recommend Microsoft Entra ID authentication with [managed identities for Azure resources](#) to avoid storing credentials with your applications that run in the cloud.

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If using API keys, store them securely in Azure Key Vault, rotate the keys regularly, and restrict access to Azure Key Vault using role based access control and network access restrictions. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

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

- To set the `SPEECH_KEY` environment variable, replace *your-key* with one of the keys for your resource.
- To set the `ENDPOINT` environment variable, replace *your-endpoint* with one of the endpoints for your resource.

Windows

Console

```
setx SPEECH_KEY your-key
setx ENDPOINT your-endpoint
```

Note

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

After you add the environment variables, you might need to restart any programs that need to read the environment variables, including the console window. For example, if you're using Visual Studio as your editor, restart Visual Studio before you run the example.

Create the application

Follow these steps to create a console application and install the Speech SDK.

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

```
C#
```

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

class Program
{
    // This example requires environment variables named "SPEECH_KEY" and
    "END_POINT"
    static string speechKey =
Environment.GetEnvironmentVariable("SPEECH_KEY");
    static string endpoint = Environment.GetEnvironmentVariable("END_POINT");

    static void OutputSpeechSynthesisResult(SpeechSynthesisResult
speechSynthesisResult, string text)
    {
        switch (speechSynthesisResult.Reason)
        {
            case ResultReason.SynthesizingAudioCompleted:
                Console.WriteLine($"Speech synthesized for text: [{text}]");
                break;
            case ResultReason.Canceled:
                var cancellation =
SpeechSynthesisCancellationDetails.FromResult(speechSynthesisResult);
                Console.WriteLine($"CANCELED: Reason={cancellation.Reason}");

                if (cancellation.Reason == CancellationReason.Error)
                {
```

```

                Console.WriteLine($"CANCELED: ErrorCode=
{cancellation.ErrorCode}");
                Console.WriteLine($"CANCELED: ErrorDetails=
[{cancellation.ErrorDetails}]);");
                Console.WriteLine($"CANCELED: Did you set the speech
resource key and endpoint values?");}
            }
            break;
        default:
            break;
    }
}

async static Task Main(string[] args)
{
    var speechConfig = SpeechConfig.FromEndpoint(speechKey, endpoint);

        // The neural multilingual voice can speak different languages based
on the input text.
    speechConfig.SpeechSynthesisVoiceName = "en-US-
AvaMultilingualNeural";

    using (var speechSynthesizer = new SpeechSynthesizer(speechConfig))
    {
        // Get text from the console and synthesize to the default
speaker.
        Console.WriteLine("Enter some text that you want to speak >"); string text = Console.ReadLine();

        var speechSynthesisResult = await
speechSynthesizer.SpeakTextAsync(text);
        OutputSpeechSynthesisResult(speechSynthesisResult, text);
    }

    Console.WriteLine("Press any key to exit..."); Console.ReadKey();
}
}

```

4. To change the speech synthesis language, replace `en-US-AvaMultilingualNeural` with another [supported voice](#).

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 set `es-ES-ElviraNeural` as the language, 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 output synthesized audio.

5. Run your new console application to start speech synthesis to the default speaker.

Console

```
dotnet run
```

ⓘ Important

Make sure that you set the `SPEECH_KEY` and `END_POINT` [environment variables](#). If you don't set these variables, the sample fails with an error message.

6. Enter some text that you want to speak. For example, type *I'm excited to try text to speech*. Select the **Enter** key to hear the synthesized speech.

Console

```
Enter some text that you want to speak >
I'm excited to try text to speech
```

Remarks

More speech synthesis options

This quickstart uses the `SpeakTextAsync` operation to synthesize a short block of text that you enter. You can also use long-form text from a file and get finer control over voice styles, prosody, and other settings.

- See [how to synthesize speech](#) and [Speech Synthesis Markup Language \(SSML\) overview](#) for information about speech synthesis from a file and finer control over voice styles, prosody, and other settings.
- See [batch synthesis API for text to speech](#) for information about synthesizing long-form text to speech.

OpenAI text to speech voices in Azure AI Speech

OpenAI text to speech voices are also supported. See [OpenAI text to speech voices in Azure AI Speech](#) and [multilingual voices](#). You can replace `en-US-AvaMultilingualNeural` with a supported OpenAI voice name such as `en-US-FableMultilingualNeural`.

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Speech resource you created.

Next step

[Learn more about speech synthesis](#)

How to synthesize speech from text

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) | [Additional samples on GitHub](#)

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. The following code snippet shows how this technique works:

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = 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](#).

 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

<code>SpeechSynthesisVoiceName</code>	<code>SpeechSynthesisLanguage</code>	<code>SSML</code>	<code>Outcome</code>
X	X	X	Default voice for <code>en-US</code> speaks
X	✓	X	Default voice for specified locale speaks.
✓	✓	X	The voice that you specify by using <code>SpeechSynthesisVoiceName</code> speaks.
✓	✓	✓	The voice that you specify by using SSML speaks.

Synthesize speech to a file

Create a `SpeechSynthesizer` object. This object shown in the following snippets runs text to speech conversions and outputs to speakers, files, or other output streams. `SpeechSynthesizer` accepts as parameters:

- The `SpeechConfig` object that you created in the previous step.
- An `AudioConfig` object that specifies how output results should be handled.

1. Create an `AudioConfig` instance to automatically write the output to a `.wav` file by using the `FromWavFileOutput()` function. Instantiate it with a `using` statement.

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
"YourSpeechRegion");
    using var audioConfig =
AudioConfig.FromWavFileOutput("path/to/write/file.wav");
}
```

A `using` statement in this context automatically disposes of unmanaged resources and causes the object to go out of scope after disposal.

2. Instantiate a `SpeechSynthesizer` instance with another `using` statement. Pass your `speechConfig` object and the `audioConfig` object as parameters. To synthesize speech and write to a file, run `SpeakTextAsync()` with a string of text.

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
"YourSpeechRegion");
    using var audioConfig =
AudioConfig.FromWavFileOutput("path/to/write/file.wav");
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig,
audioConfig);
    await speechSynthesizer.SpeakTextAsync("I'm excited to try text to speech");
}
```

When you run the program, it creates a synthesized .wav file, which is written to the location that you specify. This result is a good example of the most basic usage. Next, you can customize output and handle the output response as an in-memory stream for working with custom scenarios.

Synthesize to speaker output

To output synthesized speech to the current active output device such as a speaker, omit the `AudioConfig` parameter when you're creating the `SpeechSynthesizer` instance. Here's an example:

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
"YourSpeechRegion");
```

```
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig);
    await speechSynthesizer.SpeakTextAsync("I'm excited to try text to speech");
}
```

Get a result as an in-memory stream

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. First, remove the `AudioConfig` block, because you manage the output behavior manually from this point onward for increased control. Pass `null` for `AudioConfig` in the `SpeechSynthesizer` constructor.

! Note

Passing `null` for `AudioConfig`, rather than omitting it as 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 `AudioData` property contains a `byte []` instance for the output data. You can work with this `byte []` instance manually, or you can use the `AudioDataStream` class to manage the in-memory stream.

In this example, you use the `AudioDataStream.FromResult()` static function to get a stream from the result:

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
"YourSpeechRegion");
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig, null);

    var result = await speechSynthesizer.SpeakTextAsync("I'm excited to try text
to speech");
    using var stream = AudioDataStream.FromResult(result);
}
```

At this point, you can implement any custom behavior by using the resulting `stream` object.

Customize audio format

You can customize audio output attributes, including:

- Audio file type
- Sample rate
- Bit depth

To change the audio format, you use the `SetSpeechSynthesisOutputFormat()` function on the `SpeechConfig` object. This function 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, you use `AudioDataStream` to get an in-memory stream of the result, and then write it to a file.

C#

```
static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
    "YourSpeechRegion");

    speechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Riff24Khz1
    6BitMonoPcm);

    using var speechSynthesizer = new SpeechSynthesizer(speechConfig, null);
    var result = await speechSynthesizer.SpeakTextAsync("I'm excited to try text
    to speech");

    using var stream = AudioDataStream.FromResult(result);
    await stream.SaveToWaveFileAsync("path/to/write/file.wav");
}
```

When you run the program, it writes a `.wav` file to the specified path.

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

<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, the file is *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. First, load your SSML configuration as a string by using `File.ReadAllText()`. From this point, the result object is exactly the same as previous examples.

ⓘ Note

If you're using Visual Studio, your build configuration likely won't find your XML file by default. Right-click the XML file and select **Properties**. Change **Build Action** to **Content**. Change **Copy to Output Directory** to **Copy always**.

C#

```
public static async Task SynthesizeAudioAsync()
{
    var speechConfig = SpeechConfig.FromSubscription("YourSpeechKey",
    "YourSpeechRegion");
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig, null);

    var ssml = File.ReadAllText("./ssml.xml");
```

```

        var result = await speechSynthesizer.SpeakSsmlAsync(ssml);

        using var stream = AudioDataStream.FromResult(result);
        await stream.SaveToWaveFileAsync("path/to/write/file.wav");
    }

```

 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
<code>BookmarkReached</code>	Signals that a bookmark was reached. To trigger a bookmark reached event, a <code>bookmark</code> element is required in the SSML . This event reports the output audio's elapsed time between the beginning of synthesis and the <code>bookmark</code> element. The event's <code>Text</code> property is the string value that you set in the bookmark's <code>mark</code> attribute. The <code>bookmark</code> elements aren't spoken.	You can use the <code>bookmark</code> element to insert custom markers in SSML to get the offset of each marker in the audio stream. The <code>bookmark</code> element can be used to reference a specific location in the text or tag sequence.
<code>SynthesisCanceled</code>	Signals that the speech synthesis was canceled.	You can confirm when synthesis is canceled.
<code>SynthesisCompleted</code>	Signals that speech synthesis is complete.	You can confirm when synthesis is complete.
<code>SynthesisStarted</code>	Signals that speech synthesis started.	You can confirm when synthesis started.

Event	Description	Use case
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.

(i) Important

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If you use an API key, store it securely in Azure Key Vault. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

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 *Program.cs* file with the following C# code:

C#

```
using Microsoft.CognitiveServices.Speech;

class Program
{
    // This example requires environment variables named "SPEECH_KEY" and
    "SPEECH_REGION"
    static string speechKey = Environment.GetEnvironmentVariable("SPEECH_KEY");
    static string speechRegion =
Environment.GetEnvironmentVariable("SPEECH_REGION");

    async static Task Main(string[] args)
    {
        var speechConfig = SpeechConfig.FromSubscription(speechKey, speechRegion);

        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'>
    <voice name='{speechSynthesisVoiceName}'>
        <mstts:viseme type='redlips_front'/>
        The rainbow has seven colors: <bookmark
mark='colors_list_begin'>Red, orange, yellow, green, blue, indigo, and violet.
<bookmark mark='colors_list_end'>.
    </voice>
</speak>";

        // Required for sentence-level WordBoundary events

speechConfig SetProperty(PropertyId.SpeechServiceResponse_RequestSentenceBoundary,
"true");

        using (var speechSynthesizer = new SpeechSynthesizer(speechConfig))
        {
            // Subscribe to events

            speechSynthesizer.BookmarkReached += (s, e) =>
            {
                Console.WriteLine($"BookmarkReached event: " +
                    $"{e.AudioOffset + 5000} / 10000ms" +
                    $"\\r\\n\\tText: \\"{e.Text}\\\"");
            };

            speechSynthesizer.SynthesisCanceled += (s, e) =>
            {
                Console.WriteLine("SynthesisCanceled event");
            };

            speechSynthesizer.SynthesisCompleted += (s, e) =>
            {
                Console.WriteLine($"SynthesisCompleted event: " +
                    $"{e.Result.AudioData.Length} bytes" +
                    $"\\r\\n\\tDuration: {e.Result.AudioDuration}");
            };
        }
    }
}
```

```

};

speechSynthesizer.SynthesisStarted += (s, e) =>
{
    Console.WriteLine("SynthesisStarted event");
};

speechSynthesizer.Synthesizing += (s, e) =>
{
    Console.WriteLine($"Synthesizing event: " +
        $"{e.Result.AudioData.Length} bytes");
};

speechSynthesizer.VisemeReceived += (s, e) =>
{
    Console.WriteLine($"VisemeReceived event: " +
        $"{(e.AudioOffset + 5000) / 10000}ms" +
        $"VisemeId: {e.VisemeId}");
};

speechSynthesizer.WordBoundary += (s, e) =>
{
    Console.WriteLine($"WordBoundary event: " +
        // Word, Punctuation, or Sentence
        $"{e.BoundaryType}" +
        $"{(e.AudioOffset + 5000) / 10000}ms" +
        $"{e.Duration}" +
        $"{e.Text}" +
        $"{e.TextOffset}" +
        $"{e.WordLength}");
};

// Synthesize the SSML
Console.WriteLine($"SSML to synthesize: \r\n{ssml}");
var speechSynthesisResult = await
speechSynthesizer.SpeakSsmlAsync(ssml);

// Output the results
switch (speechSynthesisResult.Reason)
{
    case ResultReason.SynthesizingAudioCompleted:
        Console.WriteLine("SynthesizingAudioCompleted result");
        break;
    case ResultReason.Canceled:
        var cancellation =
SpeechSynthesisCancellationDetails.FromResult(speechSynthesisResult);
        Console.WriteLine($"CANCELED: Reason={cancellation.Reason}");

        if (cancellation.Reason == CancellationReason.Error)
        {
            Console.WriteLine($"CANCELED: ErrorCode={cancellation.ErrorCode}");
            Console.WriteLine($"CANCELED: ErrorDetails=[{cancellation.ErrorDetails}]");
            Console.WriteLine($"CANCELED: Did you set the speech

```

```
        resource key and region values?" );
    }
    break;
default:
    break;
}
}

Console.WriteLine("Press any key to exit...");
Console.ReadKey();
}
}
```

You can find more text to speech samples at [GitHub](#).

Use a custom endpoint

The custom endpoint is functionally identical to the standard endpoint used for text to speech requests.

One difference is that the `EndpointId` must be specified to use your custom voice via the Speech SDK. You can start with the [text to speech quickstart](#) and then update the code with the `EndpointId` and `SpeechSynthesisVoiceName`.

C#

```
var speechConfig = SpeechConfig.FromSubscription(speechKey, speechRegion);
speechConfig.SpeechSynthesisVoiceName = "YourCustomVoiceName";
speechConfig.EndpointId = "YourEndpointId";
```

To use a custom voice via [Speech Synthesis Markup Language \(SSML\)](#), specify the model name as the voice name. This example uses the `YourCustomVoiceName` voice.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
    <voice name="YourCustomVoiceName">
        This is the text that is spoken.
    </voice>
</speak>
```

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 voice](#)
- [Improve synthesis with SSML](#)

Batch synthesis API for text to speech

08/07/2025

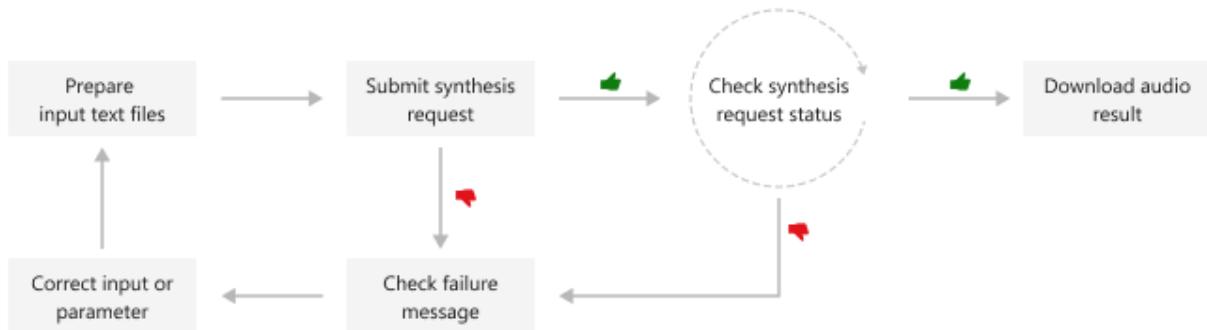
The Batch synthesis API can synthesize a large volume of text input (long and short) asynchronously. Publishers and audio content platforms can create long audio content in a batch. For example: audio books, news articles, and documents. The batch synthesis API can create synthesized audio longer than 10 minutes.

Important

The Batch synthesis API is generally available. The Long Audio API will be retired on April 1st, 2027. For more information, see [Migrate to batch synthesis API](#).

The batch synthesis API is asynchronous and doesn't return synthesized audio in real-time. You submit text files to be synthesized, poll for the status, and download the audio output when the status indicates success. The text inputs must be plain text or [Speech Synthesis Markup Language \(SSML\)](#) text.

This diagram provides a high-level overview of the workflow.



Tip

You can also use the [Speech SDK](#) to create synthesized audio longer than 10 minutes by iterating over the text and synthesizing it in chunks. For a C# example, see [GitHub](#).

You can use the following REST API operations for batch synthesis:

 Expand table

Operation	Method	REST API call
Create batch synthesis	PUT	texttospeech/batchsyntheses/YourSynthesisId
Get batch synthesis	GET	texttospeech/batchsyntheses/YourSynthesisId
List batch synthesis	GET	texttospeech/batchsyntheses
Delete batch synthesis	DELETE	texttospeech/batchsyntheses/YourSynthesisId

For code samples, see [GitHub](#).

Create batch synthesis

To submit a batch synthesis request, construct the HTTP PUT request path and body according to the following instructions:

- Set the required `inputKind` property.
- If the `inputKind` property is set to "PlainText", then you must also set the `voice` property in the `synthesisConfig`. In the following example, the `inputKind` is set to "SSML", so the `synthesisConfig` isn't set.
- Optionally you can set the `description`, `timeToLiveInHours`, and other properties. For more information, see [batch synthesis properties](#).

! Note

The maximum JSON payload size that's accepted is 2 megabytes.

Set the required `YourSynthesisId` in path. The `YourSynthesisId` must be unique. It must be 3-64 long, contains only numbers, letters, hyphens, underscores and dots, starts and ends with a letter or number.

Make an HTTP PUT request using the URI as shown in the following example. Replace `YourSpeechKey` with your Speech resource key, replace `YourSpeechRegion` with your Speech resource region, and set the request body properties as previously described.

Azure CLI

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourSpeechKey" -H "Content-Type: application/json" -d '{
  "description": "my ssml test",
  "inputKind": "SSML",
  "inputs": [
    {
      "text": "Hello, world!"
    }
  ]
}' https://YOUR-SPEECH-RESOURCE-YOU-CREATED/YourSynthesisId
```

```
        "content": "<speak version=\"1.0\" xml:lang=\"en-US\"><voice name=\"en-US-JennyNeural\">The rainbow has seven colors.</voice></speak>"  
    }  
],  
"properties": {  
    "outputFormat": "riff-24khz-16bit-mono-pcm",  
    "wordBoundaryEnabled": false,  
    "sentenceBoundaryEnabled": false,  
    "concatenateResult": false,  
    "decompressOutputFiles": false  
}  
}'  
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses/YourSynthesisId?api-version=2024-04-01"
```

You should receive a response body in the following format:

JSON

```
{  
    "id": "YourSynthesisId",  
    "status": "Running",  
    "createdDateTime": "2024-03-12T07:23:18.0097387Z",  
    "lastActionDateTime": "2024-03-12T07:23:18.0097388Z",  
    "inputKind": "SSML",  
    "customVoices": {},  
    "properties": {  
        "timeToLiveInHours": 168,  
        "outputFormat": "riff-24khz-16bit-mono-pcm",  
        "concatenateResult": false,  
        "decompressOutputFiles": false,  
        "wordBoundaryEnabled": false,  
        "sentenceBoundaryEnabled": false  
    }  
}
```

The `status` property should progress from `Running` status to `Succeeded` or `Failed`. You can call the [GET batch synthesis API](#) periodically until the returned status is `Succeeded` or `Failed`.

Get batch synthesis

To get the status of the batch synthesis job, make an HTTP GET request using the URI as shown in the following example. Replace `YourSpeechKey` with your Speech resource key, and replace `YourSpeechRegion` with your Speech resource region.

Azure CLI

```
curl -v -X GET  
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses/  
YourSynthesisId?api-version=2024-04-01" -H "Ocp-Apim-Subscription-Key:  
YourSpeechKey"
```

You should receive a response body in the following format:

JSON

```
{  
    "id": "YourSynthesisId",  
    "status": "Succeeded",  
    "createdDateTime": "2024-03-12T07:23:18.0097387Z",  
    "lastActionDateTime": "2024-03-12T07:23:18.7979669",  
    "inputKind": "SSML",  
    "customVoices": {},  
    "properties": {  
        "timeToLiveInHours": 168,  
        "outputFormat": "riff-24khz-16bit-mono-pcm",  
        "concatenateResult": false,  
        "decompressOutputFiles": false,  
        "wordBoundaryEnabled": false,  
        "sentenceBoundaryEnabled": false,  
        "sizeInBytes": 120000,  
        "succeededAudioCount": 1,  
        "failedAudioCount": 0,  
        "durationInMilliseconds": 2500,  
        "billingDetails": {  
            "neuralCharacters": 29  
        }  
    },  
    "outputs": {  
        "result": "https://stttsvcuse.blob.core.windows.net/batchsynthesis-  
output/29f2105f997c4bfea176d39d05ff201e/YourSynthesisId/results.zip?SAS_Token"  
    }  
}
```

From `outputs.result`, you can download a ZIP file that contains the audio (such as `0001.wav`), summary, and debug details. For more information, see [batch synthesis results](#).

List batch synthesis

To list all batch synthesis jobs for the Speech resource, make an HTTP GET request using the URI as shown in the following example. Replace `YourSpeechKey` with your Speech resource key and replace `YourSpeechRegion` with your Speech resource region. Optionally, you can set the `skip` and `maxpagesize` (up to 100) query parameters in URL. The default value for `skip` is 0 and the default value for `maxpagesize` is 100.

Azure CLI

```
curl -v -X GET  
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses?  
api-version=2024-04-01&skip=1&maxpagesize=2" -H "Ocp-Apim-Subscription-Key:  
YourSpeechKey"
```

You should receive a response body in the following format:

JSON

```
{  
  "value": [  
    {  
      "id": "my-job-03",  
      "status": "Succeeded",  
      "createdDateTime": "2024-03-12T07:28:32.5690441Z",  
      "lastActionDateTime": "2024-03-12T07:28:33.0042293",  
      "inputKind": "SSML",  
      "customVoices": {},  
      "properties": {  
        "timeToLiveInHours": 168,  
        "outputFormat": "riff-24kHz-16bit-mono-pcm",  
        "concatenateResult": false,  
        "decompressOutputFiles": false,  
        "wordBoundaryEnabled": false,  
        "sentenceBoundaryEnabled": false,  
        "sizeInBytes": 120000,  
        "succeededAudioCount": 1,  
        "failedAudioCount": 0,  
        "durationInMilliseconds": 2500,  
        "billingDetails": {  
          "neuralCharacters": 29  
        }  
      },  
      "outputs": {  
        "result": "https://stttsvcuse.blob.core.windows.net/batchsynthesis-  
output/29f2105f997c4bfea176d39d05ff201e/my-job-03/results.zip?SAS_Token"  
      }  
    },  
    {  
      "id": "my-job-02",  
      "status": "Succeeded",  
      "createdDateTime": "2024-03-12T07:28:29.6418211Z",  
      "lastActionDateTime": "2024-03-12T07:28:30.0910306",  
      "inputKind": "SSML",  
      "customVoices": {},  
      "properties": {  
        "timeToLiveInHours": 168,  
        "outputFormat": "riff-24kHz-16bit-mono-pcm",  
        "concatenateResult": false,  
        "decompressOutputFiles": false,  
        "wordBoundaryEnabled": false,  
        "sentenceBoundaryEnabled": false  
      }  
    }  
  ]  
}
```

```
        "sentenceBoundaryEnabled": false,
        "sizeInBytes": 120000,
        "succeededAudioCount": 1,
        "failedAudioCount": 0,
        "durationInMilliseconds": 2500,
        "billingDetails": {
            "neuralCharacters": 29
        }
    },
    "outputs": {
        "result": "https://stttsvcuse.blob.core.windows.net/batchsynthesis-
output/29f2105f997c4bfea176d39d05ff201e/my-job-02/results.zip?SAS_Token"
    }
}
],
"nextLink":
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses?
skip=3&maxpagesize=2&api-version=2024-04-01"
}
```

From `outputs.result`, you can download a ZIP file that contains the audio (such as `0001.wav`), summary, and debug details. For more information, see [batch synthesis results](#).

The `value` property in the json response lists your synthesis requests. The list is paginated, with a maximum page size of 100. The `"nextLink"` property is provided as needed to get the next page of the paginated list.

Delete batch synthesis

Delete the batch synthesis job history after you retrieved the audio output results. The Speech service keeps batch synthesis history for 168 hours (7 days) by default. Alternatively, you can specify this retention period using the `timeToLiveInHours` property, up to 744 hours (31 days). The date and time of automatic deletion (for synthesis jobs with a status of "Succeeded" or "Failed") is equal to the `lastActionDateTime` + `timeToLiveInHours` properties.

To delete a batch synthesis job, make an HTTP DELETE request using the URI as shown in the following example. Replace `YourSynthesisId` with your batch synthesis ID, replace `YourSpeechKey` with your Speech resource key, and replace `YourSpeechRegion` with your Speech resource region.

Azure CLI

```
curl -v -X DELETE
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses/
YourSynthesisId?api-version=2024-04-01" -H "Ocp-Apim-Subscription-Key:
YourSpeechKey"
```

The response headers include `HTTP/1.1 204 No Content` if the delete request was successful.

Batch synthesis results

After you [get a batch synthesis job](#) with `status` of "Succeeded", you can download the audio output results. Use the URL from the `outputs.result` property of the [batch synthesis GET](#) response.

To get the batch synthesis results file, make an HTTP GET request using the URI as shown in the following example. Replace `YourOutputsResultUrl` with the URL from the `outputs.result` property of the [batch synthesis GET](#) response. Replace `YourSpeechKey` with your Speech resource key.

Azure CLI

```
curl -v -X GET "YourOutputsResultUrl" -H "Ocp-Apim-Subscription-Key: YourSpeechKey" > results.zip
```

The results are in a ZIP file that contains the audio (such as `0001.wav`), summary, and debug details. The numbered prefix of each filename (shown below as `[nnnn]`) is in the same order as the text inputs used when you created the batch synthesis.

(!) Note

The `[nnnn].debug.json` file contains the synthesis result ID and other information that might help with troubleshooting. The properties that it contains might change, so you shouldn't take any dependencies on the JSON format.

The summary file contains the synthesis results for each text input. Here's an example `summary.json` file:

JSON

```
{
  "jobID": "7ab84171-9070-4d3b-88d4-1b8cc1cb928a",
  "status": "Succeeded",
  "results": [
    {
      "contents": ["<speak version=\"1.0\" xml:lang=\"en-US\"><voice name=\"en-US-JennyNeural\">The rainbow has seven colors.</voice></speak>"],
      "status": "Succeeded",
      "audioFileName": "0001.wav",
      "properties": {
        "sizeInBytes": "120000",
        "rate": "16000",
        "volume": "0.5",
        "pitch": "0.0",
        "duration": "1.0"
      }
    }
  ]
}
```

```
        "durationInMilliseconds": "2500"
    }
}
]
```

If sentence boundary data was requested (`"sentenceBoundaryEnabled": true`), then a corresponding `[nnnn].sentence.json` file is included in the results. Likewise, if word boundary data was requested (`"wordBoundaryEnabled": true`), then a corresponding `[nnnn].word.json` file is included in the results.

Here's an example word data file with both audio offset and duration in milliseconds:

JSON

```
[
{
  "Text": "The",
  "AudioOffset": 50,
  "Duration": 137
},
{
  "Text": "rainbow",
  "AudioOffset": 200,
  "Duration": 350
},
{
  "Text": "has",
  "AudioOffset": 562,
  "Duration": 175
},
{
  "Text": "seven",
  "AudioOffset": 750,
  "Duration": 300
},
{
  "Text": "colors",
  "AudioOffset": 1062,
  "Duration": 625
},
{
  "Text": ".",
  "AudioOffset": 1700,
  "Duration": 100
}]
```

Batch synthesis latency and best practices

When using batch synthesis for generating synthesized speech, it's important to consider the latency involved and follow best practices for achieving optimal results.

Latency in batch synthesis

The latency in batch synthesis depends on various factors, including the complexity of the input text, the number of inputs in the batch, and the processing capabilities of the underlying hardware.

The latency for batch synthesis is as follows (approximately):

- The latency of 50% of the synthesized speech outputs is within 10-20 seconds.
- The latency of 95% of the synthesized speech outputs is within 120 seconds.

Best practices

When considering batch synthesis for your application, it's recommended to assess whether the latency meets your requirements. If the latency aligns with your desired performance, batch synthesis can be a suitable choice. However, if the latency doesn't meet your needs, you might consider using real-time API.

HTTP status codes

The section details the HTTP response codes and messages from the batch synthesis API.

HTTP 200 OK

HTTP 200 OK indicates that the request was successful.

HTTP 201 Created

HTTP 201 Created indicates that the batch synthesis create request (via HTTP PUT) was successful.

HTTP 204 error

An HTTP 204 error indicates that the request was successful, but the resource doesn't exist. For example:

- You tried to get or delete a synthesis job that doesn't exist.

- You successfully deleted a synthesis job.

HTTP 400 error

Here are examples that can result in the 400 error:

- The `outputFormat` is unsupported or invalid. Provide a valid format value, or leave `outputFormat` empty to use the default setting.
- The number of requested text inputs exceeded the limit of 10,000.
- You tried to use an invalid deployment ID or a custom voice that isn't successfully deployed. Make sure the Speech resource has access to the custom voice, and the custom voice is successfully deployed. You must also ensure that the mapping of `{"your-custom-voice-name": "your-deployment-ID"}` is correct in your batch synthesis request.
- You tried to use a *F0* Speech resource, but the region only supports the *Standard* Speech resource pricing tier.

HTTP 404 error

The specified entity can't be found. Make sure the synthesis ID is correct.

HTTP 429 error

There are too many recent requests. Each client application can submit up to 100 requests per 10 seconds for each Speech resource. Reduce the number of requests per second.

HTTP 500 error

HTTP 500 Internal Server Error indicates that the request failed. The response body contains the error message.

HTTP error example

Here's an example request that results in an HTTP 400 error, because the `inputs` property is required to create a job.

Console

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourSpeechKey" -H "Content-Type: application/json" -d '{ "inputKind": "SSML" }'
```

```
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses/  
YourSynthesisId?api-version=2024-04-01"
```

In this case, the response headers include `HTTP/1.1 400 Bad Request`.

The response body resembles the following JSON example:

JSON

```
{  
  "error": {  
    "code": "BadRequest",  
    "message": "The inputs is required."  
  }  
}
```

Next steps

- [Speech Synthesis Markup Language \(SSML\)](#)
- [Batch synthesis properties](#)
- [Migrate to batch synthesis](#)

Batch synthesis properties for text to speech

08/07/2025

Important

The Batch synthesis API is generally available. The Long Audio API will be retired on April 1st, 2027. For more information, see [Migrate to batch synthesis API](#).

The Batch synthesis API can synthesize a large volume of text input (long and short) asynchronously. Publishers and audio content platforms can create long audio content in a batch. For example: audio books, news articles, and documents. The batch synthesis API can create synthesized audio longer than 10 minutes.

Some properties in JSON format are required when you create a new batch synthesis job. Other properties are optional. The batch synthesis response includes other properties to provide information about the synthesis status and results. For example, the `outputs.result` property contains the location of the batch synthesis result files with audio output and logs.

Batch synthesis properties

Batch synthesis properties are described in the following table.

 Expand table

Property	Description
<code>createdDateTime</code>	<p>The date and time when the batch synthesis job was created.</p> <p>This property is read-only.</p>
<code>customVoices</code>	<p>The map of a custom voice name and its deployment ID.</p> <p>For example: <code>"customVoices": {"your-custom-voice-name": "502ac834-6537-4bc3-9fd6-140114daa66d"}</code></p> <p>You can use the voice name in your <code>synthesisConfig.voice</code> (when the <code>inputKind</code> is set to <code>"PlainText"</code>) or within the SSML text of <code>inputs</code> (when the <code>inputKind</code> is set to <code>"SSML"</code>).</p> <p>This property is required to use a custom voice. If you try to</p>

Property	Description
	use a custom voice that isn't defined here, the service returns an error.
<code>description</code>	<p>The description of the batch synthesis.</p> <p>This property is optional.</p>
<code>id</code>	<p>The batch synthesis job ID you passed in path.</p> <p>This property is required in path.</p>
<code>inputs</code>	<p>The plain text or SSML to be synthesized.</p> <p>When the <code>inputKind</code> is set to <code>"PlainText"</code>, provide plain text as shown here: <code>"inputs": [{"content": "The rainbow has seven colors."}]</code>. When the <code>inputKind</code> is set to <code>"SSML"</code>, provide text in the Speech Synthesis Markup Language (SSML) as shown here: <code>"inputs": [{"content": "<speak version='1.0' xml:lang='en-US'><voice xml:lang='en-US' xml:gender='Female' name='en-US-AvaMultilingualNeural'>The rainbow has seven colors.</voice></speak>"}]</code>.</p> <p>Include up to 1,000 text objects if you want multiple audio output files. Here's example input text that should be synthesized to two audio output files: <code>"inputs": [{"content": "synthesize this to a file"}, {"content": "synthesize this to another file"}]</code>. However, if the <code>properties.concatenateResult</code> property is set to <code>true</code>, then each synthesized result is written to the same audio output file.</p> <p>You don't need separate text inputs for new paragraphs. Within any of the (up to 1,000) text inputs, you can specify new paragraphs using the <code>\r\n</code> (newline) string. Here's example input text with two paragraphs that should be synthesized to the same audio output file: <code>"inputs": [{"content": "synthesize this to a file\r\nsynthesize this to another paragraph in the same file"}]</code></p> <p>There are no paragraph limits, but the maximum JSON payload size (including all text inputs and other properties) is 2 megabytes.</p> <p>This property is required when you create a new batch synthesis job. This property isn't included in the response when you get the synthesis job.</p>

Property	Description
<code>lastActionDateTime</code>	The most recent date and time when the <code>status</code> property value changed.
	This property is read-only.
<code>outputs.result</code>	The location of the batch synthesis result files with audio output and logs.
	This property is read-only.
<code>properties</code>	A defined set of optional batch synthesis configuration settings.
<code>properties.sizeInBytes</code>	The audio output size in bytes.
	This property is read-only.
<code>properties.billingDetails</code>	The number of words that were processed and billed by <code>customNeuralCharacters</code> (custom voice) versus <code>neuralCharacters</code> (standard voice).
	This property is read-only.
<code>properties.concatenateResult</code>	Determines whether to concatenate the result. This optional <code>bool</code> value ("true" or "false") is "false" by default.
<code>properties.decompressOutputFiles</code>	Determines whether to unzip the synthesis result files in the destination container. This property can only be set when the <code>destinationContainerUrl</code> property is set. This optional <code>bool</code> value ("true" or "false") is "false" by default.
<code>properties.destinationContainerUrl</code>	The batch synthesis results can be stored in a writable Azure container. If you don't specify a container URI with shared access signatures (SAS) token, the Speech service stores the results in a container managed by Microsoft. SAS with stored access policies isn't supported. When the synthesis job is deleted, the result data is also deleted.
	This optional property isn't included in the response when you get the synthesis job.
<code>properties.destinationPath</code>	The prefix path for storing batch synthesis results. If no prefix path is provided, a system-generated path will be used.
	This property is optional and can only be set when the <code>destinationContainerUrl</code> property is specified.
<code>properties.durationInMilliseconds</code>	The audio output duration in milliseconds.

Property	Description
	This property is read-only.
<code>properties.failedAudioCount</code>	The count of batch synthesis inputs to audio output failed. This property is read-only.
<code>properties.outputFormat</code>	The audio output format. For information about the accepted values, see audio output formats . The default output format is <code>riff-24khz-16bit-mono-pcm</code> .
<code>properties.sentenceBoundaryEnabled</code>	Determines whether to generate sentence boundary data. This optional <code>bool</code> value ("true" or "false") is "false" by default. If sentence boundary data is requested, then a corresponding <code>[nnnn].sentence.json</code> file is included in the results data ZIP file.
<code>properties.succeededAudioCount</code>	The count of batch synthesis inputs to audio output succeeded. This property is read-only.
<code>properties.timeToLiveInHours</code>	A duration in hours after the synthesis job is completed, when the synthesis results will be automatically deleted. This optional setting is <code>168</code> (7 days) by default. The maximum time to live is <code>744</code> (31 days). The date and time of automatic deletion (for synthesis jobs with a status of "Succeeded" or "Failed") is equal to the <code>lastActionDateTime</code> + <code>timeToLiveInHours</code> properties. Otherwise, you can call the delete synthesis method to remove the job sooner.
<code>properties.wordBoundaryEnabled</code>	Determines whether to generate word boundary data. This optional <code>bool</code> value ("true" or "false") is "false" by default. If word boundary data is requested, then a corresponding <code>[nnnn].word.json</code> file is included in the results data ZIP file.
<code>status</code>	The batch synthesis processing status. The status should progress from "Running" to either "Succeeded" or "Failed". This property is read-only.

Property	Description
<code>synthesisConfig</code>	<p>The configuration settings to use for batch synthesis of plain text.</p>
	<p>This property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.backgroundAudio</code>	<p>The background audio for each audio output.</p>
	<p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.backgroundAudio.fadein</code>	<p>The duration of the background audio fade-in as milliseconds. The default value is <code>0</code>, which is the equivalent to no fade in. Accepted values: <code>0</code> to <code>10000</code> inclusive.</p>
	<p>For information, see the attributes table under add background audio in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p>
<code>synthesisConfig.backgroundAudio.fadeout</code>	<p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
	<p>The duration of the background audio fade-out in milliseconds. The default value is <code>0</code>, which is the equivalent to no fade out. Accepted values: <code>0</code> to <code>10000</code> inclusive.</p>
	<p>For information, see the attributes table under add background audio in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p>
<code>synthesisConfig.backgroundAudio.src</code>	<p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
	<p>The URI location of the background audio file.</p>
	<p>For information, see the attributes table under add background audio in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p>
<code>synthesisConfig.backgroundAudio.volume</code>	<p>This property is required when <code>synthesisConfig.backgroundAudio</code> is set.</p>
	<p>The volume of the background audio file. Accepted values: <code>0</code> to <code>100</code> inclusive. The default value is <code>1</code>.</p>
	<p>For information, see the attributes table under add background audio in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p>

Property	Description
	<p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.pitch</code>	<p>The pitch of the audio output.</p> <p>For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p> <p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.rate</code>	<p>The rate of the audio output.</p> <p>For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p> <p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.role</code>	<p>For some voices, you can adjust the speaking role-play. The voice can imitate a different age and gender, but the voice name isn't changed. For example, a male voice can raise the pitch and change the intonation to imitate a female voice, but the voice name isn't changed. If the role is missing or isn't supported for your voice, this attribute is ignored.</p> <p>For information about the available styles per voice, see voice styles and roles.</p> <p>This optional property is only applicable when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.speakerProfileId</code>	<p>The speaker profile ID of a personal voice.</p> <p>For information about available personal voice base model names, see integrate personal voice.</p> <p>For information about how to get the speaker profile ID, see language and voice support.</p> <p>This property is required when <code>inputKind</code> is set to <code>"PlainText"</code>.</p>
<code>synthesisConfig.style</code>	<p>For some voices, you can adjust the speaking style to express different emotions like cheerfulness, empathy, and calm. You can optimize the voice for different scenarios like customer</p>

Property	Description
	service, newscast, and voice assistant.
	For information about the available styles per voice, see voice styles and roles .
	This optional property is only applicable when <code>synthesisConfig.style</code> is set.
<code>synthesisConfig.styleDegree</code>	The intensity of the speaking style. You can specify a stronger or softer style to make the speech more expressive or subdued. The range of accepted values are: 0.01 to 2 inclusive. The default value is 1, which means the predefined style intensity. The minimum unit is 0.01, which results in a slight tendency for the target style. A value of 2 results in a doubling of the default style intensity. If the style degree is missing or isn't supported for your voice, this attribute is ignored.
	For information about the available styles per voice, see voice styles and roles .
	This optional property is only applicable when <code>inputKind</code> is set to "PlainText".
<code>synthesisConfig.voice</code>	The voice that speaks the audio output.
	For information about the available standard voices, see language and voice support . To use a custom voice, you must specify a valid custom voice and deployment ID mapping in the <code>customVoices</code> property. To use a personal voice, you need to specify the <code>synthesisConfig.speakerProfileId</code> property.
	This property is required when <code>inputKind</code> is set to "PlainText".
<code>synthesisConfig.volume</code>	The volume of the audio output.
	For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.
	This optional property is only applicable when <code>inputKind</code> is set to "PlainText".
<code>inputKind</code>	Indicates whether the <code>inputs</code> text property should be plain text or SSML. The possible case-insensitive values are "PlainText" and "SSML". When the <code>inputKind</code> is set to

Property	Description
	<p>"PlainText", you must also set the <code>synthesisConfig</code> voice property.</p> <p>This property is required.</p>

Batch synthesis latency and best practices

When using batch synthesis for generating synthesized speech, it's important to consider the latency involved and follow best practices for achieving optimal results.

Latency in batch synthesis

The latency in batch synthesis depends on various factors, including the complexity of the input text, the number of inputs in the batch, and the processing capabilities of the underlying hardware.

The latency for batch synthesis is as follows (approximately):

- The latency of 50% of the synthesized speech outputs is within 10-20 seconds.
- The latency of 95% of the synthesized speech outputs is within 120 seconds.

Best practices

When considering batch synthesis for your application, it's recommended to assess whether the latency meets your requirements. If the latency aligns with your desired performance, batch synthesis can be a suitable choice. However, if the latency doesn't meet your needs, you might consider using real-time API.

HTTP status codes

The section details the HTTP response codes and messages from the batch synthesis API.

HTTP 200 OK

HTTP 200 OK indicates that the request was successful.

HTTP 201 Created

HTTP 201 Created indicates that the create batch synthesis request (via HTTP POST) was successful.

HTTP 204 error

An HTTP 204 error indicates that the request was successful, but the resource doesn't exist. For example:

- You tried to get or delete a synthesis job that doesn't exist.
- You successfully deleted a synthesis job.

HTTP 400 error

Here are examples that can result in the 400 error:

- The `outputFormat` is unsupported or invalid. Provide a valid format value, or leave `outputFormat` empty to use the default setting.
- The number of requested text inputs exceeded the limit of 10,000.
- You tried to use an invalid deployment ID or a custom voice that isn't successfully deployed. Make sure the Speech resource has access to the custom voice, and the custom voice is successfully deployed. You must also ensure that the mapping of `{"your-custom-voice-name": "your-deployment-ID"}` is correct in your batch synthesis request.
- You tried to use a *F0* Speech resource, but the region only supports the *Standard* Speech resource pricing tier.

HTTP 404 error

The specified entity can't be found. Make sure the synthesis ID is correct.

HTTP 429 error

There are too many recent requests. Each client application can submit up to 100 requests per 10 seconds for each Speech resource. Reduce the number of requests per second.

HTTP 500 error

HTTP 500 Internal Server Error indicates that the request failed. The response body contains the error message.

HTTP error example

Here's an example request that results in an HTTP 400 error, because the `inputs` property is required to create a job.

Console

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourSpeechKey" -H "Content-Type: application/json" -d '{  
    "inputKind": "SSML"  
}'  
"https://YourSpeechRegion.api.cognitive.microsoft.com/texttospeech/batchsyntheses/  
YourSynthesisId?api-version=2024-04-01"
```

In this case, the response headers include `HTTP/1.1 400 Bad Request`.

The response body resembles the following JSON example:

JSON

```
{  
    "error": {  
        "code": "BadRequest",  
        "message": "The inputs is required."  
    }  
}
```

Next steps

- [Speech Synthesis Markup Language \(SSML\)](#)
- [Text to speech quickstart](#)
- [Migrate to batch synthesis](#)

Speech Synthesis Markup Language (SSML) overview

08/07/2025

Speech Synthesis Markup Language (SSML) is an XML-based markup language that you can use to fine-tune your text to speech output attributes such as pitch, pronunciation, speaking rate, volume, and more. It gives you more control and flexibility than plain text input.

💡 Tip

You can hear voices in different styles and pitches reading example text by using the [Voice Gallery](#).

Use case scenarios

SSML is designed to give you flexibility in how you want your speech output to sound, and it provides different properties for how you can customize that output. You can use SSML to:

- [Define the input text structure](#) that determines the structure, content, and other characteristics of your text to speech output. For example, you can use SSML to define a paragraph, a sentence, a break or a pause, or silence. You can wrap text with event tags, like a bookmark or viseme, that your application can process later. A viseme is the visual description of a phoneme, the individual speech sounds, in spoken language.
- [Choose the voice](#), language, name, style, and role. You can use multiple voices in a single SSML document. You can also adjust the emphasis, speaking rate, pitch, and volume. SSML can also insert prerecorded audio, such as a sound effect or a musical note.
- [Control pronunciation](#) of the output audio. For example, you can use SSML with phonemes and a custom lexicon to improve pronunciation. You can also use SSML to define how a word or mathematical expression is pronounced.

Ways to work with SSML

SSML functionality is available in various tools that might fit your use case.

ⓘ Important

You're billed for each character that's converted to speech, including punctuation. Although the SSML document itself isn't billable, the service counts optional elements that

you use to adjust how the text is converted to speech, like phonemes and pitch, as billable characters. For more information, see the [pricing note](#).

You can use SSML in the following ways:

- [The audio content creation](#) tool lets you author plain text and SSML in Speech Studio. You can listen to the output audio and adjust the SSML to improve speech synthesis. For more information, see [Speech synthesis with the Audio Content Creation tool](#).
- [The batch synthesis API](#) accepts SSML via the `inputs` property.
- [The Speech CLI](#) accepts SSML via the `spx synthesize --ssml SSML` command line argument.
- [The Speech SDK](#) accepts SSML via the "speak" SSML method across the different supported languages.

Next steps

- [SSML document structure and events](#)
- [Voice and sound with SSML](#)
- [Pronunciation with SSML](#)
- [Language and voice support for the Speech service](#)

SSML document structure and events

08/07/2025

The Speech Synthesis Markup Language (SSML) with input text determines the structure, content, and other characteristics of the text to speech output. For example, you can use SSML to define a paragraph, a sentence, a break or a pause, or silence. You can wrap text with event tags such as bookmark or viseme that can be processed later by your application.

Refer to the sections below for details about how to structure elements in the SSML document.

ⓘ Note

In addition to Azure AI Speech neural (non HD) voices, you can also use [Azure AI Speech high definition \(HD\) voices](#) and [Azure OpenAI neural \(HD and non HD\) voices](#). The HD voices provide a higher quality for more versatile scenarios.

Some voices don't support all [Speech Synthesis Markup Language \(SSML\)](#) tags. This includes neural text to speech HD voices, personal voices, and embedded voices.

- For Azure AI Speech high definition (HD) voices, check the SSML support [here](#).
- For personal voice, you can find the SSML support [here](#).
- For embedded voices, check the SSML support [here](#).

Document structure

The Speech service implementation of SSML is based on the World Wide Web Consortium's [Speech Synthesis Markup Language Version 1.0](#). The elements supported by the Speech can differ from the W3C standard.

Each SSML document is created with SSML elements or tags. These elements are used to adjust the voice, style, pitch, prosody, volume, and more.

Here's a subset of the basic structure and syntax of an SSML document:

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="string">
    <mstts:backgroundaudio src="string" volume="string" fadein="string"
      fadeout="string"/>
    <mstts:voiceconversion url="string"/>
    <voice name="string" effect="string">
      <audio src="string"></audio>
```

```

<bookmark mark="string"/>
<break strength="string" time="string" />
<emphasis level="value"></emphasis>
<lang xml:lang="string"></lang>
<lexicon uri="string"/>
<math xmlns="http://www.w3.org/1998/Math/MathML"></math>
<mstts:audioduration value="string"/>
<mstts:ttseembedding speakerProfileId="string"></mstts:ttseembedding>
<mstts:express-as style="string" styleddegree="value" role="string">
</mstts:express-as>
<mstts:silence type="string" value="string"/>
<mstts:viseme type="string"/>
<p></p>
<phoneme alphabet="string" ph="string"></phoneme>
<prosody pitch="value" contour="value" range="value" rate="value"
volume="value"></prosody>
<s></s>
<say-as interpret-as="string" format="string" detail="string"></say-as>
<sub alias="string"></sub>
</voice>
</speak>

```

Some examples of contents that are allowed in each element are described in the following list:

- `audio`: The body of the `audio` element can contain plain text or SSML markup that's spoken if the audio file is unavailable or unplayable. The `audio` element can also contain text and the following elements: `audio`, `break`, `p`, `s`, `phoneme`, `prosody`, `say-as`, and `sub`.
- `bookmark`: This element can't contain text or any other elements.
- `break`: This element can't contain text or any other elements.
- `emphasis`: This element can contain text and the following elements: `audio`, `break`, `emphasis`, `lang`, `phoneme`, `prosody`, `say-as`, and `sub`.
- `lang`: This element can contain all other elements except `mstts:backgroundaudio`, `voice`, and `speak`.
- `lexicon`: This element can't contain text or any other elements.
- `math`: This element can only contain text and MathML elements.
- `mstts:audioduration`: This element can't contain text or any other elements.
- `mstts:backgroundaudio`: This element can't contain text or any other elements.
- `<mstts:voiceconversion>`: This element can't contain text or any other elements. It specifies the source audio URL for the voice conversion.
- `mstts:embedding`: This element can contain text and the following elements: `audio`, `break`, `emphasis`, `lang`, `phoneme`, `prosody`, `say-as`, and `sub`.
- `mstts:express-as`: This element can contain text and the following elements: `audio`, `break`, `emphasis`, `lang`, `phoneme`, `prosody`, `say-as`, and `sub`.
- `mstts:silence`: This element can't contain text or any other elements.

- `mstts:viseme`: This element can't contain text or any other elements.
- `p`: This element can contain text and the following elements: `audio`, `break`, `phoneme`, `prosody`, `say-as`, `sub`, `mstts:express-as`, and `s`.
- `phoneme`: This element can only contain text and no other elements.
- `prosody`: This element can contain text and the following elements: `audio`, `break`, `p`, `phoneme`, `prosody`, `say-as`, `sub`, and `s`.
- `s`: This element can contain text and the following elements: `audio`, `break`, `phoneme`, `prosody`, `say-as`, `mstts:express-as`, and `sub`.
- `say-as`: This element can only contain text and no other elements.
- `sub`: This element can only contain text and no other elements.
- `speak`: The root element of an SSML document. This element can contain the following elements: `mstts:backgroundaudio` and `voice`.
- `voice`: This element can contain all other elements except `mstts:backgroundaudio` and `speak`.

The Speech service automatically handles punctuation as appropriate, such as pausing after a period, or using the correct intonation when a sentence ends with a question mark.

Special characters

To use the characters `&`, `<`, and `>` within the SSML element's value or text, you must use the entity format. Specifically you must use `&` in place of `&`, use `<` in place of `<`, and use `>` in place of `>`. Otherwise the SSML isn't parsed correctly.

For example, specify `green & yellow` instead of `green & yellow`. The following SSML is parsed as expected:

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
    <voice name="en-US-AvaNeural">
        My favorite colors are green & yellow.
    </voice>
</speak>
```

Special characters such as quotation marks, apostrophes, and brackets, must be escaped. For more information, see [Extensible Markup Language \(XML\) 1.0: Appendix D ↗](#).

Double or single quotation marks must enclose the attribute values. For example, `<prosody volume="90">` and `<prosody volume='90'>` are well-formed, valid elements, but `<prosody volume=90>` isn't recognized.

Speak root element

The `speak` element contains information such as version, language, and the markup vocabulary definition. The `speak` element is the root element that's required for all SSML documents. You must specify the default language within the `speak` element, whether or not the language is adjusted elsewhere such as within the `lang` element.

Here's the syntax for the `speak` element:

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"  
xml:lang="string"></speak>
```

[+] Expand table

Attribute	Description	Required or optional
<code>version</code>	Indicates the version of the SSML specification used to interpret the document markup. The current version is "1.0".	Required
<code>xml:lang</code>	The language of the root document. The value can contain a language code such as <code>en</code> (English), or a locale such as <code>en-US</code> (English - United States).	Required
<code>xmlns</code>	The URI to the document that defines the markup vocabulary (the element types and attribute names) of the SSML document. The current URI is "http://www.w3.org/2001/10/synthesis".	Required

The `speak` element must contain at least one [voice element](#).

speak examples

The supported values for attributes of the `speak` element were described previously.

Single voice example

This example uses the `en-US-AvaNeural` voice. For more examples, see [voice examples](#).

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">  
  <voice name="en-US-AvaNeural">  
    This is the text that is spoken.  
  </voice>  
</speak>
```

```
</voice>  
</speak>
```

Add a break

Use the `break` element to override the default behavior of breaks or pauses between words. Otherwise the Speech service automatically inserts pauses.

Usage of the `break` element's attributes are described in the following table.

[Expand table](#)

Attribute	Description	Required or optional
<code>strength</code>	The relative duration of a pause by using one of the following values: <ul style="list-style-type: none">• x-weak• weak• medium (default)• strong• x-strong	Optional
<code>time</code>	The absolute duration of a pause in seconds (such as <code>2s</code>) or milliseconds (such as <code>500ms</code>). Valid values range from 0 to 20000 milliseconds. If you set a value greater than the supported maximum, the service uses <code>20000ms</code> . If the <code>time</code> attribute is set, the <code>strength</code> attribute is ignored.	Optional

Here are more details about the `strength` attribute.

[Expand table](#)

Strength	Relative duration
X-weak	250 ms
Weak	500 ms
Medium	750 ms
Strong	1,000 ms
X-strong	1,250 ms

Break examples

The supported values for attributes of the `break` element were described previously. The following three ways all add 750 ms breaks.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    Welcome <break /> to text to speech.
    Welcome <break strength="medium" /> to text to speech.
    Welcome <break time="750ms" /> to text to speech.
  </voice>
</speak>
```

Add silence

Use the `mstts:silence` element to insert pauses before or after text, or between two adjacent sentences.

One of the differences between `mstts:silence` and `break` is that a `break` element can be inserted anywhere in the text. Silence only works at the beginning or end of input text or at the boundary of two adjacent sentences.

The silence setting is applied to all input text within its enclosing `voice` element. To reset or change the silence setting again, you must use a new `voice` element with either the same voice or a different voice.

Usage of the `mstts:silence` element's attributes are described in the following table.

[] Expand table

Attribute	Description	Required or optional
<code>type</code>	Specifies where and how to add silence. The following silence types are supported: <ul style="list-style-type: none">• <code>Leading</code> – Extra silence at the beginning of the text. The value that you set is added to the natural silence before the start of text.• <code>Leading-exact</code> – Silence at the beginning of the text. The value is an absolute silence length.• <code>Tailing</code> – Extra silence at the end of text. The value that you set is added to the natural silence after the last word.• <code>Tailing-exact</code> – Silence at the end of the text. The value is an absolute silence length.	Required

Attribute	Description	Required or optional
	<ul style="list-style-type: none"> • <code>Sentenceboundary</code> – Extra silence between adjacent sentences. The actual silence length for this type includes the natural silence after the last word in the previous sentence, the value you set for this type, and the natural silence before the starting word in the next sentence. • <code>Sentenceboundary-exact</code> – Silence between adjacent sentences. The value is an absolute silence length. • <code>Comma-exact</code> – Silence at the comma in half-width or full-width format. The value is an absolute silence length. • <code>Semicolon-exact</code> – Silence at the semicolon in half-width or full-width format. The value is an absolute silence length. • <code>Enumerationcomma-exact</code> – Silence at the enumeration comma in full-width format. The value is an absolute silence length. <p>An absolute silence type (with the <code>-exact</code> suffix) replaces any otherwise natural leading or trailing silence. Absolute silence types take precedence over the corresponding non-absolute type. For example, if you set both <code>Leading</code> and <code>Leading-exact</code> types, the <code>Leading-exact</code> type takes effect. The WordBoundary event takes precedence over punctuation-related silence settings including <code>Comma-exact</code>, <code>Semicolon-exact</code>, or <code>Enumerationcomma-exact</code>. When you use both the <code>WordBoundary</code> event and punctuation-related silence settings, the punctuation-related silence settings don't take effect.</p>	
<code>Value</code>	The duration of a pause in seconds (such as <code>2s</code>) or milliseconds (such as <code>500ms</code>). Valid values range from 0 to 20000 milliseconds. If you set a value greater than the supported maximum, the service uses <code>20000ms</code> .	Required

mstts:silence examples

The supported values for attributes of the `mstts:silence` element were [described previously](#).

In this example, `mstts:silence` is used to add 200 ms of silence between two sentences.

XML

```

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <mstts:silence type="Sentenceboundary" value="200ms"/>
    If we're home schooling, the best we can do is roll with what each day brings and
    try to have fun along the way.
    A good place to start is by trying out the slew of educational apps that are
    helping children stay happy and smash their schooling at the same time.

```

```
</voice>
</speak>
```

In this example, `mstts:silence` is used to add 50 ms of silence at the comma, 100 ms of silence at the semicolon, and 150 ms of silence at the enumeration comma.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts" xml:lang="zh-CN">
  <voice name="zh-CN-YunxiNeural">
    <mstts:silence type="comma-exact" value="50ms"/><mstts:silence type="semicolon-
      exact" value="100ms"/><mstts:silence type="enumerationcomma-exact" value="150ms"/>
    你好呀，云希、晓晓；你好呀。
  </voice>
</speak>
```

Specify paragraphs and sentences

The `p` and `s` elements are used to denote paragraphs and sentences, respectively. In the absence of these elements, the Speech service automatically determines the structure of the SSML document.

Paragraph and sentence examples

The following example defines two paragraphs that each contain sentences. In the second paragraph, the Speech service automatically determines the sentence structure, since they aren't defined in the SSML document.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <p>
      <s>Introducing the sentence element.</s>
      <s>Used to mark individual sentences.</s>
    </p>
    <p>
      Another simple paragraph.
      Sentence structure in this paragraph is not explicitly marked.
    </p>
  </voice>
</speak>
```

Bookmark element

You can use the `bookmark` element in SSML to reference a specific location in the text or tag sequence. Then you use the Speech SDK and subscribe to the `BookmarkReached` event to get the offset of each marker in the audio stream. The `bookmark` element isn't spoken. For more information, see [Subscribe to synthesizer events](#).

Usage of the `bookmark` element's attributes are described in the following table.

 [Expand table](#)

Attribute	Description	Required or optional
<code>mark</code>	The reference text of the <code>bookmark</code> element.	Required

Bookmark examples

The supported values for attributes of the `bookmark` element were [described previously](#).

As an example, you might want to know the time offset of each flower word in the following snippet:

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    We are selling <bookmark mark='flower_1'>roses and <bookmark
    mark='flower_2'>daisies.
  </voice>
</speak>
```

Next steps

- [SSML overview](#)
- [Voice and sound with SSML](#)
- [Language support: Voices, locales, languages](#)

Customize voice and sound with SSML

07/09/2025

You can use Speech Synthesis Markup Language (SSML) to specify the text to speech voice, language, name, style, and role for your speech output. You can also use multiple voices in a single SSML document, and adjust the emphasis, speaking rate, pitch, and volume. In addition, SSML features the ability to insert prerecorded audio, such as a sound effect or a musical note.

The article shows you how to use SSML elements to specify voice and sound. For more information about SSML syntax, see [SSML document structure and events](#).

Use voice elements

At least one `voice` element must be specified within each SSML `speak` element. This element determines the voice that's used for text to speech.

You can include multiple `voice` elements in a single SSML document. Each `voice` element can specify a different voice. You can also use the same voice multiple times with different settings, such as when you [change the silence duration](#) between sentences.

The following table describes the usage of the `voice` element's attributes:

 Expand table

Attribute	Description	Required or optional
<code>name</code>	The voice used for text to speech output. For a complete list of supported standard voices, see Language support .	Required
<code>effect</code>	<p>The audio effect processor that's used to optimize the quality of the synthesized speech output for specific scenarios on devices.</p> <p>For some scenarios in production environments, the auditory experience might be degraded due to the playback distortion on certain devices. For example, the synthesized speech from a car speaker might sound dull and muffled due to environmental factors such as speaker response, room reverberation, and background noise. The passenger might have to turn up the volume to hear more clearly. To avoid manual operations in such a scenario, the audio effect processor can make the sound clearer by compensating the distortion of playback.</p>	Optional

The following values are supported:

Attribute	Description	Required or optional
	<ul style="list-style-type: none"> • <code>eq_car</code> – Optimize the auditory experience when providing high-fidelity speech in cars, buses, and other enclosed automobiles. • <code>eq_telecomhp8k</code> – Optimize the auditory experience for narrowband speech in telecom or telephone scenarios. You should use a sampling rate of 8 kHz. If the sample rate isn't 8 kHz, the auditory quality of the output speech isn't optimized. <p>If the value is missing or invalid, this attribute is ignored and no effect is applied.</p>	Required or optional

Voice examples

For information about the supported values for attributes of the `voice` element, see [Use voice elements](#).

Single voice example

This example uses the `en-US-AvaMultilingualNeural` voice.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    This is the text that is spoken.
  </voice>
</speak>
```

Multiple voices example

Within the `speak` element, you can specify multiple voices for text to speech output. These voices can be in different languages. For each voice, the text must be wrapped in a `voice` element.

This example alternates between the `en-US-AvaMultilingualNeural` and `en-US-AndrewMultilingualNeural` voices. The neural multilingual voices can speak different languages based on the input text.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    Good morning!
  </voice>
  <voice name="en-US-AndrewMultilingualNeural">
    Good morning to you too Ava!
  </voice>
</speak>
```

Custom voice example

To use your [custom voice](#), specify the model name as the voice name in SSML.

This example uses a custom voice named `my-custom-voice`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="my-custom-voice">
    This is the text that is spoken.
  </voice>
</speak>
```

Audio effect example

You use the `effect` attribute to optimize the auditory experience for scenarios such as cars and telecommunications. The following SSML example uses the `effect` attribute with the configuration in car scenarios.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural" effect="eq_car">
    This is the text that is spoken.
  </voice>
</speak>
```

Multi-talker voice example

Multi-talker voices enable natural, dynamic conversations with multiple distinct speakers. This innovation enhances the realism of synthesized dialogues by preserving contextual flow, emotional consistency, and natural speech patterns.

Use this capability to generate engaging, podcast-style speech or conversational exchanges with seamless transitions between speakers. Unlike single-talker models, which synthesize each turn in isolation, multi-talker voices maintain coherence across dialogue, ensuring a more authentic and immersive listening experience.

For `en-US-MultiTalker-Ava-Andrew:DragonHDLatestNeural`, within the `<mstts:dialog>` element, you can specify each turn for the text to speech output, with the format below to alternates between the speaker `ava` and `andrew` for each turn.

XML

```
<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
      xmlns:mstts='https://www.w3.org/2001/mstts' xml:lang='en-US'>
    <voice name='en-US-MultiTalker-Ava-Andrew:DragonHDLatestNeural'>
      <mstts:dialog>
        <mstts:turn speaker="ava">Hello, Andrew! How's your day going?
      </mstts:turn>
        <mstts:turn speaker="andrew">Hey Ava! It's been great, just exploring
some AI advancements in communication.</mstts:turn>
        <mstts:turn speaker="ava">That sounds interesting! What kind of
projects are you working on?</mstts:turn>
        <mstts:turn speaker="andrew">Well, we've been experimenting with text-
to-speech applications, including turning emails into podcasts.</mstts:turn>
        <mstts:turn speaker="ava">Wow, that could really improve content
accessibility! Are you looking for collaborators?</mstts:turn>
        <mstts:turn speaker="andrew">Absolutely! We're open to testing new
ideas and seeing how AI can enhance communication.</mstts:turn>
      </mstts:dialog>
    </voice>
</speak>
```

For supported voices, see the [Language support](#) documentation.

Use speaking styles and roles

By default, neural voices have a neutral speaking style. You can adjust the speaking style, style degree, and role at the sentence level.

(!) Note

The Speech service supports styles, style degree, and roles for a subset of neural voices as described in the [voice styles and roles](#) documentation. To determine the supported styles and roles for each voice, you can also use the [list voices](#) API and the [audio content creation](#) ↗ web application.

The following table describes the usage of the `mstts:express-as` element's attributes:

[+] Expand table

Attribute	Description	Required or optional
<code>style</code>	The voice-specific speaking style. You can express emotions like cheerfulness, empathy, and calmness. You can also optimize the voice for different scenarios like customer service, newscast, and voice assistant. If the style value is missing or invalid, the entire <code>mstts:express-as</code> element is ignored and the service uses the default neutral speech. For custom voice styles, see the custom voice style example .	Required
<code>styledegree</code>	The intensity of the speaking style. You can specify a stronger or softer style to make the speech more expressive or subdued. The range of accepted values are: <code>0.01</code> to <code>2</code> inclusive. The default value is <code>1</code> , which means the predefined style intensity. The minimum unit is <code>0.01</code> , which results in a slight tendency for the target style. A value of <code>2</code> results in a doubling of the default style intensity. If the style degree is missing or isn't supported for your voice, this attribute is ignored.	Optional
<code>role</code>	The speaking role-play. The voice can imitate a different age and gender, but the voice name isn't changed. For example, a male voice can raise the pitch and change the intonation to imitate a female voice, but the voice name isn't changed. If the role is missing or isn't supported for your voice, this attribute is ignored.	Optional

The following table describes each supported `style` attribute:

[+] Expand table

Style	Description
<code>style="advertisement_upbeat"</code>	Expresses an excited and high-energy tone for promoting a product or service.
<code>style="affectionate"</code>	Expresses a warm and affectionate tone, with higher pitch and vocal energy. The speaker is in a state of attracting the attention of the listener. The personality of the speaker is often endearing in nature.
<code>style="angry"</code>	Expresses an angry and annoyed tone.
<code>style="assistant"</code>	Expresses a warm and relaxed tone for digital assistants.
<code>style="calm"</code>	Expresses a cool, collected, and composed attitude when speaking. Tone, pitch, and prosody are more uniform compared to other

Style	Description
	types of speech.
<code>style="chat"</code>	Expresses a casual and relaxed tone.
<code>style="cheerful"</code>	Expresses a positive and happy tone.
<code>style="customerservice"</code>	Expresses a friendly and helpful tone for customer support.
<code>style="depressed"</code>	Expresses a melancholic and despondent tone with lower pitch and energy.
<code>style="disgruntled"</code>	Expresses a disdainful and complaining tone. Speech of this emotion displays displeasure and contempt.
<code>style="documentary-narration"</code>	Narrates documentaries in a relaxed, interested, and informative style suitable for documentaries, expert commentary, and similar content.
<code>style="embarrassed"</code>	Expresses an uncertain and hesitant tone when the speaker is feeling uncomfortable.
<code>style="empathetic"</code>	Expresses a sense of caring and understanding.
<code>style="envious"</code>	Expresses a tone of admiration when you desire something that someone else has.
<code>style="excited"</code>	Expresses an upbeat and hopeful tone. It sounds like something great is happening and the speaker is happy about it.
<code>style="fearful"</code>	Expresses a scared and nervous tone, with higher pitch, higher vocal energy, and faster rate. The speaker is in a state of tension and unease.
<code>style="friendly"</code>	Expresses a pleasant, inviting, and warm tone. It sounds sincere and caring.
<code>style="gentle"</code>	Expresses a mild, polite, and pleasant tone, with lower pitch and vocal energy.
<code>style="hopeful"</code>	Expresses a warm and yearning tone. It sounds like something good is expected to happen to the speaker.
<code>style="lyrical"</code>	Expresses emotions in a melodic and sentimental way.
<code>style="narration-professional"</code>	Expresses a professional, objective tone for content reading.
<code>style="narration-relaxed"</code>	Expresses a soothing and melodious tone for content reading.
<code>style="newscast"</code>	Expresses a formal and professional tone for narrating news.
<code>style="newscast-casual"</code>	Expresses a versatile and casual tone for general news delivery.

Style	Description
<code>style="newscast-formal"</code>	Expresses a formal, confident, and authoritative tone for news delivery.
<code>style="poetry-reading"</code>	Expresses an emotional and rhythmic tone while reading a poem.
<code>style="sad"</code>	Expresses a sorrowful tone.
<code>style="serious"</code>	Expresses a strict and commanding tone. Speaker often sounds stiffer and much less relaxed with firm cadence.
<code>style="shouting"</code>	Expresses a tone that sounds as if the voice is distant or in another location and making an effort to be clearly heard.
<code>style="sports_commentary"</code>	Expresses a relaxed and interested tone for broadcasting a sports event.
<code>style="sports_commentary_excited"</code>	Expresses an intensive and energetic tone for broadcasting exciting moments in a sports event.
<code>style="whispering"</code>	Expresses a soft tone that's trying to make a quiet and gentle sound.
<code>style="terrified"</code>	Expresses a scared tone, with a faster pace and a shakier voice. It sounds like the speaker is in an unsteady and frantic status.
<code>style="unfriendly"</code>	Expresses a cold and indifferent tone.

The following table has descriptions of each supported `role` attribute:

[Expand table](#)

Role	Description
<code>role="Girl"</code>	The voice imitates a girl.
<code>role="Boy"</code>	The voice imitates a boy.
<code>role="YoungAdultFemale"</code>	The voice imitates a young adult female.
<code>role="YoungAdultMale"</code>	The voice imitates a young adult male.
<code>role="OlderAdultFemale"</code>	The voice imitates an older adult female.
<code>role="OlderAdultMale"</code>	The voice imitates an older adult male.
<code>role="SeniorFemale"</code>	The voice imitates a senior female.
<code>role="SeniorMale"</code>	The voice imitates a senior male.

mstts:express-as examples

For information about the supported values for attributes of the `mstts:express-as` element, see [Use speaking styles and roles](#).

Style and degree example

You use the `mstts:express-as` element to express emotions like cheerfulness, empathy, and calm. You can also optimize the voice for different scenarios like customer service, newscast, and voice assistant.

The following SSML example uses the `<mstts:express-as>` element with a `sad` style degree of `2`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="zh-CN">
  <voice name="zh-CN-XiaomoNeural">
    <mstts:express-as style="sad" styledegree="2">
      快走吧，路上一定要注意安全，早去早回。
    </mstts:express-as>
  </voice>
</speak>
```

Role example

Apart from adjusting the speaking styles and style degree, you can also adjust the `role` parameter so that the voice imitates a different age and gender. For example, a male voice can raise the pitch and change the intonation to imitate a female voice, but the voice name isn't changed.

This SSML snippet illustrates how the `role` attribute is used to change the role-play for `zh-CN-XiaomoNeural`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="zh-CN">
  <voice name="zh-CN-XiaomoNeural">
    女儿看见父亲走了进来，问道：
    <mstts:express-as role="YoungAdultFemale" style="calm">
      “您来的挺快的，怎么过来的？”
    </mstts:express-as>
    父亲放下手提包，说：
```

```
<mstts:express-as role="OlderAdultMale" style="calm">
    “刚打车过来的，路上还挺顺畅。”
</mstts:express-as>
</voice>
</speak>
```

Custom voice style example

You can train your custom voice to speak with some preset styles such as `cheerful`, `sad`, and `whispering`. You can also [fine-tune a professional voice](#) to speak in a custom style as determined by your training data. To use your custom voice style in SSML, specify the style name that you previously entered in Speech Studio.

This example uses a custom voice named `my-custom-voice`. The custom voice speaks with the `cheerful` preset style and style degree of `2`, and then with a custom style named `my-custom-style` and style degree of `0.01`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice name="my-custom-voice">
        <mstts:express-as style="cheerful" styledegree="2">
            That'd be just amazing!
        </mstts:express-as>
        <mstts:express-as style="my-custom-style" styledegree="0.01">
            What's next?
        </mstts:express-as>
    </voice>
</speak>
```

Speaker profile ID

You use the `mstts:ttseembedding` element to specify the `speakerProfileId` property for a [personal voice](#). Personal voice is a custom voice trained on your own voice or your customer's voice. For more information, see [create a personal voice](#).

The following SSML example uses the `<mstts:ttseembedding>` element with a voice name and speaker profile ID.

XML

```
<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
      xmlns:mstts='http://www.w3.org/2001/mstts' xml:lang='en-US'>
    <voice xml:lang='en-US' xml:gender='Male' name='PhoenixV2Neural'>
```

```

<mstts:ttseembedding speakerProfileId='your speaker profile ID here'>
    I'm happy to hear that you find me amazing and that I have made your trip
    planning easier and more fun. 我很高兴听到你觉得我很了不起，我让你的旅行计划更轻松、更
    有趣。 Je suis heureux d'apprendre que vous me trouvez incroyable et que j'ai rendu
    la planification de votre voyage plus facile et plus amusante.
</mstts:ttseembedding>
</voice>
</speak>

```

Adjust speaking languages

By default, multilingual voices can autodetect the language of the input text and speak in the language of the default locale of the input text without using SSML. Optionally, you can use the `<lang xml:lang>` element to adjust the speaking language for these voices to set the preferred accent such as `en-GB` for British English. You can adjust the speaking language at both the sentence level and word level. For information about the supported languages for multilingual voice, see [Multilingual voices with the lang element](#) for a table showing the `<lang>` syntax and attribute definitions.

The following table describes the usage of the `<lang xml:lang>` element's attributes:

[] [Expand table](#)

Attribute	Description	Required or optional
<code>xml:lang</code>	The language that you want the neural voice to speak.	Required to adjust the speaking language for the neural voice. If you're using <code>lang xml:lang</code> , the locale must be provided.

(!) Note

The `<lang xml:lang>` element is incompatible with the `prosody` and `break` elements. You can't adjust pause and prosody like pitch, contour, rate, or volume in this element.

Non-multilingual voices don't support the `<lang xml:lang>` element by design.

Multilingual voices with the lang element

Use the [multilingual voices section](#) to determine which speaking languages the Speech service supports for each neural voice, as demonstrated in the following example table. If the voice doesn't speak the language of the input text, the Speech service doesn't output synthesized audio.

Voice	Auto-detected language number	Auto-detected language (locale)	All locales number	All languages (locale) supported from SSML
en-US-AndrewMultilingualNeural ¹ (Male)	77	Afrikaans (af-ZA), Albanian (sq-AL), Amharic (am-ET), Arabic (ar-EG), Armenian (hy-AM), Azerbaijani (az-AZ), Bahasa Indonesian (id-ID), Bangla (bn-BD), Basque (eu-ES), Bengali (bn-IN), Bosnian (bs-BA), Bulgarian (bg-BG), Burmese (my-MM), Catalan (ca-ES), Chinese Cantonese (zh-HK), Chinese Mandarin (zh-CN), Chinese Taiwanese (zh-TW), Croatian (hr-HR), Czech (cs-CZ), Danish (da-DK), Dutch (nl-NL), English (en-US), Estonian (et-EE), Filipino (fil-PH), Finnish (fi-FI), French (fr-FR), Galician (gl-ES), Georgian (ka-GE), German (de-DE), Greek (el-GR), Hebrew (he-IL), Hindi (hi-IN), Hungarian (hu-HU), Icelandic (is-IS), Irish (ga-IE), Italian (it-IT), Japanese (ja-JP), Javanese (jv-ID), Kannada (kn-IN), Kazakh (kk-KZ), Khmer (km-KH),	91	Afrikaans (South Africa) (af-ZA), Albanian (Albania) (sq-AL), Amharic (Ethiopia) (am-ET), Arabic (Egypt) (ar-EG), Arabic (Saudi Arabia) (ar-SA), Armenian (Armenia) (hy-AM), Azerbaijani (Azerbaijan) (az-AZ), Basque (Basque) (eu-ES), Bengali (India) (bn-IN), Bosnian (Bosnia and Herzegovina) (bs-BA), Bulgarian (Bulgaria) (bg-BG), Burmese (Myanmar) (my-MM), Catalan (Spain) (ca-ES), Chinese (Cantonese, Traditional) (zh-HK), Chinese (Mandarin, Simplified) (zh-CN), Chinese (Taiwanese Mandarin) (zh-TW), Croatian (Croatia) (hr-HR), Czech (Czech) (cs-CZ), Danish (Denmark) (da-DK), Dutch (Belgium) (nl-BE), Dutch (Netherlands) (nl-NL), English (Australia) (en-AU), English (Canada) (en-CA), English (Hong Kong SAR) (en-HK), English (India) (en-IN), English (Ireland) (en-IE), English (United Kingdom) (en-GB), English (United States) (en-US), Estonian (Estonia) (et-EE), Filipino (Philippines) (fil-PH), Finnish (Finland) (fi-FI), French (Belgium) (fr-BE), French (Canada) (fr-CA), French (France) (fr-FR), French (Switzerland) (fr-CH), Galician (Galician) (gl-ES), Georgian (Georgia) (ka-GE),
en-US-AvaMultilingualNeural ¹ (Female)				
en-US-BrianMultilingualNeural ¹ (Male)				
en-US-EmmaMultilingualNeural ¹ (Female)				

Voice	Auto-detected language number	Auto-detected language (locale)	All locales number	All languages (locale) supported from SSML
	Korean (ko-KR), Lao (lo-LA), Latvian (lv-LV), Lithuanian (lt-LT), Macedonian (mk-MK), Malay (ms-MY), Malayalam (ml-IN), Maltese (mt-MT), Mongolian (mn-MN), Nepali (ne-NP), Norwegian Bokmål (nb-NO), Pashto (ps-AF), Persian (fa-IR), Polish (pl-PL), Portuguese (pt-BR), Romanian (ro-RO), Russian (ru-RU), Serbian (sr-RS), Sinhala (si-LK), Slovak (sk-SK), Slovene (sl-SI), Somali (so-SO), Spanish (es-ES), Sundanese (su-ID), Swahili (sw-KE), Swedish (sv-SE), Tamil (ta-IN), Telugu (te-IN), Thai (th-TH), Turkish (tr-TR), Ukrainian (uk-UA), Urdu (ur-PK), Uzbek (uz-UZ), Vietnamese (vi-VN), Welsh (cy-GB), Zulu (zu-ZA)			German (Austria) (de-AT), German (Germany) (de-DE), German (Switzerland) (de-CH), Greek (Greece) (el-GR), Hebrew (Israel) (he-IL), Hindi (India) (hi-IN), Hungarian (Hungary) (hu-HU), Icelandic (Iceland) (is-IS), Indonesian (Indonesia) (id-ID), Irish (Ireland) (ga-IE), Italian (Italy) (it-IT), Japanese (Japan) (ja-JP), Javanese (Indonesia) (jv-ID), Kannada (India) (kn-IN), Kazakh (Kazakhstan) (kk-KZ), Khmer (Cambodia) (km-KH), Korean (Korea) (ko-KR), Lao (Laos) (lo-LA), Latvian (Latvia) (lv-LV), Lithuanian (Lithuania) (lt-LT), Macedonian (North Macedonia) (mk-MK), Malay (Malaysia) (ms-MY), Malayalam (India) (ml-IN), Maltese (Malta) (mt-MT), Mongolian (Mongolia) (mn-MN), Nepali (Nepal) (ne-NP), Norwegian (Bokmål, Norway) (nb-NO), Pashto (Afghanistan) (ps-AF), Persian (Iran) (fa-IR), Polish (Poland) (pl-PL), Portuguese (Brazil) (pt-BR), Portuguese (Portugal) (pt-PT), Romanian (Romania) (ro-RO), Russian (Russia) (ru-RU), Serbian (Cyrillic, Serbia) (sr-RS), Sinhala (Sri Lanka) (si-LK), Slovak (Slovakia) (sk-SK), Slovenian (Slovenia) (sl-SI), Somali (Somalia) (so-SO), Spanish (Mexico) (es-MX), Spanish (Spain) (es-ES), Sundanese (Indonesia) (su-ID), Swahili (Kenya) (sw-

Voice	Auto-detected language number	Auto-detected language (locale)	All locales number	All languages (locale) supported from SSML
				KE), Swedish (Sweden) (sv-SE), Tamil (India) (ta-IN), Telugu (India) (te-IN), Thai (Thailand) (th-TH), Turkish (Türkiye) (tr-TR), Ukrainian (Ukraine) (uk-UA), Urdu (Pakistan) (ur-PK), Uzbek (Uzbekistan) (uz-UZ), Vietnamese (Vietnam) (vi-VN), Welsh (United Kingdom) (cy-GB), Zulu (South Africa) (zu-ZA)

¹ Those are neural multilingual voices in Azure AI Speech. All multilingual voices can speak in the language in default locale of the input text without [using SSML](#). However, you can still use the `<lang xml:lang>` element to adjust the speaking accent of each language to set preferred accent such as British accent (`en-GB`) for English. The prefix in each voice name indicates its primary locale; for example, the primary locale for `en-US-AndrewMultilingualNeural` is `en-US`.

ⓘ Note

Multilingual voices don't fully support certain SSML elements, such as `break`, `emphasis`, `silence`, and `sub`.

Lang examples

For information about the supported values for attributes of the `lang` element, see [Adjust speaking language](#).

You must specify `en-US` as the default language within the `speak` element, whether or not the language is adjusted elsewhere. In this example, the primary language for `en-US-AvaMultilingualNeural` is `en-US`.

This SSML snippet shows how to use `<lang xml:lang>` to speak `de-DE` with the `en-US-AvaMultilingualNeural` neural voice.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice name="en-US-AvaMultilingualNeural">
        <lang xml:lang="de-DE">
            Wir freuen uns auf die Zusammenarbeit mit Ihnen!
        </lang>
    </voice>
</speak>
```

Within the `speak` element, you can specify multiple languages including `en-US` for text to speech output. For each adjusted language, the text must match the language and be wrapped in a `voice` element. This SSML snippet shows how to use `<lang xml:lang>` to change the speaking languages to `es-MX`, `en-US`, and `fr-FR`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice name="en-US-AvaMultilingualNeural">
        <lang xml:lang="es-MX">
            ¡Esperamos trabajar con usted!
        </lang>
        <lang xml:lang="en-US">
            We look forward to working with you!
        </lang>
        <lang xml:lang="fr-FR">
            Nous avons hâte de travailler avec vous!
        </lang>
    </voice>
</speak>
```

Adjust prosody

You can use the `prosody` element to specify changes to pitch, contour, range, rate, and volume for the text to speech output. The `prosody` element can contain text and the following elements: `audio`, `break`, `p`, `phoneme`, `prosody`, `say-as`, `sub`, and `s`.

Because prosodic attribute values can vary over a wide range, the speech recognizer interprets the assigned values as a suggestion of what the actual prosodic values of the selected voice should be. Text to speech limits or substitutes values that aren't supported. Examples of unsupported values are a pitch of 1 MHz or a volume of 120.

The following table describes the usage of the `prosody` element's attributes:

Attribute	Description	Required or optional
contour	<p>Contour represents changes in pitch. These changes are represented as an array of targets at specified time positions in the speech output. Sets of parameter pairs define each target. For example:</p> <pre><prosody contour="(0%,+20Hz) (10%,-2st) (40%,+10Hz)"></pre> <p>The first value in each set of parameters specifies the location of the pitch change as a percentage of the duration of the text. The second value specifies the amount to raise or lower the pitch by using a relative value or an enumeration value for pitch (see <code>pitch</code>). Pitch contour doesn't work on single words and short phrases. It's recommended to adjust the pitch contour on whole sentences or long phrases.</p>	Optional
pitch	<p>Indicates the baseline pitch for the text. Pitch changes can be applied at the sentence level. The pitch changes should be within 0.5 to 1.5 times the original audio. You can express the pitch as:</p> <ul style="list-style-type: none"> • An absolute value: Expressed as a number followed by "Hz" (Hertz). For example, <code><prosody pitch="600Hz">some text</prosody></code>. • A relative value: <ul style="list-style-type: none"> ◦ As a relative number: Expressed as a number preceded by "+" or "-" and followed by "Hz" or "st" that specifies an amount to change the pitch. For example: <code><prosody pitch="+80Hz">some text</prosody></code> or <code><prosody pitch="-2st">some text</prosody></code>. The "st" indicates the change unit is semitone, which is half of a tone (a half step) on the standard diatonic scale. ◦ As a percentage: Expressed as a number preceded by "+" (optionally) or "-" and followed by "%", indicating the relative change. For example: <code><prosody pitch="50%">some text</prosody></code> or <code><prosody pitch="-50%">some text</prosody></code>. • A constant value: <ul style="list-style-type: none"> ◦ <code>x-low</code> (equivalently 0.55, -45%) ◦ <code>low</code> (equivalently 0.8, -20%) ◦ <code>medium</code> (equivalently 1, default value) ◦ <code>high</code> (equivalently 1.2, +20%) ◦ <code>x-high</code> (equivalently 1.45, +45%) 	Optional
range	A value that represents the range of pitch for the text. You can express <code>range</code> by using the same absolute values, relative values, or enumeration values used to describe <code>pitch</code> .	Optional
rate	Indicates the speaking rate of the text. Speaking rate can be applied at the word or sentence level. The rate changes should be within <code>0.5</code> to <code>2</code> times the original audio. You can express <code>rate</code> as:	Optional

Attribute	Description	Required or optional
	<ul style="list-style-type: none"> • A relative value: <ul style="list-style-type: none"> ◦ As a relative number: Expressed as a number that acts as a multiplier of the default. For example, a value of <code>1</code> results in no change in the original rate. A value of <code>0.5</code> results in a halving of the original rate. A value of <code>2</code> results in twice the original rate. ◦ As a percentage: Expressed as a number preceded by "+" (optionally) or "-" and followed by "%", indicating the relative change. For example: <code><prosody rate="50%">some text</prosody></code> or <code><prosody rate="-50%">some text</prosody></code>. • A constant value: <ul style="list-style-type: none"> ◦ <code>x-slow</code> (equivalently 0.5, -50%) ◦ <code>slow</code> (equivalently 0.64, -46%) ◦ <code>medium</code> (equivalently 1, default value) ◦ <code>fast</code> (equivalently 1.55, +55%) ◦ <code>x-fast</code> (equivalently 2, +100%) 	
<code>volume</code>	<p>Indicates the volume level of the speaking voice. Volume changes can be applied at the sentence level. You can express the volume as:</p> <ul style="list-style-type: none"> • An absolute value: Expressed as a number in the range of <code>0.0</code> to <code>100.0</code>, from <i>quietest</i> to <i>loudest</i>, such as <code>75</code>. The default value is <code>100.0</code>. • A relative value: <ul style="list-style-type: none"> ◦ As a relative number: Expressed as a number preceded by "+" or "-" that specifies an amount to change the volume. Examples are <code>+10</code> or <code>-5.5</code>. ◦ As a percentage: Expressed as a number preceded by "+" (optionally) or "-" and followed by "%", indicating the relative change. For example: <code><prosody volume="50%">some text</prosody></code> or <code><prosody volume="+3%">some text</prosody></code>. • A constant value: <ul style="list-style-type: none"> ◦ <code>silent</code> (equivalently 0) ◦ <code>x-soft</code> (equivalently 0.2) ◦ <code>soft</code> (equivalently 0.4) ◦ <code>medium</code> (equivalently 0.6) ◦ <code>loud</code> (equivalently 0.8) ◦ <code>x-loud</code> (equivalently 1, default value) 	Optional

Prosody examples

For information about the supported values for attributes of the `prosody` element, see [Adjust prosody](#).

Change speaking rate example

This SSML snippet illustrates how the `rate` attribute is used to change the speaking rate to 30% greater than the default rate.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <prosody rate="+30.00%">
      Enjoy using text to speech.
    </prosody>
  </voice>
</speak>
```

Change volume example

This SSML snippet illustrates how the `volume` attribute is used to change the volume to 20% greater than the default volume.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <prosody volume="+20.00%">
      Enjoy using text to speech.
    </prosody>
  </voice>
</speak>
```

Change pitch example

This SSML snippet illustrates how the `pitch` attribute is used so that the voice speaks in a high pitch.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    Welcome to <prosody pitch="high">Enjoy using text to speech.</prosody>
  </voice>
</speak>
```

Change pitch contour example

This SSML snippet illustrates how the `contour` attribute is used to change the contour.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <prosody contour="(60%,-60%) (100%,+80%)">
      Were you the only person in the room?
    </prosody>
  </voice>
</speak>
```

Adjust emphasis

You can use the optional `emphasis` element to add or remove word-level stress for the text.

This element can only contain text and the following elements: `audio`, `break`, `emphasis`, `lang`, `phoneme`, `prosody`, `say-as`, `sub`, and `voice`.

ⓘ Note

The word-level emphasis tuning is only available for these neural voices: `en-US-GuyNeutral`, `en-US-DavisNeutral`, and `en-US-JaneNeutral`.

For words that have low pitch and short duration, the pitch might not be raised enough to be noticed.

The following table describes the `emphasis` element's attributes:

[] Expand table

Attribute	Description	Required or optional
<code>level</code>	<p>Indicates the strength of emphasis to be applied:</p> <ul style="list-style-type: none">• <code>reduced</code>• <code>none</code>• <code>moderate</code>• <code>strong</code>	Optional

When the `level` attribute isn't specified, the default level is `moderate`. For details on each attribute, see [emphasis element ↗](#).

Emphasis examples

For information about the supported values for attributes of the `emphasis` element, see [Adjust emphasis](#).

This SSML snippet demonstrates how you can use the `emphasis` element to add moderate level emphasis for the word "meetings."

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
  <voice name="en-US-AndrewMultilingualNeural">
    I can help you join your <emphasis level="moderate">meetings</emphasis> fast.
  </voice>
</speak>
```

Add recorded audio

The `audio` element is optional. You can use it to insert prerecorded audio into an SSML document. The body of the `audio` element can contain plain text or SSML markup spoken if the audio file is unavailable or unplayable. The `audio` element can also contain text and the following elements: `audio`, `break`, `p`, `s`, `phoneme`, `prosody`, `say-as`, and `sub`.

Any audio included in the SSML document must meet these requirements:

- The audio file must be valid `*.mp3`, `*.wav`, `*.opus`, `*.ogg`, `*.flac`, or `*.wma` files.
- The combined total time for all text and audio files in a single response can't exceed 600 seconds.
- The audio must not contain any customer-specific or other sensitive information.

! Note

The `audio` element isn't supported by the [Long Audio API](#). For long-form text to speech, use the [batch synthesis API](#) instead.

The following table describes the usage of the `audio` element's attributes:

 [Expand table](#)

Attribute	Description	Required or optional
src	The URI location of the audio file. The audio must be hosted on an internet-accessible HTTPS endpoint. HTTPS is required. The domain hosting the file must present a valid, trusted TLS/SSL certificate. You should put the audio file into Blob Storage in the same Azure region as the text to speech endpoint to minimize the latency.	Required

Audio examples

For information about the supported values for attributes of the `audio` element, see [Add recorded audio](#).

This SSML snippet illustrates how to use `src` attribute to insert audio from two .wav files.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <p>
      <audio src="https://contoso.com/opinionprompt.wav"/>
      Thanks for offering your opinion. Please begin speaking after the
      beep.
      <audio src="https://contoso.com/beep.wav">
        Could not play the beep, please voice your opinion now.
      </audio>
    </p>
  </voice>
</speak>
```

Adjust the audio duration

Use the `mstts:audioduration` element to set the duration of the output audio. Use this element to help synchronize the timing of audio output completion. The audio duration can be decreased or increased between `0.5` to `2` times the rate of the original audio. The original audio is the audio without any other rate settings. The speaking rate is slowed down or sped up accordingly based on the set value.

The audio duration setting applies to all input text within its enclosing `voice` element. To reset or change the audio duration setting again, you must use a new `voice` element with either the same voice or a different voice.

The following table describes the usage of the `mstts:audioduration` element's attributes:

Attribute	Description	Required or optional
value	<p>The requested duration of the output audio in either seconds, such as <code>2s</code>, or milliseconds, such as <code>2000ms</code>.</p> <p>The maximum value for output audio duration is 300 seconds. This value should be within <code>0.5</code> to <code>2</code> times the original audio without any other rate settings. For example, if the requested duration of your audio is <code>30s</code>, then the original audio must otherwise be between 15 and 60 seconds. If you set a value outside of these boundaries, the duration is set according to the respective minimum or maximum multiple. For output audio longer than 300 seconds, first generate the original audio without any other rate settings, then calculate the rate to adjust using the prosody rate to achieve the desired duration.</p>	Required

mstts audio duration examples

For information about the supported values for attributes of the `mstts:audioduration` element, see [Adjust the audio duration](#).

In this example, the original audio is around 15 seconds. The `mstts:audioduration` element is used to set the audio duration to 20 seconds or `20s`.

XML

```

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <mstts:audioduration value="20s"/>
    If we're home schooling, the best we can do is roll with what each day brings and
    try to have fun along the way.
    A good place to start is by trying out the slew of educational apps that are
    helping children stay happy and smash their schooling at the same time.
  </voice>
</speak>
```

Add background audio

You can use the `mstts:backgroundaudio` element to add background audio to your SSML documents or mix an audio file with text to speech. With `mstts:backgroundaudio`, you can loop an audio file in the background, fade in at the beginning of text to speech, and fade out at the end of text to speech.

If the background audio provided is shorter than the text to speech or the fade out, it loops. If it's longer than the text to speech, it stops when the fade out is finished.

Only one background audio file is allowed per SSML document. You can intersperse `audio` tags within the `voice` element to add more audio to your SSML document.

! Note

The `mstts:backgroundaudio` element should be put in front of all `voice` elements. If specified, it must be the first child of the `speak` element.

The `mstts:backgroundaudio` element isn't supported by the [Long Audio API](#). For long-form text to speech, use the [batch synthesis API](#) (Preview) instead.

The following table describes the usage of the `mstts:backgroundaudio` element's attributes:

[+] Expand table

Attribute	Description	Required or optional
<code>src</code>	The URI location of the background audio file.	Required
<code>volume</code>	The volume of the background audio file. Accepted values: <code>0</code> to <code>100</code> inclusive. The default value is <code>1</code> .	Optional
<code>fadein</code>	The duration of the background audio fade-in as milliseconds. The default value is <code>0</code> , which is the equivalent to no fade in. Accepted values: <code>0</code> to <code>10000</code> inclusive.	Optional
<code>fadeout</code>	The duration of the background audio fade-out in milliseconds. The default value is <code>0</code> , which is the equivalent to no fade out. Accepted values: <code>0</code> to <code>10000</code> inclusive.	Optional

mstts backgroundaudio examples

For information about the supported values for attributes of the `mstts:backgroundaudio` element, see [Add background audio](#).

XML

```
<speak version="1.0" xml:lang="en-US" xmlns:mstts="http://www.w3.org/2001/mstts">
    <mstts:backgroundaudio src="https://contoso.com/sample.wav" volume="0.7"
    fadein="3000" fadeout="4000"/>
    <voice name="en-US-AvaMultilingualNeural">
```

The text provided in this document are spoken over the background audio.

```
</voice>
</speak>
```

Viseme element

A viseme is the visual description of a phoneme in spoken language. It defines the position of the face and mouth while a person is speaking. You can use the `mstts:viseme` element in SSML to request viseme output. For more information, see [Get facial position with viseme](#).

The viseme setting is applied to all input text within its enclosing `voice` element. To reset or change the viseme setting again, you must use a new `voice` element with either the same voice or a different voice.

Usage of the `viseme` element's attributes are described in the following table.

[+] Expand table

Attribute	Description	Required or optional
<code>type</code>	<p>The type of viseme output.</p> <ul style="list-style-type: none">• <code>redlips_front</code> – lip-sync with viseme ID and audio offset output• <code>FacialExpression</code> – blend shapes output	Required

ⓘ Note

Currently, `redlips_front` only supports neural voices in `en-US` locale, and `FacialExpression` supports neural voices in `en-US` and `zh-CN` locales.

Viseme examples

The supported values for attributes of the `viseme` element were [described previously](#).

This SSML snippet illustrates how to request blend shapes with your synthesized speech.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice name="en-US-AvaNeural">
```

```

<mstts:viseme type="FacialExpression"/>
    Rainbow has seven colors: Red, orange, yellow, green, blue, indigo, and
    violet.
</voice>
</speak>

```

Voice conversion element

Voice conversion (preview) is the process of transforming the voice characteristics of a given audio to a target voice speaker. After voice conversion, the resulting audio reserves source audio's linguistic content and prosody while the voice timbre sounds like the target speaker. For more information, see [voice conversion](#).

Use the `<mstts:voiceconversion>` tag via Speech Synthesis Markup Language (SSML) to specify the source audio URL and the target voice for the conversion. For a complete list of supported target voices, see [supported voices for voice conversion](#).

The following table describes the usage of the `mstts:voiceconversion` element's attributes:

[] [Expand table](#)

Attribute	Description	Required or optional
<code>url</code>	<p>The URL of the source audio file that provides linguistic content and prosody for the synthesized speech.</p> <p>The <code>url</code> must be accessible via HTTPS URL. For example, <code>https://example.com/source.wav</code></p> <p>Input audio must be under 100 MB.</p>	Required

Here's how the voice conversion works:

- The source audio is a prerecorded audio file that contains the spoken words and prosody.
 - Text content: The final synthesized speech follows the spoken words in the source audio.
 - Prosody and rhythm: The speech maintains the timing and intonation from the source.
- The `<voice>` tag specifies the target voice used for the output audio. For information about the supported target voices, see [supported voices for voice conversion](#).
- The output audio keeps the timbre (tone and voice quality) of the target voice, but follows the text and speaking style of the source audio.

! Note

All SSML elements related to prosody and pronunciation such as `<prosody>` or `<mstts:express-as>` are ignored.

Text input is optional and any text included in the SSML are ignored during rendering.

mstss voiceconversion examples

The following example demonstrates how to use `<mstts:voiceconversion>` to synthesize speech using a **target neural voice** while **matching both the content and prosody** of a given source audio:

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice xml:lang="en-US" xml:gender="Female" name="en-US-
      AvaMultilingualNeural">
      <mstts:voiceconversion
        url="https://your.blob.core.windows.net/sourceaudio.wav"/>
    </voice>
</speak>
```

Next steps

- [SSML overview](#)
- [SSML document structure and events](#)
- [Language and voice support for the Speech service](#)

Pronunciation with SSML

09/26/2025

You can use Speech Synthesis Markup Language (SSML) with text to speech to specify how the speech is pronounced. For example, you can use SSML with phonemes and a custom lexicon to improve pronunciation. You can also use SSML to define how a word or mathematical expression is pronounced.

Refer to the following sections for details about how to use SSML elements to improve pronunciation. For more information about SSML syntax, see [SSML document structure and events](#).

phoneme element

The `phoneme` element is used for phonetic pronunciation in SSML documents. Always provide human-readable speech as a fallback.

Phonetic alphabets are composed of phones, which are made up of letters, numbers, or characters, sometimes in combination. Each phone describes a unique sound of speech. The phonetic alphabet is in contrast to the Latin alphabet, where any letter might represent multiple spoken sounds. Consider the different `en-US` pronunciations of the letter "c" in the words "candy" and "cease" or the different pronunciations of the letter combination "th" in the words "thing" and "those."

 **Note**

For a list of locales that support phonemes, see footnotes in the [language support](#) table.

Usage of the `phoneme` element's attributes are described in the following table.

 [Expand table](#)

Attribute	Description	Required or optional
<code>alphabet</code>	The phonetic alphabet to use when you synthesize the pronunciation of the string in the <code>ph</code> attribute. The string that specifies the alphabet must be specified in lowercase letters. The following options are the possible alphabets that you can specify: <ul style="list-style-type: none">• <code>ipa</code> – See SSML phonetic alphabets• <code>sapi</code> – See SSML phonetic alphabets	Optional

Attribute	Description	Required or optional
	<ul style="list-style-type: none"> • <code>ups</code> – See Universal Phone Set ↗ • <code>x-sampa</code> – See SSML phonetic alphabets <p>The alphabet applies only to the <code>phoneme</code> in the element.</p>	
<code>ph</code>	<p>A string containing phones that specify the pronunciation of the word in the <code>phoneme</code> element. If the specified string contains unrecognized phones, text-to-speech rejects the entire SSML document and produces none of the speech output specified in the document.</p> <p>For <code>ipa</code>, to stress one syllable by placing stress symbol before this syllable, you need to mark all syllables for the word. Or else, the syllable before this stress symbol is stressed. For <code>sapi</code>, if you want to stress one syllable, you need to place the stress symbol after this syllable, whether or not all syllables of the word are marked.</p>	Required

phoneme examples

The supported values for attributes of the `phoneme` element were [described previously](#). In the first two examples, the values of `ph="tə.'meɪ.tou"` or `ph="tə'meɪ'tou"` are specified to stress the syllable `meɪ`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <phoneme alphabet="ipa" ph="tə.'meɪ.tou"> tomato </phoneme>
  </voice>
</speak>
```

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <phoneme alphabet="ipa" ph="tə'meɪ'tou"> tomato </phoneme>
  </voice>
</speak>
```

XML

```
<speak version="1.0" xmlns="https://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
```

```
<phoneme alphabet="sapi" ph="iy eh n y uw eh s"> en-US </phoneme>
</voice>
</speak>
```

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <s>His name is Mike <phoneme alphabet="ups" ph="JH AU"> Zhou </phoneme>
  </s>
  </voice>
</speak>
```

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <phoneme alphabet='x-sampa' ph='he."lou'>hello</phoneme>
  </voice>
</speak>
```

Custom lexicon

You can define how single entities (such as company, a medical term, or an emoji) are read in SSML by using the `phoneme` and `sub` elements. To define how multiple entities are read, create an XML structured custom lexicon file. Then you upload the custom lexicon XML file and reference it with the SSML `lexicon` element.

ⓘ Note

For a list of locales that support custom lexicon, see footnotes in the [language support](#) table.

The `lexicon` element isn't supported by the [Long Audio API](#). For long-form text to speech, use the [batch synthesis API](#) (Preview) instead.

Usage of the `lexicon` element's attributes are described in the following table.

[] [Expand table](#)

Attribute	Description	Required or optional
uri	The URI of the publicly accessible custom lexicon XML file with either the <code>.xml</code> or <code>.pls</code> file extension. Using Azure Blob Storage is recommended but not required. For more information about the custom lexicon file, see Pronunciation Lexicon Specification (PLS) Version 1.0 .	Required

Custom lexicon examples

The supported values for attributes of the `lexicon` element were [described previously](#).

After you publish your custom lexicon, you can reference it from your SSML. The following SSML example references a custom lexicon that was uploaded to `https://www.example.com/customlexicon.xml`. We support lexicon URLs from Azure Blob Storage. However, note that other public URLs may not be compatible.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts"
      xml:lang="en-US">
  <voice name="en-US-AvaNeural">
    <lexicon uri="https://www.example.com/customlexicon.xml"/>
    BTW, we will be there probably at 8:00 tomorrow morning.
    Could you help leave a message to Robert Benigni for me?
  </voice>
</speak>
```

Custom lexicon file

To define how multiple entities are read, you can define them in a custom lexicon XML file with either the `.xml` or `.pls` file extension.

ⓘ Note

The custom lexicon file is a valid XML document, but it can't be used as an SSML document.

Here are some limitations of the custom lexicon file:

- **File size:** The custom lexicon file size is limited to a maximum of 100 KB. If the file size exceeds the 100-KB limit, the synthesis request fails. You can split your lexicon into

multiple lexicons and include them in SSML if the file size exceeds 100 KB.

- **Lexicon cache refresh:** The custom lexicon is cached with the URI as the key on text to speech when it's first loaded. The lexicon with the same URI isn't reloaded within 15 minutes, so the custom lexicon change needs to wait 15 minutes at the most to take effect.

The supported elements and attributes of a custom lexicon XML file are described in the [Pronunciation Lexicon Specification \(PLS\) Version 1.0](#). Here are some examples of the supported elements and attributes:

- The `lexicon` element contains at least one `lexeme` element. Lexicon contains the necessary `xml:lang` attribute to indicate which locale it should be applied for. One custom lexicon is limited to one locale by design, so if you apply it for a different locale, it doesn't work. The `lexicon` element also has an `alphabet` attribute to indicate the alphabet used in the lexicon. The possible values are `ipa` and `x-microsoft-sapi`.
- Each `lexeme` element contains at least one `grapheme` element and one or more `grapheme`, `alias`, and `phoneme` elements. The `lexeme` element is case sensitive in the custom lexicon. For example, if you only provide a phoneme for the `lexeme` "Hello", it doesn't work for the `lexeme` "hello".
- The `grapheme` element contains text that describes the [orthography](#).
- The `alias` elements are used to indicate the pronunciation of an acronym or an abbreviated term.
- The `phoneme` element provides text that describes how the `lexeme` is pronounced. The syllable boundary is '.' in the IPA alphabet. The `phoneme` element can't contain white space when you use the IPA alphabet.
- When the `alias` and `phoneme` elements are provided with the same `grapheme` element, `alias` has higher priority.

Microsoft provides a [validation tool for the custom lexicon](#) that helps you find errors (with detailed error messages) in the custom lexicon file. Using the tool is recommended before you use the custom lexicon XML file in production with the Speech service.

Custom lexicon file examples

The following XML example (not SSML) would be contained in a custom lexicon `.xml` file. When you use this custom lexicon, "BTW" is read as "By the way." "Benigni" is read with the provided IPA "be'ni:nji."

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
    xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
    alphabet="ipa" xml:lang="en-US">
<lexeme>
    <grapheme>BTW</grapheme>
    <alias>By the way</alias>
</lexeme>
<lexeme>
    <grapheme>Benigni</grapheme>
    <phoneme>bɛ'ni:nji</phoneme>
</lexeme>
<lexeme>
    <grapheme>😊</grapheme>
    <alias>test emoji</alias>
</lexeme>
</lexicon>

```

You can't directly set the pronunciation of a phrase by using the custom lexicon. If you need to set the pronunciation for an acronym or an abbreviated term, first provide an `alias`, and then associate the `phoneme` with that `alias`. For example:

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
    xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
        http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
    alphabet="ipa" xml:lang="en-US">
<lexeme>
    <grapheme>Scotland MV</grapheme>
    <alias>ScotlandMV</alias>
</lexeme>
<lexeme>
    <grapheme>ScotlandMV</grapheme>
    <phoneme>'skɒtlənd.'mi:dɪəm.weɪv</phoneme>
</lexeme>
</lexicon>

```

You could also directly provide your expected `alias` for the acronym or abbreviated term. For example:

XML

```
<lexeme>
  <grapheme>Scotland MV</grapheme>
  <alias>Scotland Media Wave</alias>
</lexeme>
```

The preceding custom lexicon XML file examples use the IPA alphabet, which is also known as the IPA phone set. We suggest that you use the IPA because it's the international standard. For some IPA characters, they're the "precomposed" and "decomposed" version when they're being represented with Unicode. The custom lexicon only supports the decomposed Unicode.

The Speech service defines a phonetic set for these locales: `en-US`, `fr-FR`, `de-DE`, `es-ES`, `ja-JP`, `zh-CN`, `zh-HK`, and `zh-TW`. For more information on the detailed Speech service phonetic alphabet, see the [Speech service phonetic sets](#).

You can use the `x-microsoft-sapi` as the value for the `alphabet` attribute with custom lexicons as demonstrated here:

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
  xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2005/01/pronunciation-lexicon
    http://www.w3.org/TR/2007/CR-pronunciation-lexicon-20071212/pls.xsd"
  alphabet="x-microsoft-sapi" xml:lang="en-US">
  <lexeme>
    <grapheme>BTW</grapheme>
    <alias> By the way </alias>
  </lexeme>
  <lexeme>
    <grapheme> Benigni </grapheme>
    <phoneme> b eh 1 - n iy - n y iy </phoneme>
  </lexeme>
</lexicon>
```

say-as element

The `say-as` element indicates the content type, such as number or date, of the element's text. This element provides guidance to the speech synthesis engine about how to pronounce the text.

Usage of the `say-as` element's attributes are described in the following table.

[+] Expand table

Attribute	Description	Required or optional
<code>interpret-as</code>	Indicates the content type of an element's text. For a list of types, see the following table.	Required
<code>format</code>	Provides additional information about the precise formatting of the element's text for content types that might have ambiguous formats. SSML defines formats for content types that use them. See the following table.	Optional
<code>detail</code>	Indicates the level of detail to be spoken. For example, this attribute might request that the speech synthesis engine pronounce punctuation marks. There are no standard values defined for <code>detail</code> .	Optional

The following content types are supported for the `interpret-as` and `format` attributes. Include the `format` attribute only if `format` column isn't empty in this table.

(!) Note

The `characters` and `spell-out` values for the `interpret-as` attribute are supported for all [text to speech locales](#). Other `interpret-as` attribute values are supported for all locales of the following languages: Arabic, Catalan, Chinese, Danish, Dutch, English, French, Finnish, German, Hindi, Italian, Japanese, Korean, Norwegian, Polish, Portuguese, Russian, Spanish, and Swedish.

[+] Expand table

interpret-as	format	Interpretation
<code>characters</code> , <code>spell-out</code>		<p>The text is spoken as individual letters (spelled out). The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="characters">test</say-as></pre> <p>As "T E S T."</p>
<code>cardinal</code> , <code>number</code>	None	<p>The text is spoken as a cardinal number. The speech synthesis engine pronounces:</p> <pre>There are <say-as interpret-as="cardinal">10</say-as> options</pre> <p>As "There are ten options."</p>

interpret-as	format	Interpretation
ordinal	None	<p>The text is spoken as an ordinal number. The speech synthesis engine pronounces:</p> <pre>Select the <say-as interpret-as="ordinal">3rd</say-as> option</pre> <p>As "Select the third option."</p>
number_digit	None	<p>The text is spoken as a sequence of individual digits. The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="number_digit">123456789</say-as></pre> <p>As "1 2 3 4 5 6 7 8 9."</p>
fraction	None	<p>The text is spoken as a fractional number. The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="fraction">3/8</say-as> of an inch</pre> <p>As "three eighths of an inch."</p>
date	dmy, mdy, ymd, ydm, ym, my, md, dm, d, m, y	<p>The text is spoken as a date. The <code>format</code> attribute specifies the date's format (<i>d=day, m=month, and y=year</i>). The speech synthesis engine pronounces:</p> <pre>Today is <say-as interpret-as="date">10-12-2016</say-as></pre> <p>As "Today is October twelfth two thousand sixteen." Pronounces:</p> <pre>Today is <say-as interpret-as="date" format="dmy">10-12-2016</say-as></pre> <p>As "Today is December tenth two thousand sixteen."</p>
time	hms12, hms24	<p>The text is spoken as a time. The <code>format</code> attribute specifies whether the time is specified by using a 12-hour clock (hms12) or a 24-hour clock (hms24). Use a colon to separate numbers representing hours, minutes, and seconds. Here are some valid time examples: 12:35, 1:14:32, 08:15, and 02:50:45. The speech synthesis engine pronounces:</p> <pre>The train departs at <say-as interpret-as="time" format="hms12">4:00am</say-as></pre> <p>As "The train departs at four A M."</p>

interpret-as	format	Interpretation
duration	hms, hm, ms	<p>The text is spoken as a duration. The <code>format</code> attribute specifies the duration's format (<i>h=hour, m=minute, and s=second</i>). The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="duration">01:18:30</say-as></pre> <p>As "one hour eighteen minutes and thirty seconds". Pronounces:</p> <pre><say-as interpret-as="duration" format="ms">01:18</say-as></pre> <p>As "one minute and eighteen seconds". This tag is only supported on English and Spanish.</p>
telephone	None	<p>The text is spoken as a telephone number. The speech synthesis engine pronounces:</p> <pre>The number is <say-as interpret-as="telephone">(888) 555-1212</say-as></pre> <p>As "My number is area code eight eight eight five five five one two one two."</p>
currency	None	<p>The text is spoken as a currency. The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="currency">99.9 USD</say-as></pre> <p>As "ninety-nine US dollars and ninety cents."</p>
address	None	<p>The text is spoken as an address. The speech synthesis engine pronounces:</p> <pre>I'm at <say-as interpret-as="address">150th CT NE, Redmond, WA</say-as></pre> <p>As "I'm at 150th Court Northeast Redmond Washington."</p>
name	None	<p>The text is spoken as a person's name. The speech synthesis engine pronounces:</p> <pre><say-as interpret-as="name">ED</say-as></pre> <p>As [æd].</p> <p>In Chinese names, some characters pronounce differently when they appear in a family name. For example, the speech synthesis engine says 仇 in</p>

interpret-as	format	Interpretation
		<pre><say-as interpret-as="name">仇先生</say-as></pre> <p>As [qiú] instead of [chóu].</p>

say-as examples

The supported values for attributes of the `say-as` element were [described previously](#).

The speech synthesis engine speaks the following example as "Your first request was for one room on October nineteenth twenty ten with early arrival at twelve thirty five PM."

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <p>
      Your <say-as interpret-as="ordinal"> 1st </say-as> request was for <say-as
      interpret-as="cardinal"> 1 </say-as> room
      on <say-as interpret-as="date" format="mdy"> 10/19/2010 </say-as>, with
      early arrival at <say-as interpret-as="time" format="hms12"> 12:35pm </say-as>.
    </p>
  </voice>
</speak>
```

sub element

Use the `sub` element to indicate that the alias attribute's text value should be pronounced instead of the element's enclosed text. In this way, the SSML contains both a spoken and written form.

Usage of the `sub` element's attributes are described in the following table.

 [Expand table](#)

Attribute	Description	Required or optional
<code>alias</code>	The text value that should be pronounced instead of the element's enclosed text.	Required

sub examples

The supported values for attributes of the `sub` element were described previously.

The speech synthesis engine speaks the following example as "World Wide Web Consortium."

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <sub alias="World Wide Web Consortium">W3C</sub>
  </voice>
</speak>
```

Mathematical expressions reading

There are two ways to read a mathematical expression:

- With Math domain element,

Embed the plain text mathematical expression directly in SSML and specify the math domain using `<mstts:prompt domain="Math" />`.

See the section: [Reading plain text mathematical expressions](#)

- With MathML elements

Represent the mathematical expression with MathML elements.

See the section: [Reading mathematical expressions with MathML](#)

! Note

The two features are currently supported in the following locales: de-DE, en-AU, en-GB, en-US, all the sibling locales of English, es-ES, es-MX, all the sibling locales of Spanish, fr-CA, fr-FR, it-IT, ja-JP, ko-KR, pt-BR, and zh-CN.

Reading plain text mathematical expressions

To enable complex mathematical expression reading, you can add `<mstts:prompt domain="Math" />` element to enable math-specific pronunciation rules.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
```

```

<voice name="en-US-AvaMultilingualNeural">
  <mstts:prompt domain="Math" />
  x = (-b ± √(b² - 4ac)) / 2a
</voice>
</speak>

```

By default, parentheses aren't read out in mathematical expressions. If you'd like the parentheses read out, you can specify `<mstts:mathspeechverbosity level="verbose" />` in SSML

XML

```

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <mstts:prompt domain="Math" /><mstts:mathspeechverbosity level="verbose" />
    x = (-b ± √(b² - 4ac)) / 2a
  </voice>
</speak>

```

If you'd like the expression read out in other language with a multilingual voice, specify lang element in SSML.

XML

```

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
  xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
  <voice name="en-US-AvaMultilingualNeural">
    <mstts:prompt domain="Math" />
    <lang xml:lang="es-ES">x = (-b ± √(b² - 4ac)) / 2a</lang>
  </voice>
</speak>

```

Reading mathematical expressions with MathML

The Mathematical Markup Language (MathML) is an XML-compliant markup language that describes mathematical content and structure. The Speech service can use the MathML as input text to properly pronounce mathematical notations in the output audio.

All elements from the [MathML 2.0](#) and [MathML 3.0](#) specifications are supported, except the MathML 3.0 [Elementary Math](#) elements.

Take note of these MathML elements and attributes:

- The `xmlns` attribute in `<math xmlns="http://www.w3.org/1998/Math/MathML">` is optional.

- The `semantics`, `annotation`, and `annotation-xml` elements don't output speech, so they're ignored.
- If an element isn't recognized, it'll be ignored, but the child elements within it will still be processed.

The XML syntax doesn't support the MathML entities, so you must use the corresponding [unicode characters ↗](#) to represent the entities, for example, the entity `©` should be represented by its unicode characters `©`, otherwise an error occurs.

MathML examples

The text to speech output for this example is "a squared plus b squared equals c squared".

```
XML

<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
  xmlns:mstts='http://www.w3.org/2001/mstts' xml:lang='en-US'>
  <voice name='en-US-JennyNeural'>
    <math xmlns='http://www.w3.org/1998/Math/MathML'>
      <msup>
        <mi>a</mi>
        <mn>2</mn>
      </msup>
      <mo>+</mo>
      <msup>
        <mi>b</mi>
        <mn>2</mn>
      </msup>
      <mo>=</mo>
      <msup>
        <mi>c</mi>
        <mn>2</mn>
      </msup>
    </math>
  </voice>
</speak>
```

Next steps

- [SSML overview](#)
- [SSML document structure and events](#)
- [Language support: Voices, locales, languages](#)

SSML phonetic alphabets

08/07/2025

Phonetic alphabets are used with the [Speech Synthesis Markup Language \(SSML\)](#) to improve the pronunciation of text to speech voices. To learn when and how to use each alphabet, see [Use phonemes to improve pronunciation](#).

Speech service supports the [International Phonetic Alphabet \(IPA\)](#) ↗ suprasegmentals that are listed here. You set `ipa` as the `alphabet` in [SSML](#).

 [Expand table](#)

ipa	Symbol	Note
'	Primary stress	Don't use single quote (‘ or ’) though they look similar.
,	Secondary stress	Don't use comma (,) though it looks similar.
.	Syllable boundary	
:	Long	Don't use colon (:) or (:) though they look similar.
()	Linking	

Tip

You can use [the international phonetic alphabet keyboard](#) ↗ to create the correct `ipa` suprasegmentals.

For some locales, Speech service defines its own phonetic alphabets, which ordinarily map to the [International Phonetic Alphabet \(IPA\)](#) ↗. The eight locales that support the Microsoft Speech API (SAPI, or `sapi`) are en-US, fr-FR, de-DE, es-ES, ja-JP, zh-CN, zh-HK, and zh-TW. For those eight locales, you set `sapi` or `ipa` as the `alphabet` in [SSML](#).

See the sections in this article for the phonemes that are specific to each locale.

Note

The following tables list viseme IDs corresponding to phonemes for different locales. When viseme ID is 0, it indicates silence.

ar-EG/ar-SA

Vowels for ar-EG

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2		حرب	شَرَد
a:	2		لِسَانَهُ	أَدْيَانَهَا
i	6		رَجْلٌ	بِهِ
i:	6		كَبِيرٌ	أَزْدَوازِي
u	7		آجُّزٌ	مُنْذُ
u:	7		آخَرُونَ	أَعْاقُوا

Consonant for ar-EG

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	بازار	أب	أغطاب
d	19	دابة	أباد	أفسد
g	20	جمل	رجب	مرح
k	20	آخر	أكل	أوراك
t	19	تأخي	مُرثٰب	أخت
d ^č	19	ضاعن	مُضيّر	مرض
q	20	قمر	اتّعاقد	أرق
t ^č	19	طالبان	احاطوه	ربط
? [?]	19	أُفقي	فأْز	السماء
f	18	فن	يفُر	شرف
h	12	هياج	أَحَبَّهُ	الأَلْهَى

ipa	viseme	Example 1	Example 2	Example 3
ħ	12	حُرْ	رَحْم	الْبَلْج
s	15	سِرْ	قِسْمٌ	الْجَالِس
θ	19	ثَرَى	أَخْدَاثٌ	الْحَارَث
z	15	زَرْ	قَزْمٌ	الْحُبْز
ð	17	ظَنْ	أَظْهَرَ	قَيْط
ð	17	ذَلِكَ	أَدَّى	أَنْفَذْ
ɣ	20	غَنِيٌّ	أَدْمَعَةٌ	دِماغ
x	12	خَبْرٌ	رَخْوٌ	أَخْ
ʃ	16	شَافَتْ	خَشِيَّةٌ	وَحْش
s̪	15	صِلَهْ	وَصَلَ	عَوْص
j	6	يَوْمٌ	أَدْوَيَهْ	يَمْدِيرِي
w	7	وَلَدْ	لَوْنٌ	ما كا و
l	14	لَيْسَ	عَلْمٌ	عَمَل
m	21	مَجْدُ	ثَمَنْ	قَلْم
n	19	نَائِبُ	أَدْيَانًا	أَمْن
r	13	رَكْبُ	قَرْنُ	يَمْكَرَر
ɾ	12	عَامَّا	لَعِبْ	بَيْع

bg-BG

Vowels for bg-BG

[+] Expand table

ipa	viseme	Example 1	Example 2	Example 3
i	6	искат	сценаристката	лечими
ɛ	4	елен	цена	оспорените

ipa	viseme	Example 1	Example 2	Example 3
ɔ	3	оправя	което	колело
а	2	ангел	докосват	цена
u	7	уклончивата	хубавичка	табу
ја	6,2	ябълка	оял	залај
ұ	1	ъгъл	ъгъл	имамбаялдъ
յи	6,7	ютия	отвоювал	стою

Consonant for bg-BG

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
n	19	неразбирамия	преразпределените	искан
ʒ	16	жълт	тъжен	таралеж
k	20	камък	вкарвал	петък
tʂ	19,15	цар	меценат	царевец
t	19	тайна	натъжени	прост
p	21	приказката	натъпкан	прилеп
r	13	разлика	прозорец	хитър
s	15	сърна	месец	процес
d	19	дом	меден	джаред
x	12	хора	доход	цех
z ^j	15	зян	Замразявайки	
l ^j	14	любов	влюблуване	
l	14	лебед	блед	хмел
n ^j	19	някога	понякога	
v	18	време	навреме	лъвов

ipa	viseme	Example 1	Example 2	Example 3
m	21	море	време	корем
b	21	баба	риба	ястреб
g	20	гарван	наруган	драг
dʒ	19,16	Джорджева	тютюнджийский	пледж
f	18	филм	реформа	релеф
m̥	21	мях	смях	
t̥	19	тяхна	стяга	
r̥	13	рядка	впряженки	
p̥	21	пяхме	спя	
d̥	19	дядовите	задявайки	
j	6	йод	пейо	ручей
v̥	18	вяло	посивяло	
s̥	15	сякаш	всяка	
b̥	21	бяла	избягваме	
k̥	20	кюпа	прокурорският	
g̥	20	гюргево	панагюрското	
f̥	18	фючърс	Парфюмерия	
z	15	зима	изразходват	рассказ
ʃ	16	шума	машина	изпечеш
tʃ	19,16	чак	случай	меч
dʒ	19,15	дзифт	скрънда	Лодз

ca-ES

Vowels for ca-ES

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2	amen	amaro	està
ɔ	3	odre	ofertori	microtò
ə	1	estan	seré	aigua
e	4	érem	feta	seré
ɛ	4	ecosistema	incorrecta	haver
i	6	itinerants	itinerants	zombi
o	8	ombra	retondre	omissió
u	7	universitaris	candidatures	crono

Consonant for ca-ES

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	baba	dobra	
β	21	vià	baba	
tʃ	19,16	txadià	matxucs	faig
d	19	dedicada	conduïa	navidad
ð	17	The_Sun	dedicada	trinidad
f	18	facilitades	affectarà	àgraf
g	20	gracia	congratula	
ɣ	20		aigua	
j	6	hiena	esplaia	cofoi
ðʒ	19,16	djakarta	compostatge	george
k	20	curós	dodecà	doblec
l	14	laberint	miolar	preväl
ʎ	14	lligada	millorarà	perbull

ipa	viseme	Example 1	Example 2	Example 3
m	21	macadàmies	femar	sublim
n	19	necessaris	sanitaris	alterament
ŋ	20		algonquí	albenc
ɲ	19	nyasa	remenjar	alemany
p	21	pegues	estepa	cap
r	19		caro	càrter
r	13	rabada	carro	lofòfor
s	15	ceri	cursar	cus
ʃ	16	xacar	microxip	midraix
t	19	tabacaires	estratifica	debatut
θ	19	ceará	vecinos	Álvarez
w	7	westfalià	inaugurar	inscriu
x	12	juanita	mujeres	heinrich
z	15	zelar	brasils	alianze
ʒ	16	gebrada	ascogènic	orange

cs-CZ

Vowels for cs-CZ

[] Expand table

ipa	viseme	Example 1	Example 2	Example 3
ɪ	6	ideální	handicapované	giganty
ɛ	4	efektní	grotesce	hygieně
a	2	agens	infarktu	extra
o	8	oáz	hřbitovy	čisto

ipa	viseme	Example 1	Example 2	Example 3
u	7	ubrání	komponuje	grilu
i:	6	íránské	jedinými	generační
ɛ:	4	éry	jednoduchého	gumové
a:	2	árijské	brán	hořká
o:	8	ódu	famózního	jó
u:	7	úbočí	petúnie	grilů
oʊ	8,4	ouha	fanouška	hanbou
əʊ	2,4	auto	hydrauliky	
ɛʊ	4,4	euforie	terapeutická	
ə	1			

Consonant for cs-CZ

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
p	21	padá	hyperaktivní	handicap
b	21	balon	hraběte	
t	19	tabák	étos	agent
d	19	disco	čidel	
c	16	tuhýk	kandidáti	kaprad'
ʒ	16	d'ábel	hrdiný	
k	20	kaligrafie	důkazní	grafik
g	20	gangstera	hygienici	
ts	19,15	civilistům	evoluce	klávesnic
ðz	19,15		leckde	
tʃ	19,16	čepice	hlídači	klíč

ipa	viseme	Example 1	Example 2	Example 3
ðʒ	19,16	džusy	kilojoulů	
f	18	fádní	grafice	graf
v	18	vabank	exkluzivita	
s	15	sekundě	grimasami	impuls
z	15	záhadným	gruzínské	
r̥	13	řad	hořícím	
ʃ	16	šálků	grošů	kéž
ʒ	16	žab	loupeže	
j	6	já	číhající	línej
x	12	chlapcem	lichotí	domorodých
h	12	hektar	historického	
r̥	13	rabat	guru	faktor
l̥	14	ladu	eliminovat	netajil
m̥	21	macešsky	amatérský	cením
n̥	19	následkům	cenné	gurmán
ŋ̥	20		kolonky	
n̥	19	ňadra	komorně	kolegyň
m̥	21		komfort	
ř̥	13	třída	extratřídy	komentář

da-DK

Vowels for da-DK

[Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	abitur	kortfattet	marina
ɑ	2	abnorme	picaresk	varer
ɑ:	2	ara	pirat	vandkar
ɛ	4	edderkop	andægtig	tomatpuré
ɛ:	4	eventyr	dyrlæge	
ɔ	3	onde	dysfunktion	Suså
ɒ	2	ånd	abehånd	akkumulator
ɒ:	2	årbog	fyrreårig	rickshaw
ɔ:	3	åbne	modstående	husarblå
ə	4		buntmagere	bortgifter
æ:	1	abegilde	flagdage	hovedfag
e	4	editere	klarheden	kanapé
ø	1	ødem	mælkebøtte	miljø
ø:	1	øde	bortførere	lindø
ə	1	Eyolf	meneders	æde
e:	4	edelweiss	placebo	Culmsee
i	6	iaften	lipizzaner	bankkonti
i:	6	ilinger	industrien	bagi
o	8	oase	udelod	bravo
œ	4	øm	varmerør	bøh
æ:	4	Earl	arbejdssprøves	Edinburgh
o:	8	oberst	bevatrons	congo
u	7	udøver	spektakulær	endnu
u:	7	uge	spidsbue	eisuu
y	4	ybsalon	underkrydder	menu

ipa	viseme	Example 1	Example 2	Example 3
y:	4	yde	vidsyn	dødssyg

Consonant for da-DK

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	blandt	gabende	dåb
d	19	då	hævder	grædt
ð	17	thousand	baaden	grad
f	18	fabrik	spøgefugle	boligstof
g	20	gaader	fåborgenser	fik
h	12	håb	trægheden	
j	6	hjorte	miljø	bagtøj
k ^h	20	kabale	bortkomme	Cuc
l	14	lå	romanblad	fatal
m	21	maya	taxametret	calcium
n	19	nå	togene	bredden
ŋ	20		funktion	klarering
p ^h	21	pa	culpa	
? ^h	19		affyre	bortgå
r	13	rå	areal	
ɛ	4		jonglørkunst	bjørnebær
s	15	så	person	belønnes
c	16	Sjælland	benzinstation	affiche
t	19	toge	sidetal	administrationsapparat
v	18	vaagner	evaluere	dåkalv

ipa	viseme	Example 1	Example 2	Example 3
w	7	walkman	prøve	meterlov

de-DE/de-CH/de-AT

Suprasegmentals for German

[Expand table](#)

Example 1 (Onset for consonant, word-initial for vowel)	Example 2 (Intervocalic for consonant, word-medial nucleus for vowel)	Example 3 (Coda for consonant, word-final for vowel)	Comments
anders /a 1 n - d ax r s/	Multiplikationszeichen /m uh l - t iy - p l iy - k a - ts y ow 1 n s - ts ay - c n/	Biologie /b iy - ow - l ow - g iy 1/	Speech service phone set puts stress after the vowel of the stressed syllable
Allgemeinwissen /a 2 l - g ax - m ay 1 n - v ih - s n/	Abfallentsorgungsfirma /a 1 p - f a l - ^ eh n t - z oh 2 ax r - g uh ng s - f ih ax r - m a/	Computertomographie /k oh m - p y uw 1 - t ax r - t ow - m ow - g r a - f iy 2/	The Speech service phone set puts stress after the vowel of the sub-stressed syllable

Vowels for German

[Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
a:	a:	2	Aber	Maßstab	Schema
a	a	2	Abfall	Bach	Agatha
oh	ɔ	3	Osten	Pfosten	
eh:	ɛ:	4	Ähnlichkeit	Bär	Fasciae ¹
eh	ɛ	4	ändern	Prozent	Amygdalae
ax	ə	1	'verstauen ²	Aachen	Frage

sapi	ipa	VisemelID	Example 1	Example 2	Example 3
iy	i:	6	Iran	abbiegt	Relativitätstheorie
ih	ɪ	6	Innung	singen	Woody
eu	ø:	1	Ösen	ablösten	Malmö
ow	o, o:	8	ohne	Balkon	Treptow
oe	œ	4	Öffnung	befördern	
ey	e, e:	4	Eberhard	abfegt	b
uw	u:	7	Udo	Hut	Akku
uh	ʊ	4	Unterschiedes	bunt	
ue	y:	4	Übermut	pflügt	Menü
uy	y	7	üppig	System	

1 Only in words of foreign origin, such as *Fasciae*.

2 Word-initial only in words of foreign origin, such as *Appointment*. Syllable-initial in 'verstauen'.

Diphthong for German

[Expand table](#)

sapi	ipa	VisemelID	Example 1	Example 2	Example 3
ay	ai	2,6	einsam	Unabhängigkeit	Abtei
aw	au	2,7	außen	abbaust	Stau
oy	ɔy, ɔʏ	3,4	Euphorie	träumt	scheu

Semivowels for German

[Expand table](#)

sapi	ipa	VisemelID	Example 1	Example 2	Example 3
ax r	ə	4		abändern	locker

Consonants for German

[\[+\] Expand table](#)

sapi	ipa	VisemelID	Example 1	Example 2	Example 3
b	b	21	Bank		Pub ¹
d	d	19	danken	Lendl ²	Claude ³
jh	χ	16	Jeff	gemanagt	Change ⁴
f	f	18	Fahrtduer	angriffslustig	abbruchreif
g	g	20	gut	Greg ⁵	
h	h	12	Hausanbau		
y	j	6	Jod	Reaktion	hui
k	k	20	Koma	Aspekt	Fleck
l	l	14	lau	ähneln	zuviel
m	m	21	Mut	Amt	Lehm
n	n	19	nun	und	Huhn
ng	ŋ	20	Nguyen ⁶	Schwank	Ring
p	p	21	Partner	abrupt	Tip
pf	pf	21,18	Pferd	dampft	Topf
r	r, r̥, ɾ	13	Reise	knurrt	Haar
s	s	15	Staccato ⁷	bist	mies
sh	ʃ	16	Schule	mischt	lappisch
t	t	19	Traum	Straße	Mut
ts	ts	19,15	Zug	Arzt	Witz
ch	tʃ	19,16	Tschechien	aufgeputscht	bundesdeutsch
v	v	18	winken	Qualle	Groove ⁸
x	x ⁹ , ç ¹⁰	12	Bacherach ¹¹	Macht möglichst	Schmach 'ich
z	z	15	super		
zh	ʒ	16	Genre	Breezinski	Edvige

- 1 Only in words of foreign origin, such as *Pub*.
- 2 Only in words of foreign origin, such as *Lendl*.
- 3 Only in words of foreign origin, such as *Claude*.
- 4 Only in words of foreign origin, such as *Change*.
- 5 Word-terminally only in words of foreign origin, such as *Greg*.
- 6 Only in words of foreign origin, such as *Nguyen*.
- 7 Only in words of foreign origin, such as *Staccato*.
- 8 Only in words of foreign origin, such as *Groove*.
- 9 The IPA **x** is a hard "ch" after all non-front vowels (*a, aa, oh, ow, uh, uw*, and the diphthong *aw*).
- 10 The IPA **ç** is a soft "ch" after front vowels (*ih, iy, eh, ae, uy, ue, oe, eu*, and diphthongs *ay, oy*) and consonants.
- 11 Word-initial only in words of foreign origin, such as *Juan*. Syllable-initial also in words such as *Bacherach*.

Oral consonants for German

[] [Expand table](#)

sapi	ipa	VisemeID	Example
^	?	19	beachtlich /b ax - ^ a 1 x t - l ih c/

ⓘ Note

We need to add a [gs] phone between two distinct vowels, except when the two vowels are a genuine diphthong. This oral consonant is a glottal stop. For more information, see [glottal stop](#). Besides, de-CH, de-AT locales don't support SAPI phones now.

el-GR

Vowels for el-GR

[] [Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	άκουσμα	μαγαζάκι	ουδέτερα
e	4	αίτιο	κατέχω	ουδέποτε

ipa	viseme	Example 1	Example 2	Example 3
i	6	εισπνοή	ξυρισμένο	εαρινή
o	8	όαση	δώρο	δυ**ο
u	7	ουγγρικό	παπούτσι	μαίμου

Consonant for el-GR

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	μπορώ	φάμπρικα	παμπ
c	16	και	σακί	
ç	12	χέρι	μοναχή	
d	19	ντύνει	πέντε	οφ-σάιντ
ð	17	δρόμος	κραδασμοί	
dz	19,15	τζάμι	φλάντζα	
f	18	φεύγω	καφετέρια	ουφ
g	20	γκρέμισε	όγκος	πινγκ-πονγκ
ɣ	20	γαστρονομική	μεγαλόπνοα	
ʒ	16	γκισέ	φαράγγι	
j	6			
ɟ	12	γέρος	κραγιόν	
k	20	κάρτα	ιππικό	κρακ
l	14	λόγος	μιλώ	προφίλ
m	21	μιλώ	δραματική	πριμ
n	19	νόμιμο	Δανέζα	πριν
p	21	πίνω	απέτυχαν	σελοτέιπ
r	19	ρούχα	εαρινή	σέντερ

ipa	viseme	Example 1	Example 2	Example 3
s	15	σελίδα	μέσο	πλάτος
t	19	τότε	τότε	κιτ
θ	19	θέλω	φορολογηθείς	μαμούθ
ts	19,15	τσάγια	χαράτσι	ματς
v	18	βράδυ	διαβάζω	μοβ
x	12	χρόνος	μονάχα	αχ
z	15	ζέστη	χιονίζει	πλαζ

en-GB/en-IE/en-AU

Vowels for en-GB

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a:	2		fast	bra
æ	1		fat	
ʌ	1		bug	
ɛə	4,1			hair
aʊ	2,4	out	mouth	how
ə	1	a		driver
aɪ	2,6		five	
ɛ	4	egg	dress	
ɜː	5	ernest	shirt	fur
eɪ	4,6	ailment	lake	pay
ɪ	6		adding	
iə	6,1		beard	hear

ipa	viseme	Example 1	Example 2	Example 3
i:	6	eat	seed	see
ɒ	2		pod	
ɔ:	3		dawn	
əʊ	1,4		code	pillow
ɔɪ	3,6		point	boy
ʊ	4		look	
ʊə	4,1			tour
u:	7		food	two

Consonant for en-GB

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	bike	ribbon	rib
tʃ	19,16	challenge	nature	rich
d	19	date	caddy	slid
ð	17	this	father	breathe
f	18	face	laughing	enough
g	20	gold	bragging	beg
h	12	hurry	ahead	
j	6	yes		
dʒ	19,16	gin	badger	bridge
k	20	cat	lucky	truck
l	14	left	gallon	fill
m	21	mile	limit	ham
n	19	nose	phonetic	tin

ipa	viseme	Example 1	Example 2	Example 3
ŋ	20		singer	long
p	21	price	super	tip
ɹ	13	rate	very	
s	15	say	sissy	pass
ʃ	16	shop	cashier	leash
t	19	top	kitten	bet
θ	19	theatre	mathematics	breath
v	18	very	liver	have
w	7	will		
z	15	zero	blizzard	rose
ʒ	16		vision	beige

en-US/en-CA

Suprasegmentals for English

[Expand table](#)

Example 1 (onset for consonant, word-initial for vowel)	Example 2 (intervocalic for consonant, word-medial nucleus for vowel)	Example 3 (coda for consonant, word-final for vowel)	Comments
burger /b er 1 r - g ax r/	falafel /f ax - l aa 1 - f ax l/	guitar /g ih - t aa 1 r/	The Speech service phone set puts stress after the vowel of the stressed syllable.
inopportune /ih 2 - n aa - p ax r - t uw 1 n/	dissimilarity /d ih - s ih 2- m ax - l eh 1 - r ax - t iy/	workforce /w er 1 r k - f ao 2 r s/	The Speech service phone set puts stress after the vowel of the sub-stressed syllable.

Vowels for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
iy	i	6	eat	feel	valley
ih	ɪ	6	if	fill	
ey	eɪ	4,6	ate	gate	day
eh	ɛ	4	every	pet	meh (rare word-final)
ae	æ	1	active	cat	nah (rare word-final)
aa	a	2	obstinate	poppy	rah (rare word-final)
ao	ɔ	3	orange	cause	Utah
uh	ʊ	4	book		
ow	oʊ	8,4	old	clone	go
uw	u	7	Uber	boost	too
ah	ʌ	1	uncle	cut	
ay	aɪ	11	ice	bite	fly
aw	aʊ	9	out	south	cow
oy	ɔɪ	10	oil	join	toy
y uw	ju	6,7	Yuma	human	few
ax	ə	1	ago	woman	area

R-colored vowels for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
ih r	ɪɹ	6,13	ears	tiramisu	near
eh r	ɛɹ	4,13	airplane	apparently	scare
uh r	ʊɹ	4,13			cure
ay r	aɪɹ	11,13	Ireland	fireplace	choir

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
aw r	ɑʊ	9,13	hours	powerful	sour
ao r	ɔʊ	3,13	orange	moral	soar
aa r	ɑʊ	2,13	artist	start	car
er r	ɛ	5	earth	bird	fur
ax r	ə	1		allergy	supper

Semivowels for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
w	w	7	with, suede	always	
y	j	6	yard, few	onion	

Aspirated oral stops for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
p	p	21	put	happen	flap
b	b	21	big	number	crab
t	t	19	talk	capital	sought
d	d	19	dig	random	rod
k	k	20	cut	slacker	Iraq
g	g	20	go	ago	drag

Nasal stops for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
m	m	21	mat, smash	camera	room
n	n	19	no, snow	tent	chicken
ng	ŋ	20		link	sing

Fricatives for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
f	f	18	fork	left	half
v	v	18	value	event	love
th	θ	19	thin	empathy	month
dh	ð	17	then	mother	smooth
s	s	15	sit	risk	facts
z	z	15	zap	busy	kids
sh	ʃ	16	she	abbreviation	rush
zh	ʒ	16	Jacques	pleasure	garage
h	h	12	help	enhance	a-ha!

Affricates for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
ch	tʃ	19,16	chin	future	attach
jh	dʒ	19,16	joy	original	orange

Approximants for English

[\[+\] Expand table](#)

sapi	ipa	VisemeID	Example 1	Example 2	Example 3
l	l	14	lid, glad	palace	chill
r	r	13	red, bring	borrow	tar

ⓘ Note

en-CA locale doesn't support SAPI phones.

es-ES

Vowels for es-ES

[\[+\] Expand table](#)

sapi	ipa	viseme	Example 1	Example 2	Example 3
a	a	2	alto	cantar	casa
i	i	6	ibérica	avispa	taxis
e	e	4	elefante	atento	elefante
o	o	8	ocaso	encontrar	ocaso
u	u	7	usted	punta	Juanlu

Consonant for es-ES

[\[+\] Expand table](#)

sapi	ipa	viseme	Example 1	Example 2	Example 3
b	b	21	baobab	cambio	amb
	β	21		baobab	baobab
ch	tʃ	19,16	cheque	coche	Marraquech
d	d	19	dedo	candado	portland
	ð	17		dedo	verdad

sapi	ipa	viseme	Example 1	Example 2	Example 3
f	f	18	fácil	elefante	puf
g	g	20	ganga	ganga	dóping
	ɣ	20		agua	tuareg
j	j	6	iodo	caliente	rey
jj	ɟɟ	6,6		villa	
k	k	20	coche	boca	titánic
l	l	14	lápiz	ala	cordel
ll	ʎ	14	llave	conllevar	
m	m	21	morder	amar	álbum
n	n	19	nada	cena	ratón
nj	ɲ	19	ñaña	arañazo	
p	p	21	poca	topo	stop
r	r	19		cara	abrir
rr	r	13	radio	corre	purr
s	s	15	saco	vaso	pelos
t	t	19	toldo	atar	disquet
th	θ	19	zebra	azul	lápiz
w	w	7	hueso	agua	guau
x	x	12	jota	ajo	reloj

💡 Tip

The es-ES Speech service phone set doesn't support the following Spanish IPA: β, δ, and γ. If they're needed, consider using the IPA directly.

es-MX

Vowels for es-MX

[\[+\] Expand table](#)

ipa	VisemelID	Example 1	Example 2	Example 3
a	2	azúcar	tomate	ropa
e	4	eso	remero	amé
i	6	hilo	líquido	olí
o	8	hogar	olote	caso
u	7	uno	ninguno	tabú

Consonants for es-MX

[\[+\] Expand table](#)

ipa	VisemelID	Example 1	Example 2	Example 3
b	21	bote		
β	21	órbita	envolvente	
tʃ	19,16	chico	hacha	
d	19	dátil		
ð	17	orden	oda	
f	18	foco	oficina	
g	20	gajo		
ɣ	20	agua	hoguera	
j	6	iodo	caliente	rey
ʝ	6,6		olla	
k	20	casa	ácaro	
l	14	loco	ala	
ʎ	14	llave	enyugo	
m	21	mata	amar	

ipa	VisemelD	Example 1	Example 2	Example 3
n	19	nada	ano	
ñ	19	ññoño	año	
p	21	papa	papa	
r	19		aro	
r	13	rojo	perro	
s	15	silla	asa	
t	19	tomate		soft
w	7	huevo		
x	12	jarra	hoja	

fi-Fl

Vowels for fi-Fl

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2	avautuu	vaihtuvan	pouta
äi	2,6	aika	vaihtuu	lauantai
äu	2,7	aura	uloskirjaudu	Passau
a:	2	ay-väen	neutraali	poutaa
æ	1	äveriäs	öljyjätin	pöytä
äi	1,6	äiti	iäkkäiden	täi
äy	1,4	äyrin	täytty	käy
æ:	1	ääriryhmiä	häädetään	päivää
e	4	enköhän	terve	me
ei	4,6	ei	vaihteita	hei

ipa	viseme	Example 1	Example 2	Example 3
∅	1	öljyalan	ulkonäön	tiedänkö
∅i	1,6	öisin	töitä	viittilöi
∅y	1,4	öylätti	pöytä	
∅:	1	Öölanti	ulkoministeriöön	Bodø
eu	4,7	eurot	kyläseura	leu
ey	4,4	Eysturoy	keskeytyä	
e:	4	eesti	kyljelleen	aiheuttanee
i	6	iäkästä	viha	Berliini
ie	6,4	ientaskun	kieli	lie
iü	6,7		viulu	
iÿ	6,4		vihkiytynyt	
i:	6	lida	siika	solmii
o	8	oksa	asuintaloja	spekulaatio
oi	8,6	oivia	koittaa	spekuloi
ou	8,7	outo	autokoulu	window
o:	8	ok	koostaa	yo
u	7	ufoista	Bärlund	jätémaksu
ui	7,6	ui	muita	epäonnistui
uo	7,8	Uolevi	Suomi	Hilavuo
u:	7	url	innokkuus	kiikkuu
y	4	ydin	ökyrikas	kesy
y∅	4,1	yö	työtä	järjestöö
yî	4,6	Yichangin	syitä	järjestäyti
y:	4	yo	ryyppyy	iskeytyy

Consonant for fi-FI

ipa	viseme	Example 1	Example 2	Example 3
b	21	baareissa	Urban	Jakob
d	19	dementia	ladot	jugend
f	18	face	afgaani	Alf
g	20	gaalassa	fagotti	Aslög
h	12	ha	astumahan	Dietrich
j	6	jää	öljyä	Kaj
k	20	Kajaanin	epätarkat	idyllic
l	14	Lyytikäinen	euroseteiden	mail
m	21	mä	roimat	späm
n	19	nää	baanalle	iäkkään
ŋ	20		nähädänkin	planning
p	21	pa	epäsuoria	backup
r	13	risteilyn	baari	Player
s	15	sä	öljyisiä	Bärnäs
ʃ	16	Schauman	Bangladeshin	cash
t	19	tä	eurosta	epäsuorat
u	18	vaadi	innostava	Kiev

fr-FR/fr-CA/fr-CH

Suprasegmentals for French

The Speech service phone set puts stress after the vowel of the stressed syllable. However, the fr-FR Speech service phone set doesn't support the IPA substress '̚'. If the IPA substress is needed, you should use the IPA directly.

Vowels for French

[\[+\] Expand table](#)

sapi	ipa	viseme	Example 1	Example 2	Example 3
ae	a	2	arbre	patte	ira
af	a	2		pâtre	pas
an	ã	2	enfant	enfant	temps
ax	ə	1		petite	le
eh	ɛ	4	elle	perdu	était
eu	ø	1	œufs	creuser	queue
ey	e	4	ému	crétin	ôté
in	ɛ̃	4	important	peinture	matin
iy	i	6	idée	petite	ami
oe	œ	4	œuf	peur	
oh	ɔ	3	obstacle	corps	
on	ɔ̃	3	onze	rondeur	bon
ow	o	8	auditeur	beaucoup	pô
un	œ̃	4	un	lundi	brun
uw	u	7	outrage	introuvable	ou
uy	y	4	une	punir	élu

Consonant for French

[\[+\] Expand table](#)

sapi	ipa	viseme	Example 1	Example 2	Example 3
b	b	21	bête	habille	robe
d	d	19	dire	rondeur	chaude
f	f	18	femme	suffixe	bof
g	g	20	gauche	égale	baguette

sapi	ipa	viseme	Example 1	Example 2	Example 3
gn	j̪	19			peigne
hw	ɥ	7	huile	nuire	
k	k	20	carte	écaille	bec
l	l	14	long	élire	bal
m	m̪	21	madame	aimer	pomme
n	n̪	19	nous	tenir	bonne
ng	ŋ	20			parking
p	p̪	21	patte	repas	cap
r	r̪	13	rat	chariot	sentir
s	s̪	15	sourir	assez	passe
sh	ʃ	16	chanter	machine	poche
t	t̪	19	tête	ôter	net
v	v̪	18	vent	inventer	rêve
w	w̪	7	oui	fouine	
y	j	6	yod	piétiner	Marseille
z	z̪	15	zéro	raisonner	rose
	n̄	19			un arbre
	t̄	19			quand
	z̄	15			corps

1 Only for some foreign words.

💡 Tip

The `fr-FR` Speech service phone set doesn't support the following French liaisons, `n̄`, `t̄`, and `z̄`. If they are needed, you should consider using the IPA directly.

❗ Note

fr-CA, fr-CH locales don't support SAPI phones now.

he-IL

Vowels for he-IL

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
i	6	אֵישׁ	סִיר	כָּלִי
e	4	אֵשׁ	כֵּל	פָּה
a	2	אָף	קָרְבָּן	מָה
o	8	אוֹתָן	יָמִין	לֹא
u	7	עוֹגָה	כַּרְבוֹב	הָגִיעָן

Consonant for he-IL

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
p	21	פָּס	קָלִיפה	טִיפָּה
b	21	בָּז	סְבָא	פָּאָב
t	19	טִיפה	מְתֻנָּה	חוֹט
d	19	דְּבָר	אֲדוֹם	תְּמִיד
k	20	כְּתָב	בָּוקָר	חָלָק
g	20	גָּדוֹל	אֲגָפָה	דָּג
χ	19	אֲכִיב	שִׁיעָור	
f	18	פִּילָּטָר	סְוִיפָּר	סְוִיפָּה
v	18	וִילָּוֹן	כְּבָד	לְבָד
s	15	שְׁמַלָּה	כְּסָף	גְּנָס

ipa	viseme	Example 1	Example 2	Example 3
z	15	זאב	מזל	אגוז
ʃ	16	שולחן	פישוט	ככיש
x	12	חתול	אוכל	פרח
h	12	הולר	זהב	בת
tʃ	19,15	צד	עצם	טומליץ
m	21	סואוד	סימן	חלום
n	19	נפש	תינוק	אכן
l	14	לשון	טיליה	דגל
ç	13	ראשון	מורה	חיבור
j	6	ילד	מצין	כדי
ʒ	16	צ'אנר	מץ'ור	בצ'
tʃ	19,16	צ'יף	קפק'אל	'סנדוויץ'
dʒ	19,16	ג'ונגל	פיג'מה	'קוטג'

hr-HR

Vowels for hr-HR

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
e	4	Egipat	Games	drveće
e:	4	eri	brijegom	de1taljne
i	6	ispada	želimo	javnosti
i:	6	iako	list	kompletни
u	7	ubacio	konkurentne	jednu
u:	7	Una	funta	Yu

ipa	viseme	Example 1	Example 2	Example 3
a	2	američke	kovačić	kredita
a:	2	anđela	gradila	Divulja
o	8	oaza	nanosi	do
o:	8	Olgu	kiselog	to

Consonant for hr-HR

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
d	19	dakle	evidenciji	kod
v	18	volja	građevinskog	imperativ
s	15	sabor	informisanja	interes
t	19	tad	informirati	ispit
n	19	na	žene	jeftin
l	14	logor	konstatirali	kapital
ʎ	14	ljudski	košulju	kralj
tʂ	19,15	carina	krivice	pravac
tʃ	19,16	četvorica	kritičan	osnivač
j	6	jednostavan	kuju	ovaj
x	12	hrvatskog	mahala	ovakvih
z	15	znanstvenom	mehanizacije	prijelaz
ʒ	16	žalbu	mladeži	crtež
r	13	red	moraju	dar
k	20	kažu	nakani	dnevnik
m	21	Mađara	napadima	dobrim
p	21	Poljska	napadnut	kamp

ipa	viseme	Example 1	Example 2	Example 3
g	20	gore	negativna	kaznenog
č	16	ćelija	neisplaćene	mladić
f	18	fabula	nostrifikaciji	šef
b	21	Belgija	oba	sukob
dʒ	19,16	džempera	lhidže	George
n	19	nje	emitiranja	stupanj
đ	16	đakovačkim	gađati	vođ
ʃ	16	šef	stišati	Glavaš

hu-HU

Vowels for hu-HU

[\[+\]](#) [Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
ø	1	ördög	ördög	huszonkettő
ø:	1	ők	önöműködően	öntöző
a	2		Schumacher	
a:	2	árut	bizonyára	burzsuá
ɛ	4	előzni	kisbéresnek	hogyne
e:	4	Édes	komédia	fölfelé
i	6	idegen	omnibuszok	kollégiumi
i:	6	ígyen	áhítat	
o	8	oldali	komor	Figo
ø	2	atyját	olvasni	Olga
o:	8	ólmot	históriát	fénymásoló

ipa	viseme	Example 1	Example 2	Example 3
u	7	ugyanis	ezután	falu
u:	7	úrrá	fékút	számú
y	4	üdítőt	átsütve	alsóbbrendű
y:	4	űrállomás	gépjárművek	idejű

Consonant for hu-HU

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	busz	hiába	
b:	21		hasábbburgonya	rövidebb
d	19	dévédé	fapados	szaporítanád
ʒ	16	gyártó	franciaágynas	huszonegy
d:	19		Goddess	hadd
ʒ:	16		ebédjén	hagyj
ðʒ	19,16	Dzsó	menedzselési	college
ðʒ:	19,16		Baggio	
dz	19,15	dzémsz	edző	McDonalds
dz:	19,15		eddzeni	
f	18	figura	kofa	golf
f:	18		koffer	seriff
g	20	gondolom	számítógép	rúg
g:	20		faggatta	függ
h	12	hit	ruhában	ah
h:	12		ehhez	
j	6	János	olyan	karéj

ipa	viseme	Example 1	Example 2	Example 3
j	19	nyakán	pecsenyét	példány
j:	6	Ljeszkovai	majmolják	állj
j:	19		pihenjen	
k	20	kofa	ruhákat	ruhák
k:	20		megkondult	makk
l	14	lent	rúla	evvel
l:	14		kolléga	áll
m	21	magyar	számít	órám
m:	21		anyámmal	kilogramm
n	19	népies	szótlanul	hiszen
ŋ	20		angel	
n:	19		onnan	fenn
p	21	Pál	tapogatódzó	számítógép
p:	21		beröppent	befejezéseképp
r	13	rág	óraára	órakor
r:	13		amerre	forr
s	15	számára	fölveszi	fölmész
ʃ	16	saját	förtelmesen	főorvos
s:	15		halasszuk	hazajössz
ʃ:	16		házassága	keress
t	19	Tata	rútak	hit
c	16		egyházmegye	dirty
t:	19		hitte	vágodott
c:	16		bátyja	Pretty
ts	19,15	címe	biciklis	huszonnyolc
tʃ	19,16	csigán	húgocskám	Gregorics

ipa	viseme	Example 1	Example 2	Example 3
t̪s:	19,15		játszad	játsz
t̪ʃ:	19,16		barátságos	futballmeccs
v	18	varr	Olívia	Dönöv
v:	18		evvel	
x	12	hrabovszki	ihletével	
w	20		alapján	Kapj
z	15	zúgást	csizmám	csimpánz
ʒ	16	zsûrivel	félmázsás	Balázs
z̪	15		húzza	fékezz
ʒ̪:	16		garázzsal	

id-ID

Vowels for id-ID

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
ə	1		benar	komite
a	2		babat	engga
ai	2,6	air	raikage	pantai
au	2,4	aurat	ataupun	pisau
e	4	energi	feri	tempe
ɛ	4	enrique	nenek	
I	6	indah	yakin	key
i	6	irama	biar	deflasi
ɔ	3	off	esok	law

ipa	viseme	Example 1	Example 2	Example 3
o	8	obat	bobot	domino
ɔi	3,6		reboisasi	sepoi-sepoi
u	7	umur	buah	linu
ʊ	4		duduk	

Consonant for id-ID

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
?	19	amores	maaf	pencak
b	21	babat	nebak	dishub
d	19	deder	pedang	cloud
ðʒ	19,16	jarum	penjajah	buruj
f	18	flat	aversi	genitif
g	20	gabus	dagel	hamburg
h	12	hama	tuhan	entah
ɲ	19	nyaman	menyuci	
j	6	yakin	jaya	baduy
k	20	ketan	aku	polsek
l	14	labu	talenta	vital
m	21	masuk	namanya	sinom
n	19	nada	sekunar	proton
ŋ	20	ngengat	kenanga	abang
p	21	pacar	hampa	cakap
r	13	rabu	dikurang	nasar
s	15	sabuk	tetesan	jenius

ipa	viseme	Example 1	Example 2	Example 3
ʃ	16	syarat	isyarat	british
t	19	tabir	adaptasi	durat
tʃ	19,16	cakap	dicari	
w	7	wajah	yuwana	
x	12	khusuk	akhirnya	barzakh
z	15	zakat	pezina	mahfuz

it-IT

Vowels for it-IT

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	amo	sano	scorta
ai	2,6	aics	abbaino	mai
au	2,7	audio	rauco	bau
e	4	eroico	venti	sapore
ɛ	4	elle	avvento	lacchè
ɛj	4,6	eira	email	lei
ɛu	4,7	euro	neuro	
ei	4,6		aseità	scultorei
eu	4,7	europeo	feudale	
i	6	italiano	vino	soli
u	7	unico	luna	zebù
o	8	obesità	straordinari	amico
ɔ	3	otto	botte	però

ipa	viseme	Example 1	Example 2	Example 3
ɔj	3,6		oppiodi	
oi	8,6	oibò	intellettualoide	Gameboy
ou	8,7		show	talkshow

Consonant for it-IT

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	bene	ebanista	Euroclub
b:	21		gobba	
tʃ	16	cenare	acido	french
tʃ:	19,16		braccio	
k	20		pacco	Innsbruck
d	19	dente	adorare	interland
d:	19		cadde	
ʣ	15	zero	orzo	
ʣ:	15		mezzo	
f	18	fame	afa	alef
f:	18		beffa	bluff
ɸ	16	gente	agire	beige
ɸ:	16		oggi	
g	20	gara	alghe	smog
g:	20		fugga	Zuegg
χ	14	gli	ammiragli	
χ:	14		foglia	
j:	19		bagno	

ipa	viseme	Example 1	Example 2	Example 3
j̪	19	gnocco	padrigno	Montaigne
j	6	ieri	piede	freewifi
k	20	caro	anche	tic
l	14	lana	alato	col
l:	14		colla	full
m	21	mano	amare	Adam
m:	21		grammo	
n	19	naso	lana	non
n:	19		panna	
p	21	pane	epico	stop
p:	21		coppa	
r	19	rana	motore	per
r:	13		carro	Starr
s	15	sano	cascata	lapis
s:	15		cassa	cordless
ʃ	16	scemo	Gramsci	slash
ʃ:	16		ascia	fiches
t	19	tana	eterno	alt
t:	19		zitto	
t̪	15	tsunami	turbolenza	subtests
t̪:	15		bozza	
v	18	vento	avarо	Asimov
v:	18		bevvi	
w	7	uovo	duomo	Marlowe
z	15	smodato	casa	elections

ja-JP

The Speech service phone set for ja-JP is based on the native phone Kana ↗ set.

Please see the following tables for kana and corresponding viseme in parentheses.

Katakana for ja-JP

 Expand table

Katakana	ア	イ	ウ	エ	オ
ア	ア (19,2)	イ (6,6)	ウ (7,6)	エ (19,4)	オ (19,8)
カ	カ (20,2)	キ (20,6)	ク (20,6)	ケ (20,4)	コ (20,8)
サ	サ (15,2)	シ (16,6)	ス (15,6)	セ (15,4)	ソ (15,8)
タ	タ (19,2)	チ (16,6)	ツ (19,15,6)	テ (19,4)	ト (19,8)
ナ	ナ (19,2)	ニ (19,6)	ヌ (19,6)	ネ (19,4)	ノ (19,8)
ハ	ハ (12,2)	ヒ (12,6)	フ (12,6)	ヘ (12,4)	ホ (12,8)
マ	マ (21,2)	ミ (21,6)	ム (21,6)	メ (21,4)	モ (21,8)
ヤ	ヤ (6,2)	n/a	ユ (6,6)	n/a	ヨ (6,8)
ラ	ラ (19,2)	リ (19,6)	ル (19,6)	レ (19,4)	ロ (19,8)
ワ	ワ (7,2)	n/a	n/a	n/a	ヲ (19,8)
	ン (19)	n/a	n/a	n/a	n/a

Katakana diacritics for ja-JP

 Expand table

Katakana diacritics	ア	イ	ウ	エ	オ
ガ	ガ (20,2)	ギ (20,6)	グ (20,6)	ゲ (20,4)	ゴ (20,8)
ザ	ザ (15,2)	ジ (16,6)	ズ (15,6)	ゼ (15,4)	ゾ (15,8)
ダ	ダ (19,2)	ヂ (16,6)	ヅ (15,6)	ヂ (19,4)	ド (19,8)
バ	バ (21,2)	ビ (21,6)	ブ (21,6)	ベ (21,4)	ボ (21,8)

Katakana diacritics	ア	イ	ウ	エ	オ
パ	パ (21,2)	ピ (21,6)	プ (21,6)	ペ (21,4)	ボ (21,8)

Katakana Yōon for ja-JP

[] Expand table

Katakana Yōon	ヤ	ユ	ヨ
キ	キヤ(20,6,2)	キユ(20,6,6)	キヨ(20,6,8)
シ	シヤ(16,6,2)	シユ(16,6,6)	シヨ(16,6,8)
チ	チヤ(16,6,2)	チユ(16,6,6)	チヨ(16,6,8)
ニ	ニヤ(19,6,2)	ニユ(19,6,6)	ニヨ(19,6,8)
ヒ	ヒヤ(12,6,2)	ヒユ(12,6,6)	ヒヨ(12,6,8)
ミ	ミヤ(21,6,2)	ミユ(21,6,6)	ミヨ(21,6,8)
リ	リヤ(19,6,2)	リユ(19,6,6)	リヨ(19,6,8)
ギ	ギヤ(20,6,2)	ギユ(20,6,6)	ギヨ(20,6,8)
ジ	ジヤ(16,6,2)	ジユ(16,6,6)	ジヨ(16,6,8)
ヂ	ヂヤ(16,6,2)	ヂユ(16,6,6)	ヂヨ(16,6,8)
ビ	ビヤ(21,6,2)	ビユ(21,6,6)	ビヨ(21,6,8)
ピ	ピヤ(21,6,2)	ピユ(21,6,6)	ピヨ(21,6,8)

Examples for ja-JP

[] Expand table

Character	sapi	ipa
合成	ゴ'ウセ	go'wuseji
所有者	ショユ'ウ?ヤ	ɕoju'wuya
最適化	サイテキカ+	sajitecika,

ko-KR

Vowels for ko-KR

[] Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2	아가씨	강	하다
ɛ	4	애국가	백	세번째
e	4	에너지	가겠구나	가게
w	6	으레	으쓱해	가스
i	6	이거	아직까지	어미
ʌ	1	어미	차선변경	가시겠어
o	8	오래	의혹을	차고
u	7	우간다	은둔자의	가시나무
wi	20,6	의자	배추흰나비가	가요계의
ɸ	1	외가댁	배치된다	하되
wa	7,2	와글와글	가치관과	가지와
wɛ	7,4	왜관	호쾌가	안돼
wɛ	7,4	웨딩드레스		
wi	7,6	위계적	구조위원회가	한가위
wʌ	7,1	워낙	가까워서	가까워
ja	6,2	야구	기술집약도가	끌려가야
jɛ	6,4	얘가	가수얘기예요	
je	6,4	예감	적대관계가	의례
jʌ	6,1	여가	감정평가사	이리하여
jo	6,8	요구가	사용중지	가거든요
ju	6,7	유가적	경제교류가	소유

Consonant for ko-KR

ipa	viseme	Example 1	Example 2	Example 3
b	21	바가		밥
p	21	빠른가	막부정부가	
b	21		사범학교가	
tʃʰ	19,16	참가비는	아침과	
d	19	동네가		바깥
t	19	따라가지	깍두기가	
d	19		산도가	
g	20	가조		각
k	20	까마귀가	젖가락으로	
g	20		단추가	
h	12		손가락질하거나	
h	12	하기가	손가락질하며	
dʒ	19,16		손잡이가	
dʒ	19,16	자유가		
tʃ	19,16	짜가면서	손가락질할	
kʰ	20	키가	아킬레스건	
l	14			국가체제를
m	21	마다가스카르	통나무가	기침
n	19	나가서는	아느냐	따라가다보면
ŋ	20		강아지	한강
pʰ	21	파티가	아파트	
r	19	라디오가	아름답게	
sʰ	15	사고가	아스팔트	
s	15	쌍둥이가	멕시코가	

ipa	viseme	Example 1	Example 2	Example 3
t ^h	19	택시가	여타의	

ms-MY

Vowels for ms-MY

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
i	6	ibu	iklim	ahli
u	7	uang	buah	bahu
ə	1		kerja	nasionalisme
e	4	edar	aktres	kue
o	8	orang	anggota	pidato
a	2	anjing	anak	ada
ai	2,6			cerai
au	2,7	auto	akaun	bakau
oi	8,6			amboi

Consonant for ms-MY

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
p	21	pekat	sekeping	cakap
b	21	banjir	lebih	jawab
t	19	tidak	peta	sikit
d	19	dekat	adakah	akad
k	20	ketat	akak	book

ipa	viseme	Example 1	Example 2	Example 3
g	20	gabung	bogel	dialog
?	19	abang	kepercayaan	letak
tʃ	19,16	cepat	baca	
dʒ	19,16	jabatan	aja	kolej
m	21	memang	laman	malam
n	19	negeri	tanam	taman
ŋ	19	nyanyi	tanya	
ŋ	20		mangga	sayang
f	18	filem	artifak	aktif
v	18	vaksin	aktiviti	
s	15	sahabat	akses	tumis
z	15	zaman	lazat	
ʃ	16	syarikat	bersyarat	
x	12	khabar	akhir	tarikh
r	13	racun	merah	lebar
h	12	hingga	aduhai	boleh
j	6	yang	ayah	
w	7	walau	bawah	
l	14	lidah	alam	katil

nb-NO

Vowels for nb-NO

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	annonse	betrakte	hoppa
æ	1	ergre	Palermo	
æ:	1	ærlig	belære	bæ
a:	2	are	betale	bedra
ɛ	4	energi	kadetten	hoppe
ø:	1	øre	behøve	adjø
e:	4	ener	berede	distre
i	6	ikke	setningen	taxi
i:	6	Eagle	bevise	konditori
ɔ	3	åtte	kontrolle	altså
œ	4	ønske	belønning	Mossø
o:	8	år	område	begå
u	7	økse	økokrim	ego
u:	7	ord	telefonen	bidro
y	7	ytterst	benytte	Ally
ʉ	6	under	forundret	jaggu
ʉ:	6	ule	umulig	intervju
y:	4	yte	belyse	paraply
æɪ	1,6	eiendom**	utleide	snarvei
æʉ	1,6	aura	Litauen	fortau
ɑɪ	2,6	aibel**	Aserbajdsjan	Dubai
œy	4,7	øyer	ablegøy	syltetøy
ɔʏ	3,7	Oilers	boikotten	konvoi
ʉɪ	6,6	Bruins		Mitsui

Consonant for nb-NO

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
p	21	pil	ape	lapp
t	19	tall	matte	matt
k	20	kall	jakke	takk
b	21	bil	klubbe	lobb
d	19	dal	lide	gadd
g	20	gås	sagen	lag
f	18	fil	klaffe	klaff
h	12	hall	beholde	
s	15	sil	vise	viss
ʂ	15	sju	maskin	dusj
ç	12	tjukk	bekjenne	Korch
v	18	vår	leve	lov
m	21	mil	komme	lam
n	19	nål	minnes	søvn
ɳ	20		penger	lang
l	14	lös	måle	tal
r	13	ris	karre	tørr
j	6	jag	utjevne	detalj
ɖ	19		burde	ferd
ɿ	14		farlig	jarl
ɳ	19		barnet	jern
t	19		skjorte	gjort

nl-NL/nl-BE

Vowels for nl-NL

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	af	bak	bavarois
a:	2	aan	maal	ja
ã	2	enfin	manchet	croissant
āu	2,7	oud	bout	hou
ɛ	4	en	weg	hè
e:	4	één	heet	nee
ɛ:	4	airbag	blèr	
ɛ̄i	4,6	eis	wijn	zij
ɛ̄	4		lingerie	elektricien
ø:	1	euro	deur	milieu
i	6	ik	ding	
i	6	iets	sliep	drie
ɔ	3	op	slot	joh
u	7	oefen	hoed	doe
ɔ:	3		roze	
᷇	3			Macron
o:	8	ook	boom	zo
y	7	urn	dus	
ə	1	een	trommel	de
œ̄y	4,4	uil	juist	bui
œ̄	4	oeuvre	service	
y	4	uur	tuur	nu

Consonant for nl-NL

[\[\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	boos	fobie	
d	19	dat	kudde	
f	18	fiets	gefeest	blaf
x	12	ga	magie	hoog
χ	19		beamen	
h	12	hoek	behaard	
g	20	gujarati	again	drug
j	6	jij	boeien	haai
k	20	kat	haken	zwak
l	14	land	familie	kool
m	21	man	demon	raam
n	19	niks	kannon	pan
ŋ	20		brengen	zing
p	21	poer	rapen	heb
r	13	romp	waarom	kier
s	15	soms	precies	heus
ʃ	16	sjaal	vaasje	lunch
t	19	tot	laten	groot
w	7		flauwe	follow
v	18	voor	haven	
u	18	wat	fusiewet	
z	15	zal	lezen	
ʒ	16	jus	beige	

pl-PL

Vowels for pl-PL

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2	autor	kolację	karasia
ɛ	4	elbląska	reflektor	osiągniecie
ɛ̄	4		nieczęsto	skorupę
i	6	informatycznym	powiśle	gadali
ī	6		cudzymi	rodziły
ɔ	3	osobniki	uboju	rogatko
ɔ̄	3		mąż	intelektualistą
u	7	unosimy	arkuszy	przeznaczeniu

Consonant for pl-PL

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	bliska	wyśrubowane	
b ^j	21	biuro	zapobiegać	
tç	19,16	ciągników	uczuciowych	suchość
tʂ	19,15	cząstkowe	odpoczynek	niszcz
c	16	kije	lisikiewicz	
d	19	drogowy	porodówkę	bastard
d ^j	19	dializy	studiuję	
dz	19,15	dzwonnica	wrodnony	
dʐ	19,16	dziurawe	pochodziłam	

ipa	viseme	Example 1	Example 2	Example 3
f	18	wprost	długofalowy	konserwantów
f ^j	18	filmoteki	grafiką	
g	20	geometrią	navigacja	
ʒ	16	gitarzysta	religijnym	
ðz	19,15	dżunglę	menedżerskie	
k	20	królestwa	naukowo	matematyk
l	14	latający	populacje	handel
l ^j	14	lisek	okolica	
m	21	majątkowy	wytłumaczenia	błagam
m ^j	21	mieszkającej	dynamicznie	
n	19	napędu	poczynaniach	balkon
ŋ	20		ciągłość	
n̪	19	niewidoczna	zmieniała	poznań
p	21	potoczne	terapeuti	odstęp
p ^j	21	pijawek	skupieniu	
r	13	regionu	operową	administrator
r ^j	13	ripostuje	imperialnej	
s	15	solone	przetasowania	biogaz
ç	16	sierpień	doniesieniem	mogłaś
ʃ	16	szanowanych	wpatrzeniu	skręcasz
t	19	talentom	kwaterze	dowód
t ^j	19	tirami	marketingiem	
t̪s	19,15	cyfrą	agencyjne	pałac
v	18	wysłaniu	przeprowadzają	
v ^j	18	winiarstwa	bukowianka	
w	7	łączenie	prałata	drukował

ipa	viseme	Example 1	Example 2	Example 3
x	12	hamulcem	zdychają	kostiumach
x ^j	12	hiszpańscy	psychice	
j	6	jeździła	popijałam	najważniejszej
z	15	zaskakują	partyzanckich	wiz
z̥	16	ziemniaki	zgryzienia	
ʒ	16	żyrandol	nowożytnej	

pt-BR

Vowels for pt-BR

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
i	6	ilha	ficar	comi
ĩ	6	intacto	pintar	aberdeen
a	2	água	dada	má
ɔ	3	ora	porta	cipó
u	7	ufanista	mula	peru
ũ	7	uns	pungente	kuhn
o	8	ortopedista	fofo	avô
e	4	elefante	elefante	você
ẽ	4	anta	canta	amanhã
ə	1	aqui	amaciar	dada
ɛ	4	ela	serra	até
ẽ	4	endorfina	pender	
õ	8	ontologia	conto	

Consonant for pt-BR

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
~	7			atualização
w	7	washington	água	usou
p	21	pato	capital	
b	21	bola	cabeça	
t	19	tato	rato	
d	19	dado	amado	
g	20	gato	maragato	
m	21	mato	comer	
n	19	no	ano	
ɲ	19	nhoque	ninho	
f	18	faca	afago	
v	18	vaca	cavar	
r	19		para	amar
s	15	satisfeito	amassado	casados
z	15	zebra	azar	
ʃ	16	cheirar	machado	
ʒ	16	jaca	injusta	
x	12	rota	carreta	
tʃ	19,16	tirar	atirar	
dʒ	19,16	dia	adiar	
l	14	lata	aleto	
ʎ	14	lhama	malhado	
ʒ	6		inabalavelmente	hífen

ipa	viseme	Example 1	Example 2	Example 3
j	6		caixa	sai
k	20	casa	ensacado	

pt-PT

Vowels for pt-PT

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	ábdito	consular	medirá
e	4	abacaxi	domação	longa
ej	4,6	eidético	direita	detectei
ẽ	4	anverso	viajante	afã
ẽj	4,6	angels	viagens	também
ẽw	4,7	hão	significaçõozinha	gabão
ẽw	4,7		saudar	hello
ãj	2,6	airosa	culturais	vai
ɔ	3	hora	depósito	ló
ɔj	3,6	óis	heróico	dói
aw	2,7	outlook	incauto	pau
ə	1	extremo	sapremar	noite
e	4	eclipse	haver	buffet
ɛ	4	eco	hibérnios	paté
ɛw	4,7		pirinéus	escarcéu
ẽ	4	embaçado	dirimente	ámen
ẽw	4,7	eu	deus	bebeu

ipa	viseme	Example 1	Example 2	Example 3
i	6	igreja	aplaudido	escrevi
ĩ	6	impaciente	espinçar	manequim
íw	6,7		niue	garantiu
o	8	ofir	consumidor	stacatto
ój	8,6	oirar	noite	foi
õ	8	ombrão	barronda	dom
õj	8,6		ocupações	expõe
u	7	ubi	facultativo	fado
új	7,6	uivar	arruivado	fui
ü	7	umbilical	funcionar	fórum
új	7,6		muito	

Consonant for pt-PT

[] Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	bacalhau	tabaco	club
d	19	dado	dado	band
r	19	rename	verás	chutar
f	18	fim	eficácia	golf
g	20	gadinho	apego	blog
j	6	iode	desassociado	substitui
k	20	kiwi	traficado	snack
l	14	laborar	pelada	full
ł	14		polvo	brasil
ꝑ	14	lhanamente	antilhas	

ipa	viseme	Example 1	Example 2	Example 3
m	21	maça	amanhã	modem
n	19	nutritivo	campana	scan
ɲ	19	nhambu-grande	toalhinha	penh
p	21	pai	crápula	laptop
r	13	recordar	guerra	chauffeur
s	15	seco	grosseira	boss
ʃ	16	chuva	duchar	médios
t	19	tabaco	pelota	input
v	18	vaca	combatível	pavlov
w	7	waffle	restituir	katofio
z	15	zâmbia	prazer	jazz
ʒ	16	gelada	infligir	cuj

ro-RO

Vowels for ro-RO

expand table

ipa	viseme	Example 1	Example 2	Example 3
ə	1	ăsta	abătut	fizică
ɨ	6	înspre	hotărâre	îhî
a	2	absolut	prematur	Praga
e	4	educație	tablete	alianțe
ea	4,2		studentească	badea
eo	4,8		bleumarin	vreo
i	6	Italia	aripi	aberații

ipa	viseme	Example 1	Example 2	Example 3
o	8	oricum	catacombe	radio
oa	8,2	oără	închisoare	șamoa
u	7	umble	gradul	Alexandru

Consonant for ro-RO

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	băi	vizibil	arab
b ^j	21			microbi
d	19	doctor	video	miliard
dʒ	19,16	ger	vegetație	
dʒ ^j	19,16			colegi
f	18	furtună	efort	ecograf
f ^j	18			filosofi
g	20	galeria	egal	filolog
g ^j	20		neghină	unghi
h	12	hexagon	arhitect	monarh
j	6	ieri	băiat	ditai
k	20	cadru	baricadat	pitic
k ^j	20			urechi
l	14	laptop	alint	Paul
l ^j	14			circuli
m	21	mandarine	camera	atom
m ^j	21			consumi
n	19	nepot	Canada	Eden

ipa	viseme	Example 1	Example 2	Example 3
ŋ	20		banca	plâng
n ^j	19			dragoni
p	21	pai	opera	Filip
p ^j	21			ocupi
r	13	real	mere	distribitor
r ^j	13			palmieri
s	15	sertar	căsătorit	exclus
ʃ	16	șine	cușetă	greș
ʃ ^j	16			groși
t	19	teracota	material	abonament
t ^j	19			foști
ts	19,15	țar	cuțit	vorbăreț
tʃ	19,16	circa	meciuri	
ts ^j	19,15			uscați
tʃ ^j	19,16			indici
v	18	vaccin	gravidă	fugitiv
v ^j	18			nervi
w	7	uau	cuantificare	pliu
z	15	zoologică	frază	parbriz
ʒ	16	jar	abajur	pasaj
ʒ ^j	15			semnezi
ʒ ^j	16			dârji

ru-RU

Vowels for ru-RU

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
а	2	адрес	радость	беда
ʌ	1	облаков	застенчивость	внучка
ə	1		яблочного	
ε	4	эпос	белка	кафе
i	6	иней	лист	соловьи
ɪ	6	игра	медведь	мгновенье
ɨ	6	энергия	лысый	весы
ɔ	3	окрик	мот	весло
ʊ	7	ужин	куст	пойду

Consonant for ru-RU

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
p	21	профессор	поплавок	укроп
p ^j	21	Петербург	ослепительно	степь
b	21	большой	собака	
b ^j	21	белый	убедить	
t	19	тайна	старенький	твид
t ^j	19	тепло	учитель	синеть
d	19	доверчиво	недалеко	
d ^j	19	дядя	единица	
k	20	крыло	кукуруза	кустарник
k ^j	20	кипяток	неяркий	
g	20	гроза	немного	

ipa	viseme	Example 1	Example 2	Example 3
g ^j	20	герань	помогите	
x	12	хороший	поход	дух
x ^j	12	хилый	хихиканье	
f	18	фантазия	шкафах	кров
f ^j	18	фестиваль	кофе	верфь
v	18	внучка	синева	
v ^j	18	вертеть	свет	
s	15	сказочник	лесной	карапуз
s ^j	15	сеять	посередине	зажглись
z	15	заяц	звезда	
z ^j	15	земляника	созерцал	
ʂ	15	шуметь	пшено	мышь
ʐ	15	жилище	кружевной	
tʂ	19,15	целитель	Венеция	незнакомец
tɕ	19,16	часы	очарование	мяч
ç:	16	щелчок	ощущать	лещ
m	21	молодежь	несмотря	том
m ^j	21	меч	дымить	семь
n	19	начало	оконце	сон
n ^j	19	небо	линялый	тюлень
l	14	лужа	долгожитель	мел
l ^j	14	лицо	недалеко	соль
r	13	радость	сорока	двор
r ^j	13	рябина	набережная	дверь
j	6	есть	маяк	игрушечный

sk-SK

Vowels for sk-SK

 Expand table

ipa	viseeme	Example 1	Example 2	Example 3
i	6	idea	efektivita	éry
e	4	edícia	absencia	farme
a	2	abnormálne	eskapáda	ária
o	8	oba	banková	esperanto
u	7	udial	február	babu
ɛ	6		mäsité	
i:	6	ílu	edícií	druhý
e:	4	éra	anamnézy	ázijské
a:	2	áno	animácia	druhová
o:	8	ódy	bilión	haló
u:	7	úbočí	absolútна	druhú
ia	6,2		piatkové	maškrtia
ie	6,4		domnienka	námestie
iu	6,7			väčšiu
uo	7,8	ôsma	jahôd	malinô
au	2,7	audio	aplaudovalo	sredau
ou	8,7			
ə	1			

Consonant for sk-SK

 Expand table

ipa	viseme	Example 1	Example 2	Example 3
p	21	pád	apelu	cap
b	21	babička	abiogenézy	dub
t	19	tabak	foto	art
d	19	delta	editor	backhand
c	16	ťahač	docenti	farnosť
č	16	ďalej	stredē	loď
k	20	kabaret	bábkový	chudáčik
g	20	galón	demagógia	mozog
ts	19,15	certifikácie	cicavce	bác
dz	19,15		medzí	
tʃ	19,16	čajový	frčí	boháč
dʒ	19,16	džúsu	brandže	
f	18	fabrík	biografie	fotograf
v	18	vokály	evanjelické	
s	15	sekunda	esá	algoritmus
z	15	zábal	fazulú	aníz
ʃ	16	šablá	duševná	chápeš
ʒ	16	žaba	veži	kaluž
x	12	charakter	biochémie	bohatých
h	12	háčik	bohmi	
r	13	rabat	eróziou	éter
r̩	13		chatrče	leicester
r̩:	13		vŕtal	
l	14	lámpa	električka	čakal
l̩	14		dlhý	nóbľ
l̩:	14		jablk	

ipa	viseme	Example 1	Example 2	Example 3
ʌ	14	l'ad	citeľné	byľ
m	21	meter	emisný	aktom
ŋ	21		amfiteáter	
n	19	nábeh	colnému	fajn
N	19		slovinský	
ŋ	20		banket	
ɲ	19	ňom	ani	jačmeň
u	7		cestovní	aktív
ı	6		fajka	chatovej
j	6	ja	eseje	
w	7	vzbudí	krivdí	

sl-Sl

Vowels for sl-Sl

[Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
ə	1	rjava	pisemski	december
a	2	azilantov	hišam	delniška
a:	2	avto	marketplace	lucia
ɛ	4	edicija	pridem	ničle
e:	4	ebola	pridevnik	prepove
ɛ:	4	ena	producenta	janže
i	6	ideja	pouďarim	pouđariti
i:	6	igla	ilirska	jedmi

ipa	viseme	Example 1	Example 2	Example 3
ɔ	3	oba	morfološke	Marko
ɔ:	3	oče	črnomorskem	
o:	8	občina	reformam	seno
u	7	ulova	mamut	mandatu
u:	7	ura	dramaturgom	intervju

Consonant for sl-Sl

[\[+\]](#) Expand table

ipa	viseme	Example 1	Example 2	Example 3
b	21	brez	bober	lebdeča
d	19	dajal	degradacija	navedbah
d ¹	19	dleto		navedla
dn	19,19	dna		dohodne
đž	19,16	džezovske	bridža	menedžment
đz	19,15	odživajo		Kocbek
f	18	fagota	fotografa	golf
ŋ	21			nimfa
ɣ	20			aħdeloja
g	20	gaber	lagal	ragbi
ɪ	6		pojdi	emisij
j	6	jadra	kajak	najostreje
k	20	kabel	akademija	alkoholik
l	14	labirint	olajša	šal
l ^j	14			poljski
m	21	maček	omara	potem

ipa	viseme	Example 1	Example 2	Example 3
ŋ	20	Ngaliemski_slapovi		banka
n	19	nabave	obarvana	obarvan
n ^j	19			konjska
p	21	pada	sapa	sestop
r	13	rabilia	sorazmerna	spor
s	15	srajca	virusa	virus
ʃ	16	šah	sušijo	tovariš
t	19	tabela	gojiti	flavtist
t ¹	19	tla	škatla	desetletne
tn	19,19			devetnajst
tʃ	19,16	čaj	gneča	hlač
tʂ	19,15	car	raca	rejec
ụ	7			avto
v	18	veja	avanture	
w	7	vgradila		slabokrvnost
m	7	vstatí		
x	12	hiša	jahači	jamah
ž	16	žaba	ježa	možgani
z	15	zmaj	doza	izvoza

sv-SE

Vowels for sv-SE

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	andas	betrakta	gryta
æ	1	ärter	smärtar	
æ:	1	ärliga	besvära	
ɑ:	2	avig	beta	bedra
ɔ	3	åtta	kontrollen	jaså
a <u>u</u>	2,7	aura	pauser	Olau
ə	1		äpple	pojke
e	4	energi	servetten	Arjepluovve
ɛ	4	äpple	berätta	Siviä
ɛ:	4	äta	beträda	trä
e:	4	eka	konkreta	café
œ	8	ört	störta	
œ:	4	ören	beröra	
ø	4	öppen	Alströmer	Päiviö
ɸ:	1	öl	belöning	adjö
i	6	idé	vitsippa	kiwi
i:	6	ivrig	kris	parti
ʊ	4	oas	betrodda	konto
u:	7	oro	förtroende	bero
ɔ:	8	åtala	telefonen	nivå
ə	1	uppenbar	förundrad	farstu
ʉ:	6	ute	bestulen	intervju
y	4	ytterst	rykte	Tommy
y:	4	yta	förtydliga	paraply

Consonant for sv-SE

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
p	21	pil	apa	topp
t	19	tal	matta	akut
k	20	kål	jacka	tak
b	21	bil	klubba	jobb
d	19	dal	lida	stad
g	20	gås	såga	lag
f	18	fil	klaffa	klaff
h	12	hal	behålla	
s	15	sil	visa	viss
ɧ	16	sjuk	maskin	dusch
ç	16	tjock	åkkänslan	coach
v	18	vår	leva	torv
m	21	mil	kamma	arm
n	19	nål	minnas	sömn
ŋ	20		ringa	ung
l	14	lös	måla	tal
r	13	ris	kärra	borr
j	6	jag	tröja	haj
ɖ	19		borda	bord
ɫ	14		porlande	kärl
ɳ	19		gärna	barn
ʂ	15		forsa	fors
t̪	19		skjorta	gjort

th-TH

Vowels for th-TH

[Expand table

ipa	viseme	Example 1	Example 2	Example 3
a	2		ນະ	
a:	2		ຫາດ	ໜາ
e	4		ເຕະ	
e:	4		ເທ	ເດ້
i	6		ປຶດ	
i:	6		ປຶກ	ປີ
ia	6,2		ເງິຍນ	
o	8		ນດ	
o:	8		ແລກໂຕສ	ຈາວໂລ່
ə	1		ໃບຝາກເງິນ	
ə:	1		ເກີນ	ເໜັກ
u	7		ຈຸດ	
u:	7		ດູດ	ດູ
ua	7,2		ກວນ	ຮ້ວ
w	6		ຢືດ	
w:	6		ມືດ	ມືອ
wa	6,2		ເຮືອນ	ເຮືອ
ɛ	4		ຄູ່ແຂງ	
ɛ:	4		ແບນ	ຄຳແປລ
ɔ	3		ນົອຕ	
ɔ:	3		ນອນ	ເຈັບຄວ

Consonant for th-TH

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	บิน	กระปี่	ยมน
t ^ç	19,16	เจ็ด	กระโใจ	
t ^ç h	19,16	เข้าๆ	วันอาสาพหุชชา	
d	19	เดช	วิดีโอ	
f	18	ฟ้า	เพื่องฟู	
h	12	หา	เศษอาหาร	
j	6	ยา	เรือยนต์	yay
k	20	กາ	เนื้อไก่	มาก
k ^h	20	ຄາ	เลขา	
l	14	ลา	หูลาม	
m	21	มา	เสมอ	รัดกุม
n	19	นา	แอนโนเนนซ์	กາລ
ŋ	20	ງາ	แต่งงาน	ແໜ່ງ
p	21	ປາ	โดยทั่วไปแล้ว	ກັນ
p ^h	21	ພາ	ສີກັບ	
r	13	ຮາ	ໃຫ້ມາລາເຮືຍ	
s	15	ສາມ	ໄມ້ມີສົດ	
t	19	ຕາ	ກຕິກາ	ອາຈ
t ^h	19	ທາ	ຄຸນເທຣມ	
w	7	ວົ່ງ	ກວີ	ແກ້ວ
?	19	ອາ	ສະອາດ	

Tone for th-TH

[\[+\] Expand table](#)

ipa	Description	Example
-	Mid	ໄຟ້ ບັບໄມ້ (to go)
˘	Low	ໄຟ້ ຂ້າຍ້ (egg)
˙	Falling	ໃຈ້ ທ້າຍ້ (yes; agreement)
˙	High	ຄວັບ ຂຮາບ້ ([spoken by a male] yes)
˙	Rising	ໜັງ ນັງ້ (cinema film)

tr-TR

Vowels for tr-TR

[\[+\] Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2	armut	fal	elma
a:	2	ağrı	kağıdı	dağ
e	4	erik	kel	akide
e:	4	eğri	değnek	yeg
œ	4	ördek	göl	banliyö
œ̄	4,16	öğlen	açıköğretim	
i	6	ilaç	kıl	kedi
ī	6,16	iğne	mevkii	tebliğ
o	8	orman	kol	vazo
ō	8,16	oğlan	doğru	doğ
u	7	uçak	kuş	koku
ū	7,16	uçur	buğra	başbuğ
ɯ	6	ıhlamur	tıp	kazı
ɯ̄	6,16	ığdır	sığlık	tığ

ipa	viseme	Example 1	Example 2	Example 3
y	4	ülke	gül	öykü
ȳʒ	4,16		düğme	

Consonant for tr-TR

[\[+\]](#) [Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
b	21	balık	ebe	sertab
c	16	keçi	ekşi	bölük
tʃ	19,16	çöp	uçak	ilgeç
d	19	dere	badem	ad
f	18	fil	kefen	çarşaf
g	20	gaz	yorgun	diyalog
ɣ	20		ağır	
ʒ	16	gavur	kevgir	
h	12	halat	ahır	külah
j	6	yer	ayva	olay
dʒ	19,16	cezve	evcimen	hac
k	20	kabak	bakla	çocuk
l	14	leylek	delik	kıl
ɫ	6	lala	kalın	pul
m	21	muz	yemek	kalem
n	19	nar	inek	sorun
ɳ	20		mangal	ring
p	21	para	kapı	rakip
r	19	renk	iri	minder

ipa	viseme	Example 1	Example 2	Example 3
s	15	sal	kısa	bahis
ʃ	16	şekil	koşu	afiş
t	19	terlik	kutup	adalet
v	18	vadi	tava	ev
w	7		tavuk	
z	15	zemin	gezi	kaz
ʒ	16	jöle	angajman	refüj

vi-VN

Vowels for vi-VN

[+] [Expand table](#)

ipa	viseme	Example 1	Example 2	Example 3
a	2		ban	ma
ɛ	4		hép	mẹ
i	6			chí
ɔ	3		học	tò
u	7		hung	thù
ua	7,2			chúa
aj	2,6			tài
ɛj	4,6			lãy
əj	1,6			chơi
o	8			xô
œw	6,4,7			riêu
ɛœ	6,1		khướt	

ipa	viseme	Example 1	Example 2	Example 3
ɔ̄i	3,6			mọi
ə	1		phần	
ie	6,4		biển	
ūj	7,6			mùi
āw	2,7			bảo
ɨ	6		chừng	từ
ɛ	4		nhặt	
ăw	2,7			màu
ăj	2,6			ngày
ɛ̄j	6,1,6			lưới
ōj	8,6			hởi
ə:	1		lợn	sõ
e	4		sên	tế
ɔ̄w	3,7			tấu
ɛ̄w	4,7			béo
īw	6,7			tịu
ɛ̄w	6,7			tựu
ēj	4,6			bêu
ɛ̄w̄	6,1,7			bươu
ɛ̄j	6,6			cửi
I	6	inh	tinh	
iə	6,1			kìa
āj	2	ach	hóach	

Consonant for vi-VN

ipa	viseme	Example 1	Example 2	Example 3
b	21	ba		
k	20	canh		diệc
z	15	diễn		
j	6	giặc		
ɹ	13	róc		
f	18	phụ		
ɣ	20	gà		
h	12	hoa		
l	14	làm		
m	21	mười		năm
n	19	núi		tiền
p	21	pín		pháp
s	15	xào		
ʂ	15	son		
t	19	tuổi		hết
v	18	vàng		
ɖ	19	đón		
ɳ	20	ngủn		lung
x	12	khùng		
ɲ	19	nhàm		sanh
tʰ	19	thông		
t̪	19	trùng		
tʃ	19,16	chim		chỉ
w	7		hoang	

Tone for vi-VN

[\[+\] Expand table](#)

ipa	Description	Example
˧	tone ngang	xem
˨	tone huyền	làm
˧˥	tone sắc	sống
˨˧	tone hỏi	phải
˨˩	tone ngã	cũng
˧˨	tone nặng	một

zh-CN

The Speech service phone set for zh-CN is based on the native phone Pinyin ↗ set.

Pinyin Initials for zh-CN

[\[+\] Expand table](#)

Pinyin	viseme	Character example	sapi example
b	21	玻	bo 1
p	21	坡	po 1
m	21	摸	mo 1
f	18	佛	fo 2
d	19	得	de 2
t	19	特	te 4
n	19	呢	ne 5
l	14	乐	le 4
g	20	哥	ge 1
k	20	科	ke 1

Pinyin	viseme	Character example	sapi example
h	12	喝	he 1
j	16	基	ji 1
q	16	欺	qi 1
x	16	希	xi 1
zh	19, 15	知	zhi 1
ch	19, 15	吃	chi 1
sh	15	诗	shi 1
r	15	日	ri 4
z	15	资	zi 1
c	15	此	ci 3
s	15	思	si 1
y	6	衣	yi 1
w	7	屋	wu 1

Pinyin Finals for zh-CN

[\[+\]](#) Expand table

Pinyin	viseme	Character example	sapi example
a	2, 2 (19, 2, 2 for no initials)	法	fa 3
o	7, 8, 8 (19, 8, 8 for no initials)	泼	po 1
e	1, 1 (19, 1, 1 for no initials)	歌	ge 1
i	6, 6, 6	理	li 3
u	7, 7, 7	步	bu 4
v	7, 4, 4	女	nv 3
ai	2, 4 (19, 2, 4 for no initials)	百	bai 3
ei	4, 6 (19, 4, 6 for no initials)	北	bei 3
ui	7, 4, 6	对	dui 4

Pinyin	viseme	Character example	sapi example
ao	2, 8 (19, 2, 8 for no initials)	号	hao 4
ou	8, 7 (19, 8, 7 for no initials)	走	zou 3
iu	6, 8, 7	牛	niu 2
ie	6, 4, 4	谢	xie 4
ue	7, 4, 4	略	lue 4
er	19, 1, 1	耳	er 3
an	2, 19 (19, 2, 19 for no initials)	喊	han 3
en	1, 19 (19, 1, 19 for no initials)	肯	ken 3
in	6, 6, 19	宾	bin 1
un	7, 1, 19	尊	zun 1
ang	2, 20 (19, 2, 20 for no initials)	朗	lang 3
eng	1, 20	恒	heng 2
ing	6, 1, 20	赢	ying 2
ong	7, 7, 20	红	hong 2
ia	6, 2, 2	家	jia 1
ian	6, 2, 19	面	mian 4
iang	6, 2, 20	象	xiang 4
iao	6, 2, 8	表	biao 3
iong	7, 7, 20	兄	xiong 1
ua	7, 2, 2	花	hua 1
uai	7, 2, 4	帅	shuai 4
uan	7, 2, 19	短	duan 3
uang	7, 2, 20	广	guang 3
uo	7, 8, 8	多	duo 1

Pinyin Whole syllable for zh-CN

[\[+\] Expand table](#)

Pinyin	viseme	Character example	sapi example
zhi	19, 15, 6, 6	知	zhi 1
chi	19, 15, 6, 6	吃	chi 1
shi	15, 6, 6	诗	shi 1
ri	15, 6, 6	日	ri 1
zi	15, 6, 6	资	zi 1
ci	15, 6, 6	词	ci 2
si	15, 6, 6	斯	si 1
yi	6, 6, 6	衣	yi 1
wu	7, 7, 7	屋	wu 1
yu	7, 4, 4	鱼	yu 2
ye	6, 4, 4	叶	ye 4
yue	7, 4, 4	月	yue 4
yuan	7, 2, 19	圆	yuan 2
yin	6, 6, 19	音	yin 1
yun	7, 1, 19	云	yun 1
ying	6, 1, 20	英	ying 1
a	19, 2, 2	啊	a 1
o	19, 8, 8	噢	o 1
e	19, 1, 1	鹅	e 2
ai	19, 2, 4	爱	ai 4
ei	19, 4, 6	欸	ei 1
ao	19, 2, 8	奥	ao 4
ou	19, 8, 7	偶	ou 3
an	19, 2, 19	安	an 1
en	19, 1, 19	恩	en 1

Pinyin	viseme	Character example	sapi example
ang	19, 2, 20	昂	ang 2

Pinyin Tone for zh-CN

[+] Expand table

Pinyin	Character example	sapi example
bā	八	ba 1
bá	拔	ba 2
bǎ	把	ba 3
bà	坝	ba 4
ba	吧	ba 5

Example for zh-CN

[+] Expand table

Character	Speech service
组织关系	zu 3 - zhi 1 - guan 1 - xi 5
累进	lei 3 - jin 4
西宅巷	xi 1 - zhai 2 - xiang 4
一会儿	yi 2 - hui r 4

zh-HK

The Speech service phone set for zh-HK is based on the native phone Jyutping [↗](#) set.

Jyutping Initials for zh-HK

[+] Expand table

Jyutping	Character example	sapi example	VisemeID
p	怕	paa 3	21
b	巴	baa 1	21
t	他	taa 1	19
d	打	daa 2	19
k	卡	kaa 1	20
g	家	gaa 1	20
f	花	faa 1	18
s	沙	saa 1	15
h	蝦	haa 1	12
m	媽	maa 1	21
n	那	naa 5	19
ng	牙	ngaa 4	20
c	叉	caa 1	19 15
z	渣	zaa 1	19 15
l	啦	laa 1	14
kw	誇	kwaa 1	20
gw	瓜	gwaa 1	20
w	蛙	waa 1	7
j	廿	jaa 6	6

Jyutping Middle for zh-HK

[Expand table](#)

Jyutping	Character example	sapi example	VisemeID
aa	沙	saa 1	2
e	些	se 1	4
i	詩	si 1	6

Jyutping	Character example	sapi example	VisemeID
o	疏	so 1	3 In [ou], [o] corresponding viseme is 8
u	夫	fu 1	7
oe	鋸	goe 3	4
yu	書	syu 1	4
a	新	san 1	4
eo	律	leot 6	1

Jyutping Ending for zh-HK

[\[+\]](#) Expand table

Jyutping	Character example	sapi example	VisemeID
i	西	sai 1	6 (In [eoi], [i] corresponding viseme is 4)
u	收	sau 1	7
p	夾	gep 6	21
t	不	bat 1	19
k	策	caak 3	20
m	心	sam 1	21
n	新	san 1	19
ng	敬	ging 3	20

Jyutping Tone for zh-HK

[\[+\]](#) Expand table

Tone number	Description	Character example	sapi example
1	High level/High falling or Entering High Level	詩	si 1
2	Mid Rising	史	si 2
3	Mid Level or Entering Mid Level	試	si 3

Tone number	Description	Character example	sapi example
4	Low Falling	時	si 4
5	Low Rising	市	si 5
6	Low Level or Entering Low Level	是	si 6

zh-TW

The Speech service phone set for zh-TW is based on the native phone [Bopomofo](#) set.

Bopomofo Initials for zh-TW

[\[+\]](#) Expand table

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄅ	b	玻 (ㄅㄢ, bo 1)
ㄆ	p	坡 (ㄆㄢ, po 1)
ㄇ	m	摸 (ㄇㄢ, mo 1)
ㄈ	f	佛 (ㄈㄢ, fo 2)
ㄉ	d	得 (ㄉㄢ, de 2)
ㄊ	t	特 (ㄊㄢ, te 4)
ㄋ	n	呢 (ㄋㄢ, ne 5)
ㄌ	l	樂 (ㄌㄢ, le 4)
ㄍ	g	哥 (ㄍㄢ, ge 1)
ㄎ	k	科 (ㄎㄢ, ke 1)
ㄏ	h	喝 (ㄏㄢ, he 1)
ㄐ	j	基 (ㄐㄧ, ji 1)
ㄑ	q	欺 (ㄑㄧ, qi 1)
ㄒ	x	希 (ㄒㄧ, xi 1)
ㄓ	zh	知 (ㄓㄧ, zhi 1)

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄔ	ch	吃 (ㄔ, chi 1)
ㄕ	sh	詩 (ㄕ, shi 1)
ㄖ	r	日 (ㄖ, ri 4)
ㄔ	z	資 (ㄔ, zi 1)
ㄎ	c	此 (ㄎ, ci 3)
ㄙ	s	思 (ㄙ, si 1)
ㄧ	y	衣 (ㄧ, yi 1)
ㄨ	w	屋 (ㄨ, wu 1)

Bopomofo Finals for zh-TW

[\[+\]](#) Expand table

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄚ	a	法 (ㄞ ㄚ, fa 3)
ㄛ	o	瀝 (ㄞ ㄛ, po 1)
ㄜ	e	歌 (ㄞ ㄜ, ge 1)
ㄧ	i	理 (ㄌㄧˋ, li 3)
ㄨ	u	步 (ㄞ ㄨ, bu 4)
ㄩ	v	女 (ㄞ ㄩ, nv 3)
ㄞ	ai	百 (ㄞ ㄞ, bai 3)
ㄟ	ei	北 (ㄞ ㄟ, bei 3)
ㄞㄟ	ui	對 (ㄞ ㄞ, dui 4)
ㄠ	ao	號 (ㄏ ㄠ, hao 4)
ㄡ	ou	走 (ㄞ ㄡ, zou 3)
ㄧㄡ	iu	牛 (ㄞ ㄧㄡ, niu 2)
ㄧㄢ	ie	謝 (ㄒ ㄧㄢ, xie 4)

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄩㄝ	ue	略 (ㄌㄩㄝ, lue 4)
ㄢ	an	喊 (ㄏㄢˇ, han 3)
ㄣ	en	肯 (ㄻㄣˇ, ken 3)
ㄧㄣ	in	賓 (ㄻㄧㄣ, bin 1)
ㄨㄣ	un	尊 (ㄭㄨㄣ, zun 1)
ㄤ	ang	朗 (ㄌㄤˇ, lang 3)
ㄥ	eng	恆 (ㄏㄥˊ, heng 2)
ㄧㄥ	ing	贏 (ㄧㄥˊ, ying 2)
ㄨㄥ	ong	紅 (ㄏㄨㄥˊ, hong 2)
ㄧㄚ	ia	家 (ㄻㄧㄚ, jia 1)
ㄧㄢ	ian	麵 (ㄇㄧㄢˋ, mian 4)
ㄧㄤ	iang	象 (ㄒㄧㄤˋ, xiang 4)
ㄧㄾ	iao	表 (ㄻㄧㄾ, biao 3)
ㄩㄥ	iong	兄 (ㄒㄩㄥ, xiong 1)
ㄨㄚ	ua	花 (ㄏㄨㄚ, hua 1)
ㄨㄞ	uai	帥 (ㄭㄨㄞˋ, shuai 4)
ㄨㄢ	uan	短 (ㄻㄨㄢˇ, duan 3)
ㄨㄤ	uang	廣 (ㄍㄨㄤˋ, guang 3)
ㄨㄛ	uo	多 (ㄻㄨㄛ, duo 1)

Bopomofo Whole syllable zh-TW

[] Expand table

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄓ	zhi	知 (ㄓ, zhi 1)
ㄔ	chi	吃 (ㄔ, chi 1)

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄕ	shi	诗 (ㄕ, shi 1)
ㄖ	ri	日 (ㄖ, ri 1)
ㄔ	zi	资 (ㄔ, zi 1)
ㄕ	ci	词 (ㄕ, ci 2)
ㄈ	si	斯 (ㄈ, si 1)
ㄧ	yi	衣 (ㄧ, yi 1)
ㄨㄛ	wo	我 (ㄨㄛ, wo 3)
ㄨ	wu	屋 (ㄨ, wu 1)
ㄩ	yu	鱼 (ㄩ, yu 2)
ㄧㄢ	ye	叶 (ㄧㄢ, ye 4)
ㄩㄢ	yue	月 (ㄩㄢ, yue 4)
ㄦ	er	耳 (ㄦ, er 3)
ㄩㄦ	yuan	圆 (ㄩㄦ, yuan 2)
ㄧㄣ	yin	音 (ㄧㄣ, yin 1)
ㄩㄣ	yun	云 (ㄩㄣ, yun 1)
ㄧㄥ	ying	英 (ㄧㄥ, ying 1)
ㄩㄥ	yong	拥 (ㄩㄥ, yong 1)
ㄚ	a	啊 (ㄚ, a 1)
ㄛ	o	噢 (ㄛ, o 1)
ㄜ	e	鹅 (ㄜ, e 2)
ㄞ	ai	爱 (ㄞ, ai 4)
ㄟ	ei	欸 (ㄟ, ei 1)
ㄞ	ao	奥 (ㄞ, ao 4)
ㄡ	ou	偶 (ㄡ, ou 3)
ㄞ	an	安 (ㄞ, an 1)
ㄣ	en	恩 (ㄣ, en 1)

Bopomofo	Pinyin	sapi Example (Bopomofo, Pinyin)
ㄤ	ang	昂 (ㄤ, ang 2)
ㄥ	eng	鞞 (ㄥ, eng 1)
ㄦ	ê	ㄦ (ㄦ, ê 1)

Bopomofo tone zh-TW

[Expand table](#)

Bopomofo	Tone number	sapi Example (Bopomofo, Pinyin)
-	1	八 (ㄩㄚ or ㄩㄚˉ, ba 1)
'	2	拔 (ㄩㄚˊ, ba 2)
ˇ	3	把 (ㄩㄚˇ, ba 3)
ˋ	4	坝 (ㄩㄚˋ, ba 4)
˙	5	吧 (ㄩㄚ˙, ba 5)

Map X-SAMPA to IPA

When using X-SAMPA phone symbols ' and " , it's important to avoid conflicts with the wrapper symbol. If the X-SAMPA string contains phone ' , we recommend using phone _j instead. If you don't want to replace the phone ' , you need to use double quotes " as a wrapper. Similarly, if the X-SAMPA string contains phone " , then double quotes shouldn't be used as a wrapper, and you must use the single quote ' as a wrapper. Otherwise, it will cause an error.

The table below shows a mapping relationship between X-SAMPA (Extended Speech Assessment Methods Phonetic Alphabet) and IPA alphabets. The X-SAMPA symbols are shown at left, with the corresponding IPA symbols to the right.

txt
x-sampa (L) ipa (R)
a a
b b
b_< b

c	c
d	d
d`	ɖ
d_<	ɖ
e	e
f	f
g	g
g_<	ɠ
h	h
h\	ɦ
i	i
j	j
j\	j
k	k
l	l
l`	l
l\	ɿ
m	m
n	n
n`	ɳ
o	o
p	p
p\	ɸ
q	q
r	r
r`	ɾ
r\	ɹ
r\`	ɻ
s	s
s`	ʂ
s\	ç
t	t
t`	t
u	u
v	v
P	ʊ
v\	ʊ
w	w
x	x
x\	ħ
y	y
z	z
z`	ʐ
z\	z
A	a
B	β
B\	b
C	ç
D	ð
E	ɛ
F	ɱ
G	ɣ
G\	g
G_<	g
H	ɥ

H\	ହ
I	ି
I\`	ୟ
J	ଜ
J\`	ଝ
J_<	ଫ
K	କ୍ଷ
K\`	ଖ
L	ଲ୍ୟ
L\`	ଲ
M	ମ
M\`	ପ
N	ନ୍ତ୍ର
N\`	ନ
O	ଓ
O\`	ଥୋ
Q	ପୋ
R	ର୍ବ
R\`	ର
S	ଶ୍ରୀ
T	ଠ୍ଠୀ
U	ୱୁ
U\`	ୱ
V	ଅୱ
W	ମୁୱ
X	ଖୁୱ
X\`	ଖୁୱୀ
Y	ୟୁୱ
Z	୩ୟୁୱ
.	.
"	'
%	'
_j	ଜ
.	ଜ
:	:
:\'	'
@	ଏ
@\`	େ
@`	େୟ
{ }	ଆ
1	ତ୍ତୀ
2	ଫୋଟୋ
3	ବ୍ୟାଙ୍ଗ
3\`	ବ୍ୟାଙ୍ଗୀ
4	ରୂପ
5	ତ୍ତୁ
6	ଅନ୍ତର୍ବାଦ
7	ଯୁଗ୍ମ
8	ଥୋରାମ
9	ଅନ୍ତର୍ବାଦୀ
&	ଅନ୍ତର୍ବାଦୀ
?	କୁଣ୍ଡଳ
?\'	କୁଣ୍ଡଳୀ
<\`	କୁଣ୍ଡଳୀ

>\n^!\n!\\-\n|\\=\n|\\|\n=\\-\n\"-\n+\n-\n/\n_0\n=\n-\n>\n_?\n_^\n_}\n`\n~\n_A\n_a\n_B\n_B_L\n_c\n_d\n_e\n<F>\n_F\n_G\n_H\n_H_T\n_h\n_k\n_L\n_1\n_M\n_m\n_N\n_n\n_O\n_o\n_q\n<R>\n_R\n_R_F\n_r\n_T\n_t\n_v

_W

_X

_X

W

v

x

Lower speech synthesis latency using Speech SDK

08/07/2025

In this article, we introduce the best practices to lower the text to speech synthesis latency and bring the best performance to your end users.

Normally, we measure the latency by `first byte latency` and `finish latency`, as follows:

 Expand table

Latency	Description	SpeechSynthesisResult property key
<code>first byte client latency</code>	Indicates the time delay between the synthesis starts and the first audio chunk is received on the client including network latency.	<code>SpeechServiceResponse_SynthesisFirstByteLatencyMs</code>
<code>finish client latency</code>	Indicates the time delay between the synthesis starts and the whole synthesized audio is received on the client including network latency.	<code>SpeechServiceResponse_SynthesisFinishLatencyMs</code>
<code>network latency</code>	The network latency between the client and Azure TTS service.	<code>SpeechServiceResponse_SynthesisNetworkLatencyMs</code>
<code>first byte service latency</code>	Indicates the time delay between Azure TTS service received synthesis request and the first audio chunk is returned.	<code>SpeechServiceResponse_SynthesisServiceLatencyMs</code>

The Speech SDK puts the latency durations in the Properties collection of `SpeechSynthesisResult`. The following sample code shows these values.

C#

```
var result = await synthesizer.SpeakTextAsync(text);
Console.WriteLine($"first byte client latency:
\{result.Properties.GetProperty(PropertyId.SpeechServiceResponse_SynthesisFirstByteLatencyMs)} ms");
Console.WriteLine($"finish client latency:
\{result.Properties.GetProperty(PropertyId.SpeechServiceResponse_SynthesisFinishLatencyMs)} ms");
Console.WriteLine($"network latency:
\{result.Properties.GetProperty(PropertyId.SpeechServiceResponse_SynthesisNetworkLatencyMs)} ms");
Console.WriteLine($"first byte service latency:
```

```
\t{result.Properties.GetProperty(PropertyId.SpeechServiceResponse_SynthesisServiceLatencyMs)} ms");
// you can also get the result id, and send to us when you need help for diagnosis
var resultId = result.ResultId;
```

The first byte latency is lower than finish latency in most cases. The first byte latency is independent from text length, while finish latency increases with text length.

Ideally, we want to minimize the user-experienced latency (the latency before user hears the sound) to one network route trip time plus the first audio chunk latency of the speech synthesis service.

Streaming

Streaming is critical to lowering latency. Client code can start playback when the first audio chunk is received. In a service scenario, you can forward the audio chunks immediately to your clients instead of waiting for the whole audio.

You can use the [PullAudioOutputStream](#), [PushAudioOutputStream](#), [Synthesizing event](#), and [AudioDataStream](#) of the Speech SDK to enable streaming.

Taking `AudioDataStream` as an example:

```
C#
using (var synthesizer = new SpeechSynthesizer(config, null as AudioConfig))
{
    using (var result = await synthesizer.StartSpeakingTextAsync(text))
    {
        using (var audioDataStream = AudioDataStream.FromResult(result))
        {
            byte[] buffer = new byte[16000];
            uint filledSize = 0;
            while ((filledSize = audioDataStream.ReadData(buffer)) > 0)
            {
                Console.WriteLine($"{filledSize} bytes received.");
            }
        }
    }
}
```

Pre-connect and reuse SpeechSynthesizer

The Speech SDK uses a websocket to communicate with the service. Ideally, the network latency should be one route trip time (RTT). If the connection is newly established, the network

latency includes extra time to establish the connection. The establishment of a websocket connection needs the TCP handshake, SSL handshake, HTTP connection, and protocol upgrade, which introduces time delay. To avoid the connection latency, we recommend pre-connecting and reusing the `SpeechSynthesizer`.

Pre-connect

To pre-connect, establish a connection to the Speech service when you know the connection is needed soon. For example, if you're building a speech bot in client, you can pre-connect to the speech synthesis service when the user starts to talk, and call `SpeakTextAsync` when the bot reply text is ready.

C#

```
using (var synthesizer = new SpeechSynthesizer(uspConfig, null as AudioConfig))
{
    using (var connection = Connection.FromSpeechSynthesizer(synthesizer))
    {
        connection.Open(true);
    }
    await synthesizer.SpeakTextAsync(text);
}
```

ⓘ Note

If the text is available, just call `SpeakTextAsync` to synthesize the audio. The SDK will handle the connection.

Reuse `SpeechSynthesizer`

Another way to reduce the connection latency is to reuse the `SpeechSynthesizer` so you don't need to create a new `SpeechSynthesizer` for each synthesis. We recommend using object pool in service scenario. See our sample code for [C# ↗](#) and [Java ↗](#).

Transmit compressed audio over the network

When the network is unstable or with limited bandwidth, the payload size also affects latency. Meanwhile, a compressed audio format helps to save the users' network bandwidth, which is especially valuable for mobile users.

We support many compressed formats including `opus`, `webm`, `mp3`, `silk`, and so on, see the full list in [SpeechSynthesisOutputFormat](#). For example, the bitrate of `Riff24Khz16BitMonoPcm` format is 384 kbps, while `Audio24Khz48KBitRateMonoMp3` only costs 48 kbps. The Speech SDK automatically uses a compressed format for transmission when a `pcm` output format is set. For Linux and Windows, `GStreamer` is required to enable this feature. Refer [this instruction](#) to install and configure `GStreamer` for Speech SDK. For Android, iOS, and macOS, no extra configuration is needed starting version 1.20.

Input text streaming

Text streaming allows real-time text processing for rapid audio generation. It's perfect for dynamic text vocalization, such as reading outputs from AI models like GPT in real-time. This feature minimizes latency and improves the fluidity and responsiveness of audio outputs, making it ideal for interactive applications, live events, and responsive AI-driven dialogues.

How to use text streaming

Text streaming is supported in C#, C++ and Python with Speech SDK.

To use the text streaming feature, connect to the websocket V2 endpoint:

```
wss://{{region}}.tts.speech.microsoft.com/cognitiveservices/websocket/v2
```

See the sample code for setting the endpoint:

C#

```
// IMPORTANT: MUST use the websocket v2 endpoint
var ttsEndpoint =
    $"wss://{{Environment.GetEnvironmentVariable("AZURE_TTS_REGION")}}.tts.speech.microsoft.com/cognitiveservices/websocket/v2";
var speechConfig = SpeechConfig.FromEndpoint(
    new Uri(ttsEndpoint),
    Environment.GetEnvironmentVariable("AZURE_TTS_API_KEY"));
```

Key steps

- 1. Create a text stream request:** Use `SpeechSynthesisRequestInputType.TextStream` to initiate a text stream.
- 2. Set global properties:** Adjust settings such as output format and voice name directly, as the feature handles partial text inputs and doesn't support SSML. Refer to the following sample code for instructions on how to set them. OpenAI text to speech voices aren't

supported by the text streaming feature. See this [language table](#) for full language support.

```
C#
```

```
// Set output format  
speechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Raw24Khz16BitMonoPcm);  
  
// Set a voice name  
SpeechConfig SetProperty(PropertyId.SpeechServiceConnection_SynthVoice, "en-US-AvaMultilingualNeural");
```

3. **Stream your text:** For each text chunk generated from a GPT model, use

```
request.InputStream.Write(text);
```

to send the text to the stream.

4. **Close the stream:** Once the GPT model completes its output, close the stream using

```
request.InputStream.Close();
```

For detailed implementation, see the [sample code on GitHub](#) ↗

Others tips

Cache CRL files

The Speech SDK uses CRL files to check the certification. Caching the CRL files until expired helps you avoid downloading CRL files every time. See [How to configure OpenSSL for Linux](#) for details.

Use latest Speech SDK

We keep improving the Speech SDK's performance, so try to use the latest Speech SDK in your application.

Load test guideline

You can use load test to test the speech synthesis service capacity and latency. Here are some guidelines:

- The speech synthesis service has the ability to autoscale, but takes time to scale out. If the concurrency is increased in a short time, the client might get long latency or `429` error

code (too many requests). So, we recommend you increase your concurrency step by step in load test. [See this article](#) for more details, especially [this example of workload patterns](#).

- You can use our sample using object pool ([C#](#) and [Java](#)) for load test and getting the latency numbers. You can modify the test turns and concurrency in the sample to meet your target concurrency.
- The service has quota limitation based on the real traffic, therefore, if you want to perform load test with the concurrency higher than your real traffic, connect before your test.

Next steps

- [See the samples](#) on GitHub

Get facial position with viseme

08/07/2025

ⓘ Note

To explore the locales supported for viseme ID and blend shapes, refer to [the list of all supported locales](#). Scalable Vector Graphics (SVG) is only supported for the en-us locale.

A *viseme* is the visual description of a phoneme in spoken language. It defines the position of the face and mouth while a person is speaking. Each viseme depicts the key facial poses for a specific set of phonemes.

You can use visemes to control the movement of 2D and 3D avatar models, so that the facial positions are best aligned with synthetic speech. For example, you can:

- Create an animated virtual voice assistant for intelligent kiosks, building multi-mode integrated services for your customers.
- Build immersive news broadcasts and improve audience experiences with natural face and mouth movements.
- Generate more interactive gaming avatars and cartoon characters that can speak with dynamic content.
- Make more effective language teaching videos that help language learners understand the mouth behavior of each word and phoneme.
- People with hearing impairment can also pick up sounds visually and "lip-read" speech content that shows visemes on an animated face.

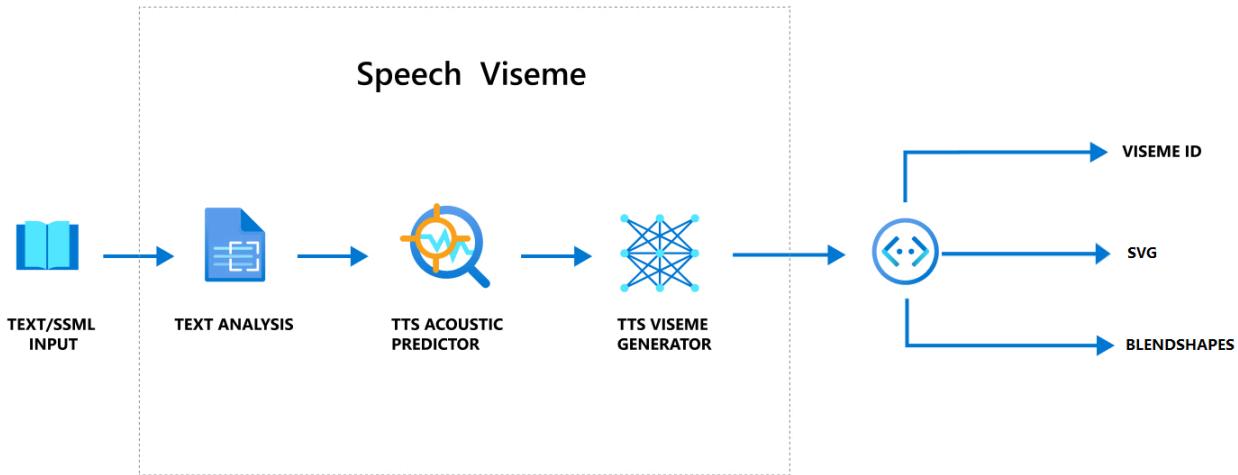
For more information about visemes, view this [introductory video ↗](#).

<https://www.youtube-nocookie.com/embed/ui9XT47uwxs> ↗

Overall workflow of producing viseme with speech

Neural Text to speech (Neural TTS) turns input text or SSML (Speech Synthesis Markup Language) into lifelike synthesized speech. Speech audio output can be accompanied by viseme ID, Scalable Vector Graphics (SVG), or blend shapes. Using a 2D or 3D rendering engine, you can use these viseme events to animate your avatar.

The overall workflow of viseme is depicted in the following flowchart:



Viseme ID

Viseme ID refers to an integer number that specifies a viseme. We offer 22 different visemes, each depicting the mouth position for a specific set of phonemes. There's no one-to-one correspondence between visemes and phonemes. Often, several phonemes correspond to a single viseme, because they looked the same on the speaker's face when they're produced, such as `s` and `z`. For more specific information, see the table for [mapping phonemes to viseme IDs](#).

Speech audio output can be accompanied by viseme IDs and `Audio offset`. The `Audio offset` indicates the offset timestamp that represents the start time of each viseme, in ticks (100 nanoseconds).

Map phonemes to visemes

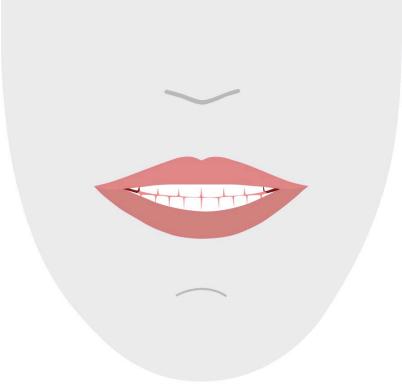
Visemes vary by language and locale. Each locale has a set of visemes that correspond to its specific phonemes. The [SSML phonetic alphabets](#) documentation maps viseme IDs to the corresponding International Phonetic Alphabet (IPA) phonemes. The table in this section shows a mapping relationship between viseme IDs and mouth positions, listing typical IPA phonemes for each viseme ID.

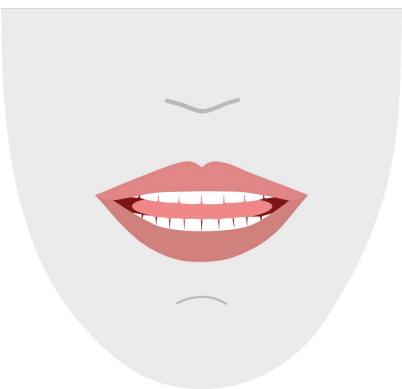
 Expand table

Viseme ID	IPA	Mouth position
0	Silence	
1	æ, ə,ʌ	
2	a	
3	ɔ	

Viseme ID	IPA	Mouth position
4	ɛ, ʊ	
5	ɔ	
6	j, i, ɪ	
7	w, u	

Viseme ID	IPA	Mouth position
8	o	
9	au	
10	ɔɪ	
11	aɪ	

Viseme ID	IPA	Mouth position
12	h	
13	ɥ	
14	l	
15	s, z	

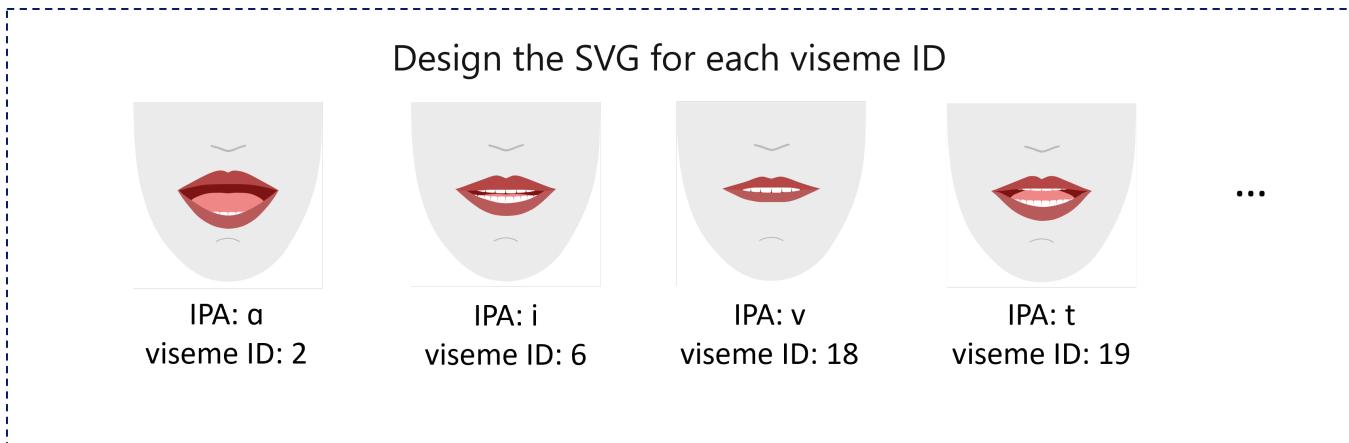
Viseme ID	IPA	Mouth position
16	ʃ, tʃ, dʒ, ʒ	
17	ð	
18	f, v	
19	d, t, n, θ	

Viseme ID	IPA	Mouth position
20	k, g, ŋ	
21	p, b, m	

2D SVG animation

For 2D characters, you can design a character that suits your scenario and use Scalable Vector Graphics (SVG) for each viseme ID to get a time-based face position.

With temporal tags that are provided in a viseme event, these well-designed SVGs are processed with smoothing modifications, and provide robust animation to the users. For example, the following illustration shows a red-lipped character designed for language learning.



3D blend shapes animation

You can use blend shapes to drive the facial movements of a 3D character that you designed.

The blend shapes JSON string is represented as a 2-dimensional matrix. Each row represents a frame. Each frame (in 60 FPS) contains an array of 55 facial positions.

Get viseme events with the Speech SDK

To get viseme with your synthesized speech, subscribe to the `VisemeReceived` event in the Speech SDK.

! Note

To request SVG or blend shapes output, you should use the `mstts:viseme` element in SSML. For details, see [how to use viseme element in SSML](#).

The following snippet shows how to subscribe to the viseme event:

C#

```
using (var synthesizer = new SpeechSynthesizer(speechConfig, audioConfig))
{
    // Subscribes to viseme received event
    synthesizer.VisemeReceived += (s, e) =>
    {
        Console.WriteLine($"Viseme event received. Audio offset: " +
            $"{e.AudioOffset / 10000}ms, viseme id: {e.VisemeId}.");

        // `Animation` is an xml string for SVG or a json string for blend shapes
        var animation = e.Animation;
    };

    // If VisemeID is the only thing you want, you can also use `SpeakTextAsync()`
    var result = await synthesizer.SpeakSsmlAsync(ssml);
}
```

Here's an example of the viseme output.

Viseme ID

```
text

(Viseme), Viseme ID: 1, Audio offset: 200ms.

(Viseme), Viseme ID: 5, Audio offset: 850ms.

.....
```

(Viseme), Viseme ID: 13, Audio offset: 2350ms.

After you obtain the viseme output, you can use these events to drive character animation. You can build your own characters and automatically animate them.

Next steps

- [SSML phonetic alphabets](#)
- [How to improve synthesis with SSML](#)

What are high definition voices?

Azure AI Speech continues to advance in the field of text to speech technology with the introduction of neural text to speech high definition (HD) voices. The HD voices can understand the content, automatically detect emotions in the input text, and adjust the speaking tone in real-time to match the sentiment. HD voices maintain a consistent voice persona from their neural (and non HD) counterparts, and deliver even more value through enhanced features.

Key features of neural text to speech HD voices

The following are the key features of Azure AI Speech HD voices:

 Expand table

Key features	Description
Human-like speech generation	Neural text to speech HD voices can generate highly natural and human-like speech. The model is trained on millions of hours of multilingual data, enabling it to accurately interpret input text and generate speech with the appropriate emotion, pace, and rhythm without manual adjustments.
Conversational	Neural text to speech HD voices can replicate natural speech patterns, including spontaneous pauses and emphasis. When given conversational text, the model can reproduce common phonemes like pauses and filler words. The generated voice sounds as if someone is conversing directly with you.
Prosody variations	Neural text to speech HD voices introduce slight variations in each output to enhance realism. These variations make the speech sound more natural, as human voices naturally exhibit variation.
High fidelity	The primary objective of neural text to speech HD voices is to generate high-fidelity audio. The synthetic speech produced by our system can closely mimic human speech in both quality and naturalness.

Comparison of Azure AI Speech HD voices to other Azure text to speech voices

How do Azure AI Speech HD voices compare to other Azure text to speech voices? How do they differ in terms of features and capabilities?

Here's a comparison of features between Azure AI Speech HD voices, Azure OpenAI HD voices, and Azure AI Speech voices:

Feature	Azure AI Speech HD voices	Azure OpenAI HD voices	Azure AI Speech voices (not HD)
Region	See Speech service regions	See Speech service regions	Available in dozens of regions. See the Speech service regions .
Number of voices	30	6	More than 500
Multilingual	Yes	Yes	Yes (applicable only to multilingual voices)
SSML support	Support for a subset of SSML elements .	Support for a subset of SSML elements .	Support for the full set of SSML in Azure AI Speech.
Development options	Speech SDK, Speech CLI, REST API	Speech SDK, Speech CLI, REST API	Speech SDK, Speech CLI, REST API
Deployment options	Cloud only	Cloud only	Cloud, embedded, hybrid, and containers.
Real-time or batch synthesis	Real-time only	Real-time and batch synthesis	Real-time and batch synthesis
Latency	Less than 300 ms	Greater than 500 ms	Less than 300 ms
Sample rate of synthesized audio	8, 16, 24, and 48 kHz	8, 16, 24, and 48 kHz	8, 16, 24, and 48 kHz
Speech output audio format	opus, mp3, pcm, truesilk	opus, mp3, pcm, truesilk	opus, mp3, pcm, truesilk

Supported Azure AI Speech HD voices

The Azure AI Speech HD voice values are in the format `voicename:basemodel:version`. The name before the colon, such as `en-US-Ava`, is the voice persona name and its original locale. The base model is tracked by versions in subsequent updates.

Currently, `DragonHD` is the only base model available for Azure AI Speech HD voices. To ensure that you're using the latest version of the base model that we provide without having to make a code change, use the `LatestNeural` version.

For example, for the persona `en-US-Ava` you can specify the following HD voice values:

- `en-US-Ava:DragonHDLatestNeural`: Always uses the latest version of the base model that we provide later.

The following table lists the Azure AI Speech HD voices that are currently available.

[Expand table](#)

Voice Name	Gender	Status	Note
de-DE-Florian:DragonHDLatestNeural	Male	GA	
de-DE-Seraphina:DragonHDLatestNeural	Female	GA	
en-US-Adam:DragonHDLatestNeural	Male	GA	
en-US-Alloy:DragonHDLatestNeural	Male	Preview	
en-US-Andrew:DragonHDLatestNeural	Male	GA	
en-US-Andrew2:DragonHDLatestNeural	Male	GA	Optimized for conversational content
en-US-Andrew3:DragonHDLatestNeural	Male	Preview	Optimized for podcast content
en-US-Aria:DragonHDLatestNeural	Female	Preview	
en-US-Ava:DragonHDLatestNeural	Female	GA	
en-US-Ava3:DragonHDLatestNeural	Female	Preview	Optimized for podcast content
en-US-Brian:DragonHDLatestNeural	Male	GA	
en-US-Davis:DragonHDLatestNeural	Male	GA	
en-US-Emma:DragonHDLatestNeural	Female	GA	
en-US-Emma2:DragonHDLatestNeural	Female	GA	Optimized for conversational content
en-US-Jenny:DragonHDLatestNeural	Female	Preview	
en-US-MultiTalker-Ava-Andrew:DragonHDLatestNeural	Male	Preview	
en-US-Nova:DragonHDLatestNeural	Female	Preview	
en-US-Phoebe:DragonHDLatestNeural	Female	Preview	
en-US-Serena:DragonHDLatestNeural	Female	Preview	
en-US-Steffan:DragonHDLatestNeural	Male	GA	
es-ES-Tristan:DragonHDLatestNeural	Male	GA	
es-ES-Ximena:DragonHDLatestNeural	Female	GA	

Voice Name	Gender	Status	Note
fr-FR-Remy:DragonHDLatestNeural	Male	GA	
fr-FR-Vivienne:DragonHDLatestNeural	Female	GA	
ja-JP-Masaru:DragonHDLatestNeural	Male	GA	
ja-JP-Nanami:DragonHDLatestNeural	Female	GA	
zh-CN-Xiaochen:DragonHDLatestNeural	Female	GA	
zh-CN-Yunfan:DragonHDLatestNeural	Male	GA	

How to use Azure AI Speech HD voices

You can use HD voices with the same Speech SDK and REST APIs as the non HD voices.

Here are some key points to consider when using Azure AI Speech HD voices:

- **Voice locale:** The locale in the voice name indicates its original language and region.
- **Base models:**
 - HD voices come with a base model that understands the input text and predicts the speaking pattern accordingly. You can specify the desired model (such as DragonHDLatestNeural) according to the availability of each voice.
- **SSML usage:** To reference a voice in SSML, use the format `voicename:basemodel:version`. The name before the colon, such as `de-DE-Seraphina`, is the voice persona name and its original locale. The base model is tracked by versions in subsequent updates.
- **Temperature parameter:**
 - The temperature value is a float ranging from 0 to 1, influencing the randomness of the output. You can also adjust the temperature parameter to control the variation of outputs. Less randomness yields more stable results, while more randomness offers variety but less consistency.
 - Lower temperature results in less randomness, leading to more predictable outputs. Higher temperature increases randomness, allowing for more diverse outputs. The default temperature is set at 1.0.

Here's an example of how to use Azure AI Speech HD voices in SSML:

ssml

```
<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
xmlns:mstts='https://www.w3.org/2001/mstts' xml:lang='en-US'>
<voice name='en-US-Ava:DragonHDLatestNeural' parameters='temperature=0.8'>Here is a
```

```
test</voice>
</speak>
```

Supported and unsupported SSML elements for Azure AI Speech HD voices

The Speech Synthesis Markup Language (SSML) with input text determines the structure, content, and other characteristics of the text to speech output. For example, you can use SSML to define a paragraph, a sentence, a break or a pause, or silence. You can wrap text with event tags such as bookmark or viseme that your application processes later.

The Azure AI Speech HD voices don't support all SSML elements or events that other Azure AI Speech voices support. Of particular note, Azure AI Speech HD voices don't support [word boundary events](#).

For detailed information on the supported and unsupported SSML elements for Azure AI Speech HD voices, refer to the following table. For instructions on how to use SSML elements, refer to the [Speech Synthesis Markup Language \(SSML\) documentation](#).

[] Expand table

SSML element	Description	Supported in Azure AI Speech HD voices
<code><voice></code>	Specifies the voice and optional effects (<code>eq_car</code> and <code>eq_telecomhp8k</code>).	Yes
<code><mstts:express-as></code>	Specifies speaking styles and roles.	No
<code><mstts:ttseembedding></code>	Specifies the <code>speakerProfileId</code> property for a personal voice.	No
<code><lang xml:lang></code>	Specifies the speaking language.	Yes
<code><prosody></code>	Adjusts pitch, contour, range, rate, and volume.	No
<code><emphasis></code>	Adds or removes word-level stress for the text.	No
<code><audio></code>	Embeds prerecorded audio into an SSML document.	No
<code><mstts:audioduration></code>	Specifies the duration of the output audio.	No
<code><mstts:backgroundaudio></code>	Adds background audio to your SSML documents or mixes an audio file with text to speech.	No

SSML element	Description	Supported in Azure AI Speech HD voices
<phoneme>	Specifies phonetic pronunciation in SSML documents.	No
<lexicon>	Defines how multiple entities are read in SSML.	Yes (only supports alias)
<say-as>	Indicates the content type, such as number or date, of the element's text.	Yes
<sub>	Indicates that the alias attribute's text value should be pronounced instead of the element's enclosed text.	Yes
<math>	Uses the MathML as input text to properly pronounce mathematical notations in the output audio.	No
<bookmark>	Gets the offset of each marker in the audio stream.	No
<break>	Overrides the default behavior of breaks or pauses between words.	No
<mstts:silence>	Inserts pause before or after text, or between two adjacent sentences.	No
<mstts:viseme>	Defines the position of the face and mouth while a person is speaking.	No
<p>	Denotes paragraphs in SSML documents.	Yes
<s>	Denotes sentences in SSML documents.	Yes

! Note

Although a [previous section in this guide](#) also compared Azure AI Speech HD voices to Azure OpenAI HD voices, the SSML elements supported by Azure AI Speech aren't applicable to Azure OpenAI voices.

HD Flash voices

HD voices are currently supported in `eastus`, `westeurope`, and `southeastasia` regions, to provide similar capabilities for customers in China regions (`chinaeast2`, `chinanorth2`, `chinanorth3`), we offer HDFlash versions of selected HD voices. These HDFlash voices deliver

enhanced naturalness compared to standard voices. You also can find those HDFlash voices in `eastus`, `westeurope`, and `southeastasia`.

Below is the complete list of available HD Flash voices:

 Expand table

Voice Name	Gender
zh-CN-Xiaochen:DragonHDFlashLatestNeural	Female
zh-CN-Xiaoxiao:DragonHDFlashLatestNeural	Female
zh-CN-Xiaoxiao2:DragonHDFlashLatestNeural	Female
zh-CN-Yunxia:DragonHDFlashLatestNeural	Male
zh-CN-Yunxiao:DragonHDFlashLatestNeural	Male
zh-CN-Yunye:DragonHDFlashLatestNeural	Male
zh-CN-Yunyi:DragonHDFlashLatestNeural	Male

 Note

HD Flash only support text in `zh-CN` and `en-US`.

Related content

- [Try the text to speech quickstart in Azure AI Speech](#)
- [Learn more about how to use SSML and events](#)

Last updated on 11/06/2025

What is voice conversion? (Preview)

07/10/2025

! Note

This feature is currently in public preview. This preview is provided without a service-level agreement, and is not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Voice conversion is the process of transforming the voice characteristics of a given audio to a target voice speaker. After voice conversion, the resulting audio reserves source audio's linguistic content and prosody while the voice timbre sounds like the target speaker.

There are 3 reasons users need voice conversion functionality:

- Voice conversion can replicate your content using a different voice identity while maintaining the original prosody and emotion. For instance, in education, teachers can record themselves reading stories, and voice conversion can deliver these stories using a pre-designed cartoon character's voice. This method preserves the expressiveness of the teacher's reading while incorporating the unique timbre of the cartoon character's voice.
- Another application is multilingual dubbing. When localized content is read by different voices, voice conversion can transform them into a uniform voice, ensuring a consistent experience across all languages while keeping the most localized voice characters.
- Voice conversion enhances the control over the expressiveness of a voice. By transforming various speaking styles, such as adopting a unique tone or conveying exaggerated emotions, a voice gains greater versatility in expression and can be more dynamic in different scenarios.

Key capabilities

Voice conversion (or voice changer or speech to speech conversion) is built on state-of-the-art generative models and offers high-quality voice conversion. It delivers the following core capabilities:

[+] [Expand table](#)

Capability	Description
High speaker similarity	Captures the timbre and vocal identity of the target speaker.

Capability	Description
	Generates audio that accurately matches the target voice.
Prosody preservation	Maintains rhythm, stress, and intonation of source audio. Preserves expressive and emotional qualities.
High audio fidelity	Generates realistic, natural-sounding audio. Minimizes artifacts.
Multilingual support	Enables multilingual voice conversion. Supports 91 locales (same as standard text to speech locale support). See supported voices for voice conversion for the complete list.

Use voice conversion

You can use Azure AI Speech voice conversion with either the Speech SDK or text to speech REST APIs.

Use the `<mstts:voiceconversion>` tag via Speech Synthesis Markup Language (SSML) to specify the source audio URL and the target voice for the conversion. For a complete list of supported target voices, see [supported voices for voice conversion](#).

Example SSML

XML

```

<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice xml:lang="en-US" xml:gender="Female" name="en-US-
      AvaMultilingualNeural">
      <mstts:voiceconversion
        url="https://your.blob.core.windows.net/sourceaudio.wav"/>
    </voice>
</speak>

```

For details about the SSML structure and usage, see the [Speech Synthesis Markup Language \(SSML\) reference](#) documentation.

Related content

- [Text to speech overview](#)
- [Speech synthesis markup voice](#)

What is custom voice?

08/07/2025

Custom voice is a text to speech feature that lets you create a one-of-a-kind, customized, synthetic voice for your applications. With custom voice, you can build a highly natural-sounding voice for your brand or characters by providing human speech samples as fine-tuning data.

Important

Custom voice access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

Out of the box, [text to speech](#) can be used with standard voices for each [supported language](#). The standard voices work well in most text to speech scenarios if a unique voice isn't required.

Custom voice is based on the neural text to speech technology and the multilingual, multi-speaker, universal model. You can create synthetic voices that are rich in speaking styles, or adaptable across languages. The realistic and natural sounding voice of custom voice can represent brands, personify machines, and allow users to interact with applications conversationally. See the [supported languages](#) for custom voice.

How does it work?

To create a custom voice, use [Speech Studio](#) to upload the recorded audio and corresponding scripts, train the model, and deploy the voice to a custom endpoint.

Creating a great custom voice requires careful quality control in each step, from voice design and data preparation, to the deployment of the voice model to your system.

Before you get started in Speech Studio, here are some considerations:

- [Design a persona](#) of the voice that represents your brand by using a persona brief document. This document defines elements such as the features of the voice, and the character behind the voice. This helps to guide the process of creating a custom voice model, including defining the scripts, selecting your voice talent, training, and voice tuning.
- [Select the recording script](#) to represent the user scenarios for your voice. For example, you can use the phrases from bot conversations as your recording script if you're creating a customer service bot. Include different sentence types in your scripts, including statements, questions, and exclamations.

Here's an overview of the steps to create a custom voice in Speech Studio:

1. [Create a project](#) to contain your data, voice models, tests, and endpoints. Each project is specific to a country/region and language. If you're going to create multiple voices, it's recommended that you create a project for each voice.
2. [Set up voice talent](#). Before you can fine-tune a professional voice, you must submit a recording of the voice talent's consent statement. The voice talent statement is a recording of the voice talent reading a statement that they consent to the usage of their speech data for professional voice fine-tuning.
3. [Prepare fine-tuning data](#) in the right [format](#). It's a good idea to capture the audio recordings in a professional quality recording studio to achieve a high signal-to-noise ratio. The quality of the voice model depends heavily on your fine-tuning data. Consistent volume, speaking rate, pitch, and consistency in expressive mannerisms of speech are required.
4. [Train your voice model](#). Select at least 300 utterances to create a custom voice. A series of data quality checks are automatically performed when you upload them. To build high-quality voice models, you should fix any errors and submit again.
5. [Test your voice](#). Prepare test scripts for your voice model that cover the different use cases for your apps. It's a good idea to use scripts within and outside the training dataset, so you can test the quality more broadly for different content.
6. [Deploy and use your voice model](#) in your apps.

You can tune, adjust, and use your custom voice, similarly as you would use a standard voice. Convert text into speech in real-time, or generate audio content offline with text input. You use the [REST API](#), the [Speech SDK](#), or the [Speech Studio](#).

Tip

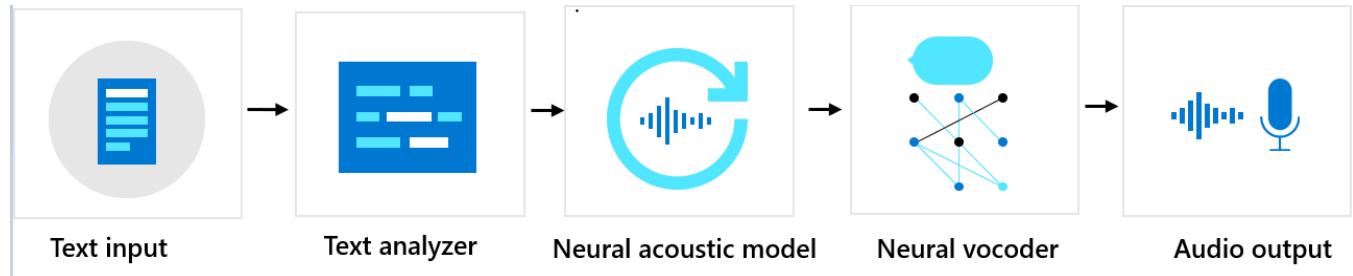
Check out the code samples in the [Speech SDK repository on GitHub](#) to see how to use custom voice in your application.

The style and the characteristics of the trained voice model depend on the style and the quality of the recordings from the voice talent used for training. However, you can make several adjustments by using [SSML \(Speech Synthesis Markup Language\)](#) when you make the API calls to your voice model to generate synthetic speech. SSML is the markup language used to communicate with the text to speech service to convert text into audio. The adjustments you can make include change of pitch, rate, intonation, and pronunciation correction. If the voice model is built with multiple styles, you can also use SSML to switch the styles.

Components sequence

Custom voice consists of three major components: the text analyzer, the neural acoustic model, and the neural vocoder. To generate natural synthetic speech from text, text is first input into the text analyzer, which provides output in the form of phoneme sequence. A *phoneme* is a basic unit of sound that distinguishes one word from another in a particular language. A sequence of phonemes defines the pronunciations of the words provided in the text.

Next, the phoneme sequence goes into the neural acoustic model to predict acoustic features that define speech signals. Acoustic features include the timbre, the speaking style, speed, intonations, and stress patterns. Finally, the neural vocoder converts the acoustic features into audible waves, so that synthetic speech is generated.



Neural text to speech voice models are trained by using deep neural networks based on the recording samples of human voices. For more information, see [this Microsoft blog post](#). To learn more about how a neural vocoder is trained, see [this Microsoft blog post](#).

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who are affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

- [Transparency note and use cases for custom voice](#)
- [Characteristics and limitations for using custom voice](#)
- [Limited access to custom voice](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)
- [Disclosure for voice talent](#)
- [Disclosure design guidelines](#)
- [Disclosure design patterns](#)
- [Code of Conduct for Text to speech integrations](#)
- [Data, privacy, and security for custom voice](#)

Next steps

- [Create a project](#)
- [Prepare fine-tuning data](#)

- Train model

Professional voice fine-tuning data

08/07/2025

When you're ready to create a custom voice for your application, the first step is to gather audio recordings and associated scripts to start professional voice fine-tuning. "Custom voice" is an umbrella term that includes both professional voice fine-tuning and personal voice. The Speech service uses this data for professional voice fine-tuning, creating a unique voice tuned to match the voice in the recordings. After you fine-tune a professional voice, you can start synthesizing speech in your applications.

Tip

To create a voice for production use, we recommend you use a professional recording studio and voice talent. For more information, see [record voice samples for professional voice fine-tuning](#).

Types of data for professional voice fine-tuning

A dataset for professional voice fine-tuning includes audio recordings and a text file with the associated transcriptions. Each audio file should contain a single utterance (a single sentence or a single turn for a dialog system), and be less than 15 seconds long.

In some cases, you might not have the right dataset ready. You can test professional voice fine-tuning with available audio files, short or long, with or without transcripts.

This table lists data types and how each is used for professional voice fine-tuning.

[] Expand table

Data type	Description	When to use	Extra processing required	Processed as
Individual utterances + matching transcript	A collection (.zip) of audio files (.wav) as individual utterances. Each audio file should be 15 seconds or less in length, paired with a formatted transcript (.txt).	Professional recordings with matching transcripts	Ready for fine-tuning.	Segmented
Long audio + transcript	A collection (.zip) of long, unsegmented audio files (.wav or .mp3, longer than 20 seconds, at most 1,000 audio	You have audio files and matching transcripts, but they aren't	Segmentation (using batch transcription).	Segmented, Contextual Audio format

Data type	Description	When to use	Extra processing required	Processed as
	files), paired with a collection (.zip) of transcripts that contains all spoken words.	segmented into utterances.	transformation wherever required.	
Audio only (Preview)	A collection (.zip) of audio files (.wav or .mp3, at most 1,000 audio files) without a transcript.	You only have audio files available, without transcripts.	Segmentation + transcript generation (using batch transcription). Audio format transformation wherever required.	Segmented, Contextual

Files should be grouped by type into a dataset and uploaded as a zip file. Each dataset can only contain a single data type.

ⓘ Note

The maximum number of datasets allowed to be imported per subscription is 500 zip files for standard subscription (S0) users.

Processed as Contextual would retain the audio as a whole to keep the contextual information for more natural intonations.

Individual utterances + matching transcript

You can prepare recordings of individual utterances and the matching transcript in two ways. Either [write a script and have it read by a voice talent](#) or use publicly available audio and transcribe it to text. If you do the latter, edit disfluencies from the audio files, such as "um" and other filler sounds, stutters, mumbled words, or mispronunciations.

To produce a good voice model, create the recordings in a quiet room with a high-quality microphone. Consistent volume, speaking rate, speaking pitch, and expressive mannerisms of speech are essential.

For data format examples, refer to the sample dataset on [GitHub](#). The sample dataset includes the sample script and the associated audio.

Audio data for Individual utterances + matching transcript

Each audio file should contain a single utterance (a single sentence or a single turn of a dialog system), less than 15 seconds long. All files must be in the same spoken language. Multi-language custom Text to speech voices aren't supported, except for the Chinese-English bilingual. Each audio file must have a unique filename with the filename extension .wav.

Follow these guidelines when preparing audio.

[+] Expand table

Property	Value
File format	RIFF (.wav), grouped into a .zip file
File name	File name characters supported by Windows OS, with .wav extension. The characters \ / : * ? " < > \ aren't allowed. It can't start or end with a space, and can't start with a dot. No duplicate file names allowed.
Sampling rate	24 KHz and higher required when fine-tuning a professional voice.
Sample format	PCM, at least 16-bit
Audio length	Shorter than 15 seconds
Archive format	.zip
Maximum archive size	2048 MB

! Note

The default sampling rate for professional voice fine-tuning is 24 KHz. Audio files with a sampling rate lower than 16,000 Hz will be rejected. If a .zip file contains .wav files with different sample rates, only those equal to or higher than 16,000 Hz will be imported. Your audio files with a sampling rate higher than 16,000 Hz and lower than 24 KHz will be up-sampled to 24 KHz for fine-tuning. It's recommended that you use a sample rate of 24 KHz and higher for your fine-tuning data.

Transcription data for Individual utterances + matching transcript

The transcription file is a plain text file. Use these guidelines to prepare your transcriptions.

[+] Expand table

Property	Value
File format	Plain text (.txt)
Encoding format	ANSI, ASCII, UTF-8, UTF-8-BOM, UTF-16-LE, or UTF-16-BE. For zh-CN, ANSI and ASCII encoding aren't supported.
# of utterances per line	One - Each line of the transcription file should contain the name of one of the audio files, followed by the corresponding transcription. You must use a tab (\t) to separate the file name and transcription.
Maximum file size	2048 MB

Here's an example of how the transcripts are organized utterance by utterance in one .txt file:

```
0000000001[tab] This is the waistline, and it's falling.
0000000002[tab] We have trouble scoring.
0000000003[tab] It was Janet Maslin.
```

It's important that the transcripts are 100% accurate transcriptions of the corresponding audio. Errors in the transcripts introduce quality loss during the fine-tuning process.

Long audio + transcript (Preview)

(!) Note

For **Long audio + transcript (Preview)**, only these languages are supported: Chinese (Mandarin, Simplified), Chinese (Cantonese, Traditional), Chinese (Taiwanese Mandarin), English (India), English (United Kingdom), English (United States), French (France), German (Germany), Hindi (India), Italian (Italy), Japanese (Japan), Portuguese (Brazil), Spanish (Spain) and Spanish (Mexico).

Processed as Contextual is currently only available for Chinese (Mandarin, Simplified) and English (United States).

In some cases, you might not have segmented audio available. The Speech Studio can help you segment long audio files and create transcriptions. The long-audio segmentation service uses the [Batch Transcription API](#) feature of speech to text.

The service offers two processing modes:

- **Segmented:** The default processing mode that works with all supported languages
- **Contextual:** An enhanced mode that retains the audio as a whole to keep the contextual information for more natural intonations.

During the processing of the segmentation, your audio files and the transcripts are also sent to the custom speech service to refine the recognition model so the accuracy can be improved for your data. No data is retained during this process. After the segmentation is done, only the utterances segmented and their mapping transcripts will be stored for your downloading and fine-tuning.

Audio data for Long audio + transcript

Follow these guidelines when preparing audio for segmentation.

[] [Expand table](#)

Property	Value
File format	RIFF (.wav) or .mp3, grouped into a .zip file
File name	File name characters supported by Windows OS, with .wav extension. The characters \ / : * ? " < > \ aren't allowed. It can't start or end with a space, and can't start with a dot. No duplicate file names allowed.
Sampling rate	24 KHz and higher required when fine-tuning a professional voice.
Sample format	RIFF(.wav): PCM, at least 16-bit. mp3: At least 256 KBps bit rate.
Audio length	Longer than 30 seconds
Archive format	.zip
Maximum archive size	2048 MB, at most 1,000 audio files included

! Note

The default sampling rate for professional voice fine-tuning is 24 KHz. Audio files with a sampling rate lower than 16,000 Hz will be rejected. Your audio files with a sampling rate higher than 16,000 Hz and lower than 24 KHz will be up-sampled to 24 KHz for fine-tuning. It's recommended that you should use a sample rate of 24 KHz and higher for your fine-tuning data.

Segmented utterances should ideally be between 5 and 15 seconds long. For optimal segmentation results, it is recommended to include natural pauses of 0.5 to 1 second every 5 to 15 seconds of speech, preferably at the end of phrases or sentences.

All audio files should be grouped into a zip file. It's OK to put .wav files and .mp3 files into the same zip file. For example, you can upload a 45-second audio file named 'kingstory.wav' and a 200-second long audio file named 'queenstory.mp3' in the same zip file. All .mp3 files will be transformed into the .wav format after processing.

Transcription data for Long audio + transcript

Transcripts must be prepared to the specifications listed in this table. Each audio file must be matched with a transcript.

[+] [Expand table](#)

Property	Value
File format	Plain text (.txt), grouped into a .zip
File name	Use the same name as the matching audio file
Encoding format	ANSI, ASCII, UTF-8, UTF-8-BOM, UTF-16-LE, or UTF-16-BE. For zh-CN, ANSI and ASCII encoding aren't supported.
# of utterances per line	No limit
Maximum file size	2048 MB

All transcripts files in this data type should be grouped into a zip file. For example, you might upload a 45-second audio file named 'kingstory.wav' and a 200-second long audio file named 'queenstory.mp3' in the same zip file. You need to upload another zip file containing the corresponding two transcripts--one named 'kingstory.txt' and the other one named 'queenstory.txt'. Within each plain text file, you provide the full correct transcription for the matching audio.

After your dataset is successfully uploaded, we'll help you segment the audio file into utterances based on the transcript provided. You can check the segmented utterances and the matching transcripts by downloading the dataset. Unique IDs are assigned to the segmented utterances automatically. It's important that you make sure the transcripts you provide are 100% accurate. Errors in the transcripts can reduce the accuracy during the audio segmentation and further introduce quality loss in the fine-tuning phase that comes later.

Audio only (Preview)

! Note

For **Audio only (Preview)**, only these languages are supported: Chinese (Mandarin, Simplified), Chinese (Cantonese, Traditional), Chinese (Taiwanese Mandarin), English (India), English (United Kingdom), English (United States), French (France), German (Germany), Hindi (India), Italian (Italy), Japanese (Japan), Portuguese (Brazil), Spanish (Spain) and Spanish (Mexico).

Processed as Contextual is currently only available for Chinese (Mandarin, Simplified) and English (United States).

If you don't have transcriptions for your audio recordings, use the **Audio only** option to upload your data. Our system can help you segment and transcribe your audio files.

The service offers two processing modes:

- **Segmented:** The default processing mode that works with all supported languages
- **Contextual:** An enhanced mode that retains the audio as a whole to keep the contextual information for more natural intonations.

Follow these guidelines when preparing audio.

 Expand table

Property	Value
File format	RIFF (.wav) or .mp3, grouped into a .zip file
File name	File name characters supported by Windows OS, with .wav extension. The characters \ / : * ? " < > \ aren't allowed. It can't start or end with a space, and can't start with a dot. No duplicate file names allowed.
Sampling rate	24 KHz and higher required when fine-tuning a professional voice.
Sample format	RIFF(.wav): PCM, at least 16-bit mp3: At least 256 KBps bit rate.
Audio length	No limit
Archive format	.zip
Maximum archive size	2048 MB, at most 1,000 audio files included

Note

The default sampling rate for professional voice fine-tuning is 24 KHz. Your audio files with a sampling rate higher than 16,000 Hz and lower than 24 KHz will be up-sampled to 24 KHz for fine-tuning. It's recommended that you should use a sample rate of 24 KHz and higher for your fine-tuning data.

Segmented utterances should ideally be between 5 and 15 seconds long. For optimal segmentation results, it is recommended to include natural pauses of 0.5 to 1 second every 5 to 15 seconds of speech, preferably at the end of phrases or sentences.

All audio files should be grouped into a zip file. Once your dataset is successfully uploaded, the Speech service helps you segment the audio file into utterances based on our speech batch transcription service. You can select either the Standard or Contextual processing mode, depending on your language and requirements. Unique IDs are assigned to the segmented utterances automatically. Matching transcripts are generated through speech recognition. All .mp3 files will be transformed into the .wav format after processing. You can check the segmented utterances and the matching transcripts by downloading the dataset.

Next steps

- [Train your voice model](#)
- [Deploy and use your voice model](#)
- [How to record voice samples](#)

Recording voice samples for custom voice

08/07/2025

This article provides you with best practices on preparing high-quality voice samples for professional voice fine-tuning. To understand how the data is processed and the minimum requirements for data acceptance, please refer to [upload your data](#).

Creating a high-quality professional voice from scratch isn't a casual undertaking. The central component of a custom voice is a large collection of audio samples of human speech. It's vital that these audio recordings be of high quality. Choose a voice talent who has experience making these kinds of recordings, and have them recorded by a recording engineer using professional equipment.

Before you can make these recordings, though, you need a script: the words are spoken by your voice talent to create the audio samples.

Many small but important details go into creating a professional voice recording. This guide is a roadmap for a process that will help you get good, consistent results.

Tips for preparing data for a high-quality voice

A highly natural custom voice depends on several factors, like the quality and size of your training data.

The quality of your training data is a primary factor. For example, in the same training set, consistent volume, speaking rate, speaking pitch, and speaking style are essential to create a high-quality custom voice. You should also avoid background noise in the recording and make sure the script and recording match. To ensure the quality of your data, you need to follow [script selection criteria](#) and [recording requirements](#).

Regarding the size of the training data, in most cases you can build a reasonable custom voice with 300 utterances. According to our tests, adding more training data in most languages doesn't necessarily improve naturalness of the voice itself (tested using the MOS score), however, with more training data that covers more word instances, you have higher possibility to reduce the ratio of dissatisfactory parts of speech for the voice, such as the glitches. To hear what dissatisfactory parts of speech sound like, refer to [the GitHub examples ↗](#).

In some cases, you might want a voice persona with unique characteristics. For example, a cartoon persona needs a voice with a special speaking style, or a voice that is dynamic in intonation. For such cases, we recommend that you prepare at least 1000 (preferably 2000) utterances, and record them at a professional recording studio. To learn more about how to

improve the quality of your voice model, see [characteristics and limitations for using custom voice](#).

Voice recording roles

There are four basic roles in a custom voice recording project:

 Expand table

Role	Purpose
Voice talent	This person's voice forms the basis of the custom voice.
Recording engineer	Oversees the technical aspects of the recording and operates the recording equipment.
Director	Prepares the script and coaches the voice talent's performance.
Editor	Finalizes the audio files and prepares them for upload to the Speech service.

An individual can fill more than one role. This guide assumes that you are filling the director role and hiring both a voice talent and a recording engineer. If you want to make the recordings yourself, this article includes some information about the recording engineer role. The editor role isn't needed until after the recording session. In the meantime, the director or the recording engineer can fill this role.

Choose your voice talent

Actors with experience in voiceover, voice character work, announcing or news reading make good voice talent. Choose voice talent whose natural voice you like. It's possible to create unique "character" voices, but it's harder for most talent to perform them consistently, and the effort can cause voice strain. The single most important factor for choosing voice talent is consistency. Your recordings for the same voice style should all sound like they were made on the same day in the same room. You can approach this ideal through good recording practices and engineering.

Your voice talent must be able to speak with consistent rate, volume level, pitch, and tone with clear dictation. They also need to be able to control their pitch variation, emotional effect, and speech mannerisms. Recording voice samples can be more fatiguing than other kinds of voice work, so most voice talents can only record for two or three hours a day. Limit sessions to three or four days a week, with a day off in-between if possible.

Work with your voice talent to develop a persona that defines the overall sound and emotional tone of the custom voice. Define the speaking styles for your persona and ask your voice talent to read the script in a way that aligns with your desired styles. Ensure that the speaking style remains consistent throughout the recordings for a set of training data.

For example, a persona with a naturally upbeat personality would carry a note of optimism in their voice. However, this personality should be expressed consistently across all recordings for a set of training data. Listen to existing voices to get a sense of what you're aiming for.

Tip

Usually, you'll want to own the voice recordings you make. Your voice talent should be amenable to a work-for-hire contract for the project.

Create a script

The starting point of any custom voice recording session is the script, which contains the utterances to be spoken by your voice talent. The term "utterances" encompasses both full sentences and shorter phrases. Building a custom voice requires at least 300 recorded utterances as training data.

The utterances in your script can come from anywhere: fiction, non-fiction, transcripts of speeches, news reports, and anything else available in printed form. For a brief discussion of potential legal issues, see the "[Legalities](#)" section. You can also write your own text.

Your utterances don't need to come from the same source, the same kind of source, or have anything to do with each other. However, if you use set phrases (for example, "You have successfully logged in") in your speech application, make sure to include them in your script. It gives your custom voice a better chance of pronouncing those phrases well.

We recommend the recording scripts include both general sentences and domain-specific sentences. For example, if you plan to record 2,000 sentences, 1,000 of them could be general sentences, another 1,000 of them could be sentences from your target domain or the use case of your application.

We provide [sample scripts in the 'General', 'Chat' and 'Customer Service' domains for each language](#) to help you prepare your recording scripts. You can use these Microsoft shared scripts for your recordings directly or use them as a reference to create your own.

Script selection criteria

Below are some general guidelines that you can follow to create a good corpus (recorded audio samples) for professional voice fine-tuning.

- For most use cases, sentences are recommended to be between 2 and 15 seconds long, containing 5 to 30 words for Latin-based languages or 4 to 80 words for non-Latin languages. Aim to balance your script to include a variety of sentence types and lengths. Ensure your script does not include any duplicate sentences.

If your use case requires a high emphasis on questions, exclamations, or a mix of particularly long and short sentences, it is recommended to include a good portion of sentences as questions or exclamations, along with very short phrases and longer phrases up to 20 seconds in length.

For how to balance the different sentence types, refer to the following table:

 Expand table

Sentence types	Coverage
Statement sentences	Statement sentences should be 70-80% of the script.
Short word/phrase	Short word/phrase scripts should be about 10% of total utterances, with 5 to 7 words per case. Short words or phrases should be separated by commas to help remind voice talent to pause briefly while reading.
Question sentences (Optional)	Question sentences should be about 10%-20% of your domain script, including 5%-10% of rising and 5%-10% of falling tones. These sentences are required if you want the generated voice to accurately convey questions.
Exclamation sentences (Optional)	Exclamation sentences should be about 10%-20% of your script. These sentences are required if you want the generated voice to accurately convey exclamations.

 Note

You can estimate the number of words in a sentence by assuming a speech rate in words per second based on your language.

Best practices include:

- Balanced coverage for Parts of Speech, like verbs, nouns, adjectives, and so on.
- Balanced coverage for pronunciations. Include all letters from A to Z so the Text to speech engine learns how to pronounce each letter in your style.
- Readable, understandable, common-sense scripts for the speaker to read.

- Avoid too many similar patterns for words/phrases, like "easy" and "easier".
- Include different formats of numbers: address, unit, phone, quantity, date, and so on, in all sentence types.
- Include spelling sentences if it's something your custom voice will read. For example, "The spelling of Apple is A P P L E".

ⓘ Note

For Contextual processing mode, which provides more natural intonations and better conversational capabilities:

- Use paragraph-level text rather than sentence-level text for recordings. This approach helps capture natural speech flow between sentences and preserves contextual information.
- Each recording should ideally be longer than 30 seconds (containing more than 60 words for Latin-based languages or 160 words for non-Latin languages).
- A contextual training set with more than 30 minutes of total audio or 300 utterances can be used for training a custom voice.

- Don't put multiple sentences into one line/one utterance. Separate each line by utterance.
- Make sure the sentence is clean. Generally, don't include too many nonstandard words like numbers or abbreviations as they're hard to read. Some applications might require the reading of many numbers or acronyms. In these cases, you can include these words, but normalize them in their spoken form.

Below are some best practices for example:

- For lines with abbreviations, instead of "BTW", write "by the way".
- For lines with digits, instead of "911", write "nine one one".
- For lines with acronyms, instead of "ABC", write "A B C".

With that, make sure your voice talent pronounces these words in an expected way. Keep your script and recordings matched during the training process.

- Your script should include many different words and sentences with different kinds of sentence lengths, structures, and moods.
- Check the script carefully for errors. If possible, have someone else check it too. When you run through the script with your voice talent, you might catch more mistakes.

Difference between voice talent script and training script

The training script can differ from the voice talent script, especially for scripts that contain digits, symbols, abbreviations, date, and time. Scripts prepared for the voice talent must follow native reading conventions, such as 50% and \$45. The scripts used for training must be normalized to match the audio recording, such as *fifty percent* and *forty-five dollars*.

 **Note**

We provide some example scripts for the voice talent on [GitHub](#). To use the example scripts for training, you must normalize them according to the recordings of your voice talent before uploading the file.

The following table shows the difference between scripts for voice talent and the normalized script for training.

 [Expand table](#)

Category	Voice talent script example	Training script example (normalized)
Digits	123	one hundred and twenty-three
Symbols	50%	fifty percent
Abbreviation	ASAP	as soon as possible
Date and time	March 3rd at 5:00 PM	March third at five PM

Typical defects of a script

The script's poor quality can adversely affect the training results. To achieve high-quality training results, it's crucial to avoid defects.

Script defects generally fall into the following categories:

 [Expand table](#)

Category	Example
Meaningless content.	"Colorless green ideas sleep furiously."
Incomplete sentences.	- "This was my last eve" (no subject, no specific meaning) - "They're already funny (no quote mark in the end, it's not a complete sentence)
Typo in the sentences.	- Start with a lower case - No ending punctuation if needed

Category	Example
	<ul style="list-style-type: none"> - Misspelling - Lack of punctuation: no period in the end (except news title) - End with symbols, except comma, question, exclamation - Wrong format, such as: <ul style="list-style-type: none"> - 45\$ (should be \$45) - No space or excess space between word/punctuation
Duplication in similar format, one per each pattern is enough.	<ul style="list-style-type: none"> - "Now is 1pm in New York" - "Now is 2pm in New York" - "Now is 3pm in New York" - "Now is 1pm in Seattle" - "Now is 1pm in Washington D.C."
Uncommon foreign words: only commonly used foreign words are acceptable in the script.	In English one might use the French word "faux" in common speech, but a French expression such as "coincer la bulle" would be uncommon.
Emoji or any other uncommon symbols	

Script format

The script is for use during recording sessions, so you can set it up any way you find easy to work with. Create the text file that's required by Speech Studio separately.

A basic script format contains three columns:

- The number of the utterance, starting at 1. Numbering makes it easy for everyone in the studio to refer to a particular utterance ("let's try number 356 again"). You can use the Microsoft Word paragraph numbering feature to number the rows of the table automatically.
- A blank column where you write the take number or time code of each utterance to help you find it in the finished recording.
- The text of the utterance itself.



Time Code	Utterance
1	Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.
2	Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure.
3	We are met on a great battlefield of that war.
4	We have come to dedicate a portion of that field as a final resting place for those who here gave their lives that that nation might live.

ⓘ Note

Most studios record in short segments known as "takes". Each take typically contains 10 to 24 utterances. Just noting the take number is sufficient to find an utterance later. If you're recording in a studio that prefers to make longer recordings, you'll want to note the time code instead. The studio will have a prominent time display.

Leave enough space after each row to write notes. Be sure that no utterance is split between pages. Number the pages, and print your script on one side of the paper.

Print three copies of the script: one for the voice talent, one for the recording engineer, and one for the director (you). Use a paper clip instead of staples: an experienced voice artist separates the pages to avoid making noise as the pages are turned.

Voice talent statement

To train a neural voice, you must [create a voice talent profile](#) with an audio file recorded by the voice talent consenting to the usage of their speech data to fine-tune a professional voice model. When preparing your recording script, make sure you include the statement sentence.

Legalities

Under copyright law, an actor's reading of copyrighted text might be a performance for which the author of the work should be compensated. This performance won't be recognizable in the final product, the custom voice. Even so, the legality of using a copyrighted work for this

purpose isn't well established. Microsoft can't provide legal advice on this issue; consult your own legal counsel.

Fortunately, it's possible to avoid these issues entirely. There are many sources of text you can use without permission or license.

 Expand table

Text source	Description
CMU Arctic corpus 	About 1100 sentences selected from out-of-copyright works specifically for use in speech synthesis projects. An excellent starting point.
Works no longer under copyright	Typically works published prior to 1923. For English, Project Gutenberg  offers tens of thousands of such works. You might want to focus on newer works, as the language is closer to modern English.
Government works	Works created by the United States government aren't copyrighted in the United States, though the government can claim copyright in other countries/regions.
Public domain	Works for which copyright is explicitly disclaimed or dedicated to the public domain. It might not be possible to waive copyright entirely in some jurisdictions.
Permissively licensed works	Works distributed under a license like Creative Commons or the GNU Free Documentation License (GFDL). Wikipedia uses the GFDL. Some licenses, however, may impose restrictions on performance of the licensed content that might affect the creation of a custom voice model, so read the license carefully.

Recording your script

Record your script at a professional recording studio that specializes in voice work. They have a recording booth, the right equipment, and the right people to operate it. It's recommended not to skimp on recording.

Discuss your project with the studio's recording engineer and listen to their advice. The recording should have little or no dynamic range compression (maximum of 4:1). It's critical that the audio has consistent volume and a high signal-to-noise ratio, while being free of unwanted sounds.

Recording requirements

To achieve high-quality training results, follow the following requirements during recording or data preparation:

- Clear and well pronounced

- Natural speed: not too slow or too fast between audio files.
- Appropriate volume, prosody and break: stable within the same sentence or between sentences, correct break for punctuation.
- No noise during recording
- Fit your persona design
- No wrong accent: fit to the target design
- No wrong pronunciation

You can refer to below specification to prepare for the audio samples as best practice.

 Expand table

Property	Value
File format	*.wav, Mono
Sampling rate	24 KHz
Sample format	16 bit, PCM
Peak volume levels	-3 dB to -6 dB
SNR	> 35 dB
Silence	<ul style="list-style-type: none"> - There should have some silence (recommend 100 ms) at the beginning and ending, but no longer than 200 ms - Silence between words or phrases < -30 dB - Silence in the wave after last word is spoken <-60 dB
Environment noise or echo	<ul style="list-style-type: none"> - The level of noise at start of the wave before speaking < -70 dB

Note

You can record at higher sampling rate and bit depth, for example in the format of 48 KHz 24 bit PCM. During the professional voice fine-tuning, we'll down sample it to 24 KHz 16 bit PCM automatically.

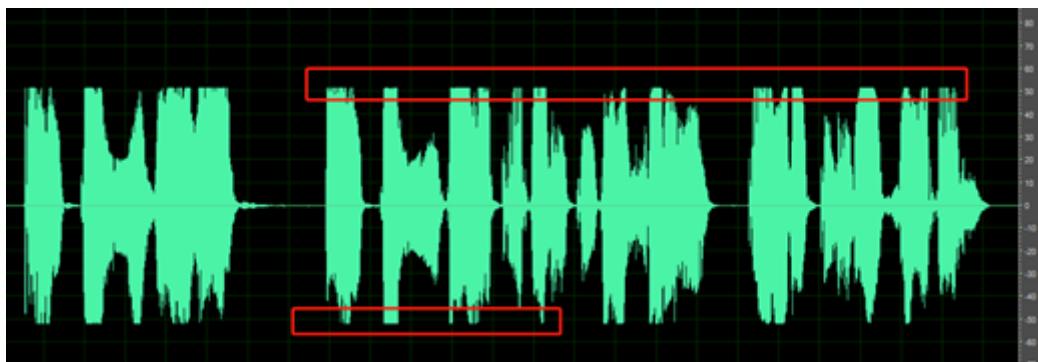
A higher signal-to-noise ratio (SNR) indicates lower noise in your audio. You can typically reach a 35+ SNR by recording at professional studios. Audio with an SNR below 20 can result in obvious noise in your generated voice.

Consider re-recording any utterances with low pronunciation scores or poor signal-to-noise ratios. If you can't re-record, consider excluding those utterances from your data.

Typical audio errors

For high-quality training results, avoiding audio errors is highly recommended. Audio errors are usually within following categories:

- Audio file name doesn't match the script ID.
- WAR file has an invalid format and can't be read.
- Audio sampling rate is lower than 16 KHz. It's recommended that the .wav file sampling rate be equal or higher than 24 KHz for high-quality neural voice.
- Volume peak isn't within the range of -3 dB (70% of max volume) to -6 dB (50%).
- Waveform overflow: the waveform is cut at its peak value and is thus not complete.

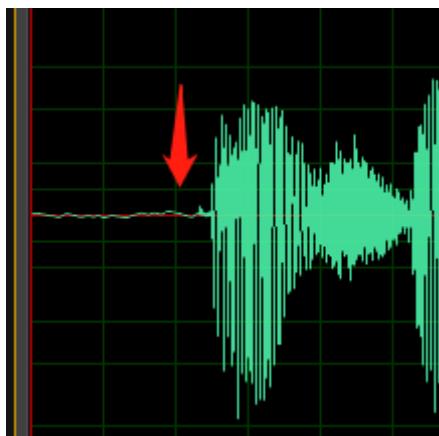


- The silent parts of the recording aren't clean; you can hear sounds such as ambient noise, mouth noise and echo.

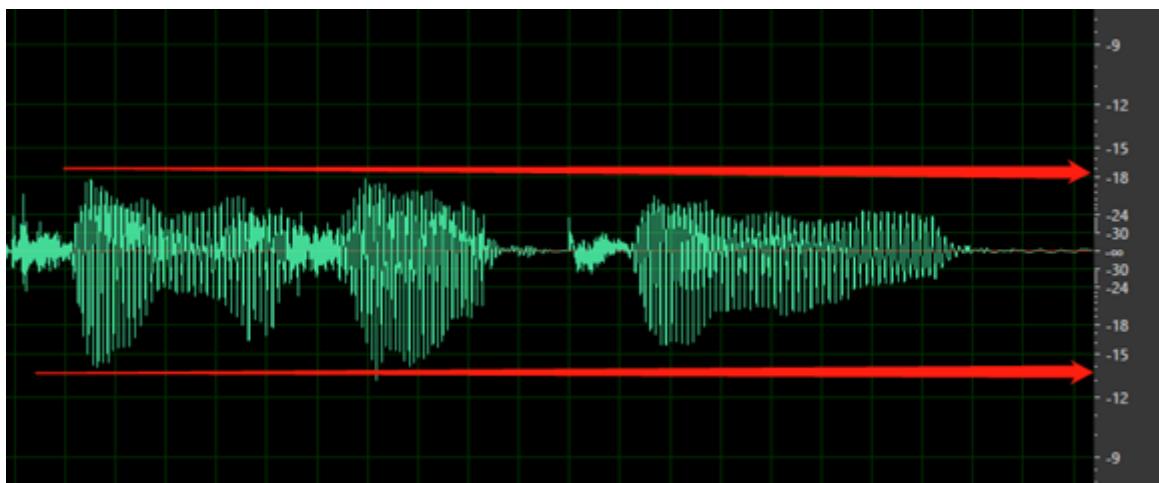
For example, below audio contains the environment noise between speeches.



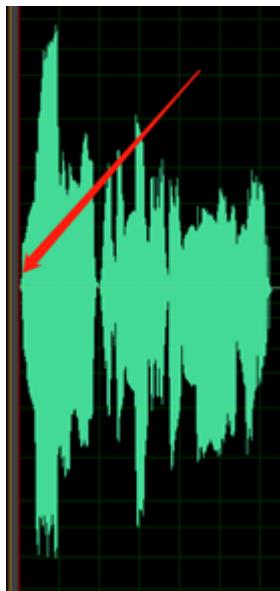
Below sample contains signs of DC offset or echo.



- The overall volume is too low. Your data is tagged as an issue if the volume is lower than -18 dB (10% of max volume). Make sure all audio files should be consistent at the same level of volume.



- No silence before the first word or after the last word. Also, the start or end silence shouldn't be longer than 200 ms or shorter than 100 ms.



Do it yourself

If you want to make the recording yourself, instead of going into a recording studio, here's a short primer. Thanks to the rise of home recording and podcasting, it's easier than ever to find good recording advice and resources online.

Your "recording booth" should be a small room with no noticeable echo or "room tone." It should be as quiet and soundproof as possible. Drapes on the walls can be used to reduce echo and neutralize or "deaden" the sound of the room.

Use a high-quality studio condenser microphone ("mic" for short) intended for recording voice. Sennheiser, AKG, and even newer Zoom mics can yield good results. You can buy a mic, or rent one from a local audio-visual rental firm. Look for one with a USB interface. This type of mic conveniently combines the microphone element, preamp, and analog-to-digital converter into one package, simplifying hookup.

You can also use an analog microphone. Many rental houses offer "vintage" microphones known for their voice character. Professional analog gear uses balanced XLR connectors, rather than the 1/4-inch plug that's used in consumer equipment. If you go analog, you'll also need a preamp and a computer audio interface with these connectors.

Install the microphone on a stand or boom, and install a pop filter in front of the microphone to eliminate noise from "plosive" consonants like "p" and "b." Some microphones come with a suspension mount that isolates them from vibrations in the stand, which is helpful.

The voice talent must stay at a consistent distance from the microphone. Use tape on the floor to mark where they should stand. If the talent prefers to sit, take special care to monitor mic

distance and avoid chair noise.

Use a stand to hold the script. Avoid angling the stand so that it can reflect sound toward the microphone.

The person operating the recording equipment — the recording engineer — should be in a separate room from the talent, with some way to talk to the talent in the recording booth (a *talkback circuit*).

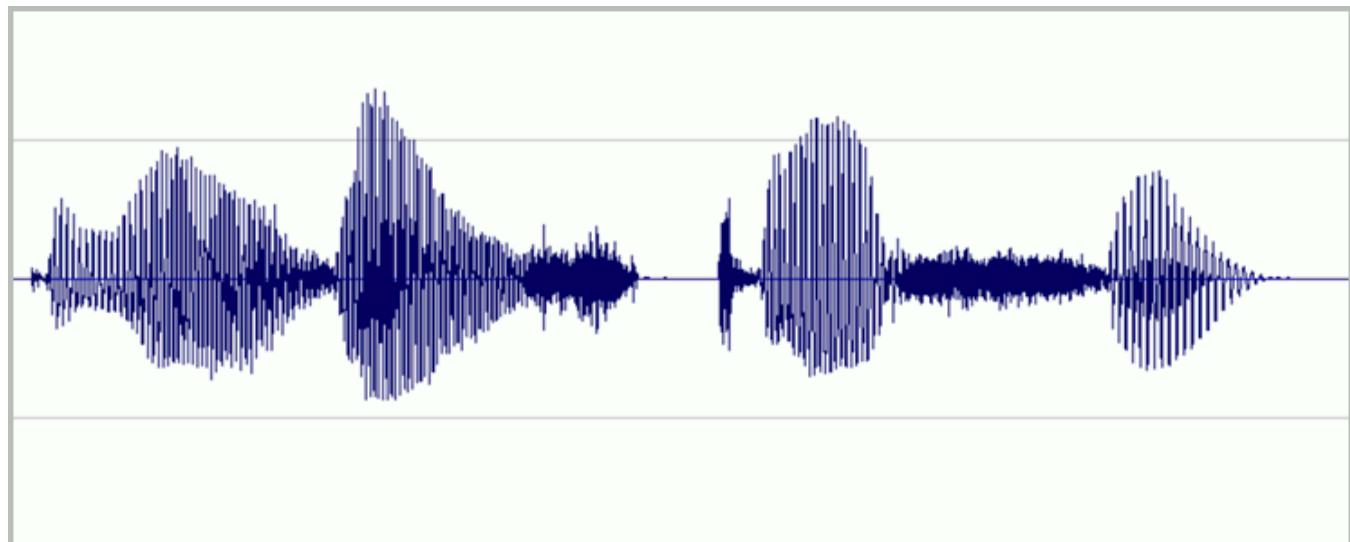
The recording should contain as little noise as possible, with a goal of -80 dB.

Listen closely to a recording of silence in your "booth," figure out where any noise is coming from, and eliminate the cause. Common sources of noise are air vents, fluorescent light ballasts, traffic on nearby roads, and equipment fans (even notebook PCs might have fans). Microphones and cables can pick up electrical noise from nearby AC wiring, usually a hum or buzz. A buzz can also be caused by a *ground loop*, which is caused by having equipment plugged into more than one electrical circuit.

Tip

In some cases, you might be able to use an equalizer or a noise reduction software plug-in to help remove noise from your recordings, although it is always best to stop it at its source.

Set levels so that most of the available dynamic range of digital recording is used without overdriving. That means set the audio loud, but not so loud that it becomes distorted. An example of the waveform of a good recording is shown in the following image:



Here, most of the range (height) is used, but the highest peaks of the signal don't reach the top or bottom of the window. You can also see that the silence in the recording approximates a

thin horizontal line, indicating a low noise floor. This recording has acceptable dynamic range and signal-to-noise ratio.

Record directly into the computer via a high-quality audio interface or a USB port, depending on the mic you're using. For analog, keep the audio chain simple: mic, preamp, audio interface, computer. You can license both [Avid Pro Tools](#) and [Adobe Audition](#) monthly at a reasonable cost. If your budget is extremely tight, try the free [Audacity](#).

Record at 44.1 KHz 16 bit monophonic (CD quality) or better. Current state-of-the-art is 48 KHz 24 bit, if your equipment supports it. You'll down-sample your audio to 24 KHz 16-bit before you submit it to Speech Studio. Still, it pays to have a high-quality original recording in the event that edits are needed.

Ideally, have different people serve in the roles of director, engineer, and talent. Don't try to do it all yourself. In a pinch, one person can be both the director and the engineer.

Before the session

To avoid wasting studio time, run through the script with your voice talent before the recording session. While the voice talent becomes familiar with the text, they can clarify the pronunciation of any unfamiliar words.

! Note

Most recording studios offer electronic display of scripts in the recording booth. In this case, type your run-through notes directly into the script's document. You'll still want a paper copy to take notes on during the session, though. Most engineers will want a hard copy, too. And you'll still want a third printed copy as a backup for the talent in case the computer is down.

Your voice talent might ask which word you want emphasized in an utterance (the "operative word"). Tell them that you want a natural reading with no particular emphasis. Emphasis can be added when speech is synthesized; it shouldn't be a part of the original recording.

Direct the talent to pronounce words distinctly. Every word of the script should be pronounced as written. Sounds shouldn't be omitted or slurred together, as is common in casual speech, *unless they have been written that way in the script*.

[] [Expand table](#)

Written text	Unwanted casual pronunciation
never going to give you up	never gonna give you up
there are four lights	there're four lights
how's the weather today	how's th' weather today
say hello to my little friend	say hello to my lil' friend

The talent shouldn't add distinct pauses between words. The sentence should still flow naturally, even while sounding a little formal. This fine distinction might take practice to get right.

The recording session

Create a reference recording, or *match file*, of a typical utterance at the beginning of the session. Ask the talent to repeat this line every page or so. Each time, compare the new recording to the reference. This practice helps the talent remain consistent in volume, tempo, pitch, and intonation. Meanwhile, the engineer can use the match file as a reference for levels and overall consistency of sound.

The match file is especially important when you resume recording after a break or on another day. Play it a few times for the talent and have them repeat it each time until they're matching well.

To record a corpus with a specific style, carefully choose scripts that showcase the desired style. During recording, ensure the voice talent maintains consistent in volume, tempo, pitch, and tone to achieve recordings that embody the intended style.

Coach your talent to take a deep breath and pause for a moment before each utterance. Record a couple of seconds of silence between utterances. Words should be pronounced the same way each time they appear, considering context. For example, "record" as a verb is pronounced differently from "record" as a noun.

Record approximately five seconds of silence before the first recording to capture the "room tone". This practice helps Speech Studio compensate for noise in the recordings.

Tip

All you need to capture is the voice talent, so you can make a monophonic (single-channel) recording of just their lines. However, if you record in stereo, you can use the second channel to record the chatter in the control room to capture discussion of

particular lines or takes. Remove this track from the version that's uploaded to Speech Studio.

Listen closely, using headphones, to the voice talent's performance. You're looking for good but natural diction, correct pronunciation, and a lack of unwanted sounds. Don't hesitate to ask your talent to re-record an utterance that doesn't meet these standards.

Tip

If you are using a large number of utterances, a single utterance might not have a noticeable effect on the resultant custom voice. It might be more expedient to simply note any utterances with issues, exclude them from your dataset, and see how your custom voice turns out. You can always go back to the studio and record the missed samples later.

Note the take number or time code on your script for each utterance. Ask the engineer to mark each utterance in the recording's metadata or cue sheet as well.

Take regular breaks and provide a beverage to help your voice talent keep their voice in good shape.

After the session

Modern recording studios run on computers. At the end of the session, you receive one or more audio files, not a tape. These files are probably WAV or AIFF format in CD quality (44.1 KHz 16-bit) or better. 24 KHz 16-bit is common and desirable. The default sampling rate for a custom voice is 24 KHz. It's recommended that you should use a sample rate of 24 KHz and higher for your training data. Higher sampling rates, such as 96 KHz, aren't usually needed.

Speech Studio requires each provided utterance to be in its own file. Each audio file delivered by the studio contains multiple utterances. So the primary post-production task is to split up the recordings and prepare them for submission. The recording engineer might have placed markers in the file (or provided a separate cue sheet) to indicate where each utterance starts.

Use your notes to find the exact takes you want, and then use a sound editing utility, such as [Avid Pro Tools](#), [Adobe Audition](#), or the free [Audacity](#), to copy each utterance into a new file.

Listen to each file carefully. At this stage, you can edit out small unwanted sounds that you missed during recording, like a slight lip smack before a line, but be careful not to remove any actual speech. If you can't fix a file, remove it from your dataset and note that you've done so.

Convert each file to 16 bits and a sample rate of 24 KHz and higher before saving and if you recorded the studio chatter, remove the second channel. Save each file in WAV format, naming the files with the utterance number from your script.

Finally, create the transcript that associates each WAV file with a text version of the corresponding utterance. [Train your voice model](#) includes details of the required format. You can copy the text directly from your script. Then create a Zip file of the WAV files and the text transcript.

Archive the original recordings in a safe place in case you need them later. Preserve your script and notes, too.

Next steps

You're ready to upload your recordings and create your custom voice.

[Train your voice model](#)

Create a project for professional voice

05/20/2025

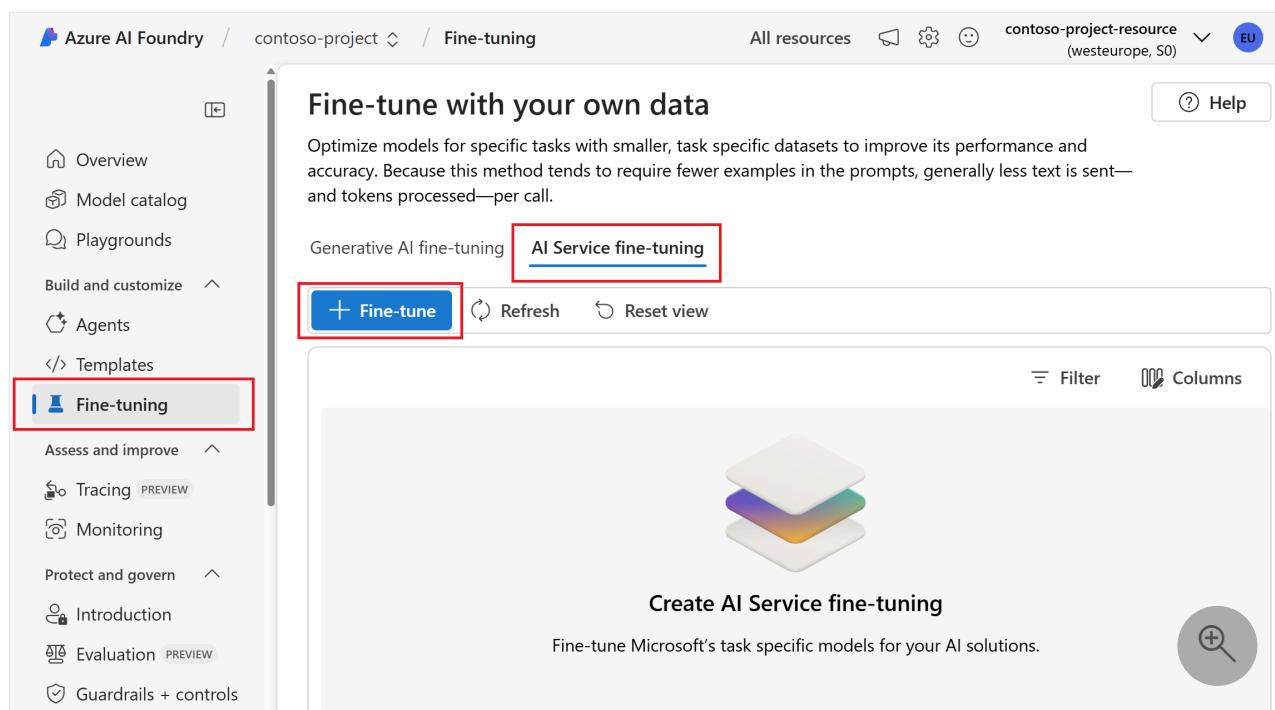
All it takes to get started are a handful of audio files and the associated transcriptions. See if custom voice supports your [language](#) and [region](#).

Start fine-tuning

In the [Azure AI Foundry portal](#), you can fine-tune some Azure AI services models. For example, you can fine-tune a professional voice model.

To fine-tune a professional voice model, follow these steps:

1. Go to your Azure AI Foundry project in the [Azure AI Foundry portal](#). If you need to create a project, see [Create an Azure AI Foundry project](#).
2. Select **Fine-tuning** from the left pane.
3. Select **AI Service fine-tuning > + Fine-tune**.



4. In the wizard, select **Custom voice (professional voice fine-tuning)**.
5. Select **Next**.
6. Follow the instructions provided by the wizard to create your fine-tuning workspace.

Continue fine-tuning

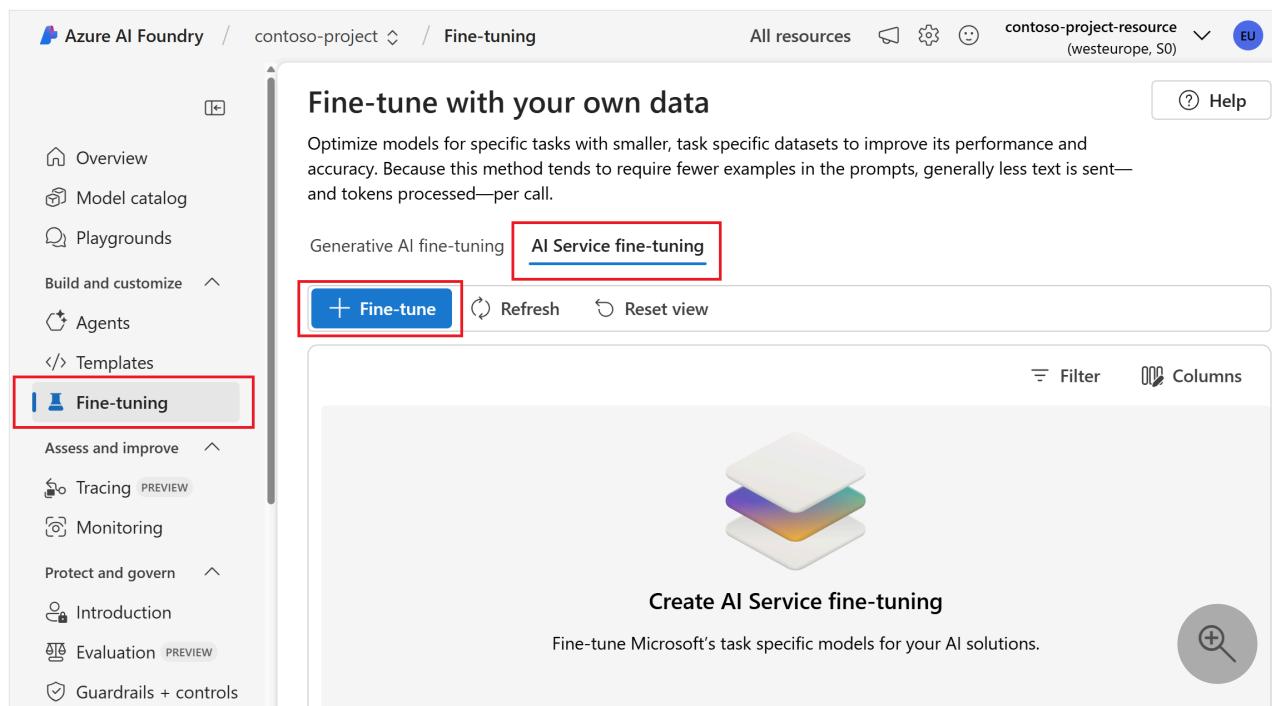
Go to the Azure AI Speech documentation to learn how to continue fine-tuning your professional voice model:

- Add voice talent consent
- Add training datasets
- Train your voice model
- Deploy your professional voice model as an endpoint

View fine-tuned models

After fine-tuning, you can access your custom voice models and deployments from the **Fine-tuning** page.

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane.
3. Select **AI Service fine-tuning**. You can view the status of your fine-tuning tasks and the models that were created.



Next steps

Add voice talent consent to the professional voice project.

Add voice talent consent to the professional voice project

05/20/2025

A voice talent is an individual or target speaker whose voices are recorded and used to create neural voice models.

Before you can fine-tune a professional voice, you must submit a recording of the voice talent's consent statement. The voice talent statement is a recording of the voice talent reading a statement that they consent to the usage of their speech data for professional voice fine-tuning. The consent statement is also used to verify that the voice talent is the same person as the speaker in the fine-tuning data.

💡 Tip

Before you get started in Azure AI Foundry portal, define your voice [persona and choose the right voice talent](#).

You can find the verbal consent statement in multiple languages on [GitHub](#). The language of the verbal statement must be the same as your recording. See also the [disclosure for voice talent](#).

Add voice talent

💡 Tip

For a sample consent statement and training data, see the [GitHub repository](#).

To add a voice talent profile and upload their consent statement, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the professional voice fine-tuning task (by model name) that you [started as described in the create professional voice article](#).
4. Select **Set up voice talent > + Add voice talent**.

- In the **Add new voice talent** wizard, select the target scenarios for the voice talent. The target scenarios must be consistent with what you provided in the application form. The scenarios are used to help identify the voice talent and to ensure that the voice model is trained for the intended use cases.
- Optionally in the **Voice characteristics** text box, enter a description of the characteristics of the voice you're going to create.
- Select **Next**.
- On the **Upload verbal statement** page, follow the instructions to upload the voice talent statement you recorded beforehand.

- Enter the voice talent name and company name. The voice talent name must be the name of the person who recorded the consent statement. Enter the name in the same language used in the recorded statement. The company name must match the company name that was spoken in the recorded statement. Ensure the company name is entered in the same language as the recorded statement.
- Make sure the verbal statement was **recorded** with the same settings, environment, and speaking style as your fine-tuning data.

Add new voice talent

Voice characteristics

Statement upload

Review and add

Upload verbal statement

Upload an audio recording of your voice talent saying the sentence below. This audio file will be used to create a voice signature of your voice talent and to verify against your training data when you create a voice model. Learn more about [Microsoft's processing, use and retention of voice talent data](#).

Voice talent name *

Sample voice actor

Company name *

Contoso

"I **Sample voice actor** am aware that recordings of my voice will be used by **Contoso** to create and use a synthetic version of my voice."

Verbal statement file *

Back Next Add voice talent Cancel

- Select **Next**.
- Review the voice talent and persona details, and select **Add voice talent**.

After the voice talent status is *Succeeded*, you can **add fine-tuning data**.

Next steps

Add training data for professional voice fine-tuning

Add a professional voice training dataset

05/20/2025

When you're ready to create a custom text to speech voice for your application, the first step is to gather audio recordings and associated scripts to start training the voice model. For details on recording voice samples, see [the tutorial](#). The Speech service uses this data to create a unique voice tuned to match the voice in the recordings. After you've trained the voice, you can start synthesizing speech in your applications.

All data you upload must meet the requirements for the data type that you choose. It's important to correctly format your data before it's uploaded, which ensures the data will be accurately processed by the Speech service. To confirm that your data is correctly formatted, see [Training data types](#).

! Note

- Standard subscription (S0) users can upload five data files simultaneously. If you reach the limit, wait until at least one of your data files finishes importing. Then try again.
- The maximum number of data files allowed to be imported per subscription is 500 .zip files for standard subscription (S0) users. Please see out [Speech service quotas and limits](#) for more details.

Upload your data

💡 Tip

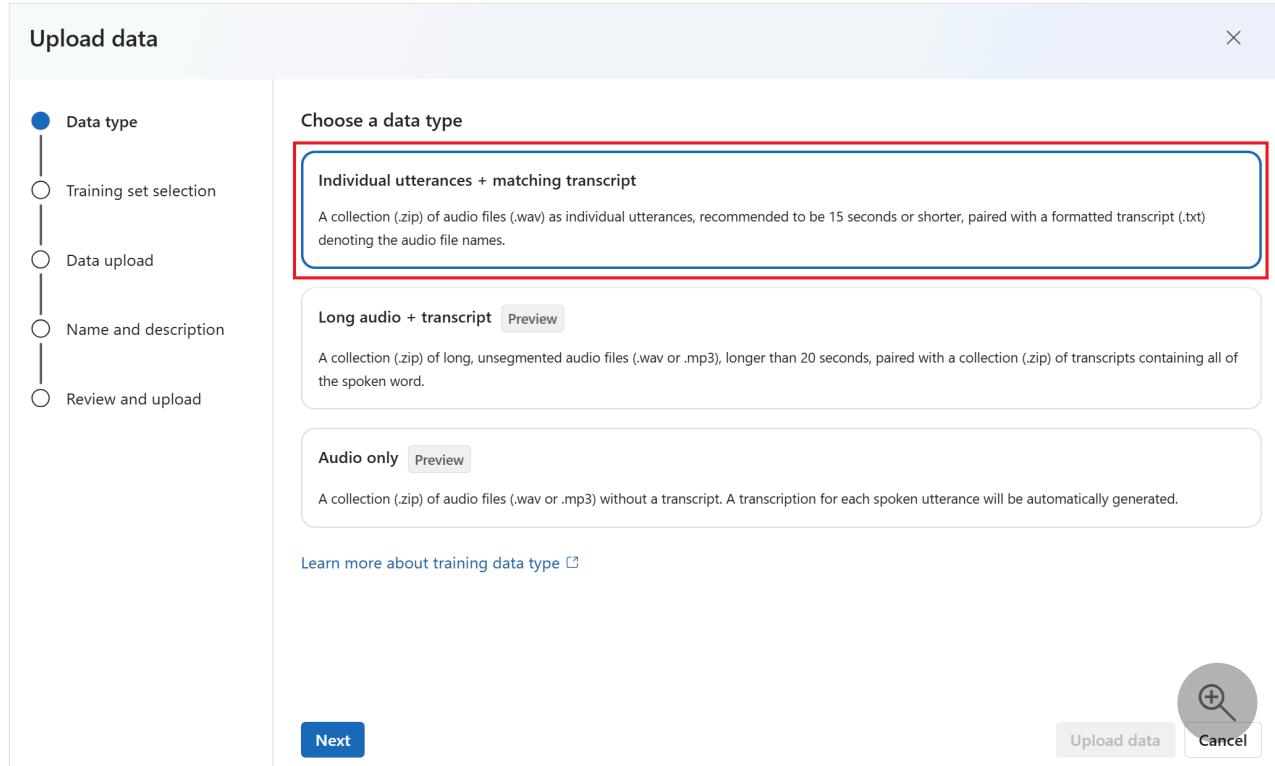
For a sample consent statement and training data, see the [GitHub repository](#).

When you're ready to upload your data, go to the **Prepare training data** tab to add your first training set and upload data. A *training set* is a set of audio utterances and their mapping scripts used for training a voice model. You can use a training set to organize your training data. The service checks data readiness per each training set. You can import multiple data to a training set.

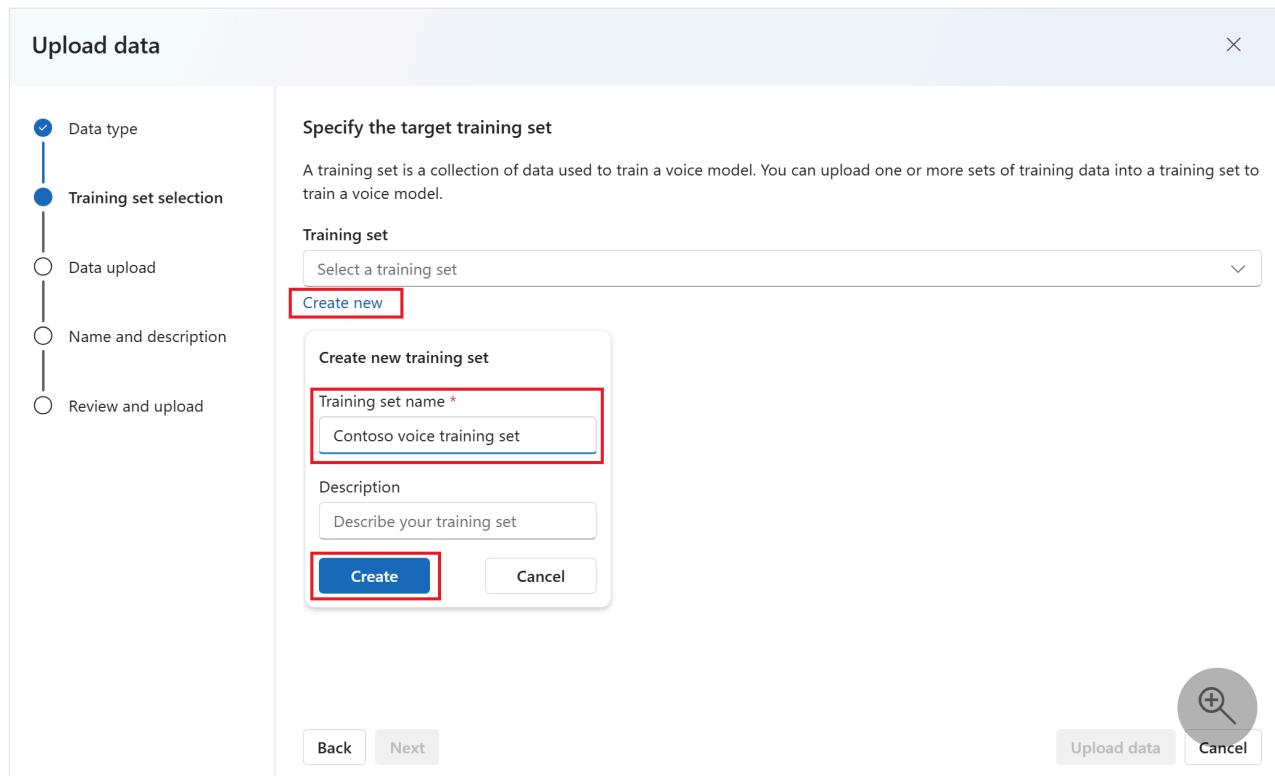
To upload training data, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).

2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the professional voice fine-tuning task (by model name) that you [started as described in the create professional voice article](#).
4. Select **Prepare training data > Upload data**.
5. In the **Upload data** wizard, choose a [data type](#). If you're using the sample data, select **Individual utterances + matching transcript**.



6. Select **Next**.
7. On the **Specify the target training set** page, select **Create new**.
8. Enter a training set name and then select **Create**.



9. Select **Next**.

10. On the **Data upload** page, select a **Recording file** and **Script file** in the respective tiles.

You can select local files from your computer or enter the Azure Blob storage URL to upload data.

11. Select **Next**.

12. Enter a name and description for your data and then select **Next**.

13. Review the upload details, and select **Upload data**.

(!) Note

Duplicate IDs aren't accepted. Utterances with the same ID will be removed.

Duplicate audio names are removed from the training. Make sure the data you select don't contain the same audio names within the .zip file or across multiple .zip files. If utterance IDs (either in audio or script files) are duplicates, they're rejected.

Data files are automatically validated when you select **Upload data**. Data validation includes a series of checks on the audio files to verify their file format, size, and sampling rate. If there are any errors, fix them and submit again.

After you upload the data, you can check the details in the training set detail view. On the detail page, you can further check the pronunciation issue and the noise level for each of your data. The pronunciation score at the sentence level ranges from 0-100. A score below 70

normally indicates a speech error or script mismatch. Utterances with an overall score lower than 70 will be rejected. A heavy accent can reduce your pronunciation score and affect the generated digital voice.

Resolve data issues online

After upload, you can check the data details of the training set. Before continuing to [train your voice model](#), you should try to resolve any data issues.

Typical data issues

The issues are divided into three types. Refer to the following tables to check the respective types of errors.

Auto-rejected

Data with these errors won't be used for training. Imported data with errors will be ignored, so you don't need to delete them. You can [fix these data errors online](#) or upload the corrected data again for training.

 [Expand table](#)

Category	Name	Description
Script	Invalid separator	You must separate the utterance ID and the script content with a Tab character.
Script	Invalid script ID	The script line ID must be numeric.
Script	Duplicated script	Each line of the script content must be unique. The line is duplicated with {}.
Script	Script too long	The script must be less than 1,000 characters.
Script	No matching audio	The ID of each utterance (each line of the script file) must match the audio ID.
Script	No valid script	No valid script is found in this dataset. Fix the script lines that appear in the detailed issue list.
Audio	No matching script	No audio files match the script ID. The name of the .wav files must match with the IDs in the script file.
Audio	Invalid audio format	The audio format of the .wav files is invalid. Check the .wav file format by using an audio tool like SoX .
Audio	Low sampling rate	The sampling rate of the .wav files can't be lower than 16 KHz.

Category	Name	Description
Audio	Too long audio	Audio duration is longer than 30 seconds. Split the long audio into multiple files. It's a good idea to make utterances shorter than 15 seconds.
Audio	No valid audio	No valid audio is found in this dataset. Check your audio data and upload again.
Mismatch	Low scored utterance	Sentence-level pronunciation score is lower than 70. Review the script and the audio content to make sure they match.

Auto-fixed

The following errors are fixed automatically, but you should review and confirm the fixes are made correctly.

[\[+\] Expand table](#)

Category	Name	Description
Mismatch	Silence auto fixed	The start silence is detected to be shorter than 100 ms, and has been extended to 100 ms automatically. Download the normalized dataset and review it.
Mismatch	Silence auto fixed	The end silence is detected to be shorter than 100 ms, and has been extended to 100 ms automatically. Download the normalized dataset and review it.
Script	Text auto normalized	Text is automatically normalized for digits, symbols, and abbreviations. Review the script and audio to make sure they match.

Manual check required

Unresolved errors listed in the next table affect the quality of training, but data with these errors won't be excluded during training. For higher-quality training, it's a good idea to fix these errors manually.

[\[+\] Expand table](#)

Category	Name	Description
Script	Non-normalized text	This script contains symbols. Normalize the symbols to match the audio. For example, normalize / to slash.
Script	Not enough question utterances	At least 10 percent of the total utterances should be question sentences. This helps the voice model properly express a questioning tone.

Category	Name	Description
Script	Not enough exclamation utterances	At least 10 percent of the total utterances should be exclamation sentences. This helps the voice model properly express an excited tone.
Script	No valid end punctuation	Add one of the following at the end of the line: full stop (half-width '.' or full-width '。'), exclamation point (half-width '!' or full-width '！'), or question mark (half-width '?' or full-width '？').
Audio	Low sampling rate for neural voice	It's recommended that the sampling rate of your .wav files should be 24 KHz or higher for creating neural voices. If it's lower, it will be automatically raised to 24 KHz.
Volume	Overall volume too low	Volume shouldn't be lower than -18 dB (10 percent of max volume). Control the volume average level within proper range during the sample recording or data preparation.
Volume	Volume overflow	Overflowing volume is detected at {}s. Adjust the recording equipment to avoid the volume overflow at its peak value.
Volume	Start silence issue	The first 100 ms of silence isn't clean. Reduce the recording noise floor level, and leave the first 100 ms at the start silent.
Volume	End silence issue	The last 100 ms of silence isn't clean. Reduce the recording noise floor level, and leave the last 100 ms at the end silent.
Mismatch	Low scored words	Review the script and the audio content to make sure they match, and control the noise floor level. Reduce the length of long silence, or split the audio into multiple utterances if it's too long.
Mismatch	Start silence issue	Extra audio was heard before the first word. Review the script and the audio content to make sure they match, control the noise floor level, and make the first 100 ms silent.
Mismatch	End silence issue	Extra audio was heard after the last word. Review the script and the audio content to make sure they match, control the noise floor level, and make the last 100 ms silent.
Mismatch	Low signal-noise ratio	Audio SNR level is lower than 20 dB. At least 35 dB is recommended.

Category	Name	Description
Mismatch	No score available	Failed to recognize speech content in this audio. Check the audio and the script content to make sure the audio is valid, and matches the script.

Next steps

[Train the professional voice](#)

Train your professional voice model

05/20/2025

In this article, you learn how to fine-tune a professional voice through the Azure AI Foundry portal.

ⓘ Important

Professional voice fine-tuning is currently only available in some regions. After your voice model is trained in a supported region, you can [copy the professional voice model](#) to an Azure AI Foundry resource in another region as needed. For more information, see the footnotes in the [Speech service table](#).

Training duration varies depending on how much data you use. It takes about 40 compute hours on average to fine-tune a professional voice. With an Azure AI Foundry standard (S0) resource, you can train four voices simultaneously. If you reach the limit, wait until at least one of your voice models finishes training, and then try again.

ⓘ Note

Although the total number of hours required per [training method](#) varies, the same unit price applies to each. For more information, see the [custom neural training pricing details ↗](#).

Choose a training method

After you validate your data files, use them to build your custom voice model. When you create a custom voice, you can choose to train it with one of the following methods:

- [Neural](#): Create a voice in the same language of your training data.
- [Neural - cross lingual](#): Create a voice that speaks a different language from your training data. For example, with the `zh-CN` training data, you can create a voice that speaks `en-US`.

The language of the training data and the target language must both be one of the [languages that are supported](#) for cross lingual voice training. You don't need to prepare training data in the target language, but your test script must be in the target language.

- [Neural - multi style](#): Create a custom voice that speaks in multiple styles and emotions, without adding new training data. Multiple style voices are useful for video game

characters, conversational chatbots, audiobooks, content readers, and more.

To create a multiple style voice, you need to prepare a set of general training data, at least 300 utterances. Select one or more of the preset target speaking styles. You can also create multiple custom styles by providing style samples, of at least 100 utterances per style, as extra training data for the same voice. The supported preset styles vary according to different languages. See [available preset styles across different languages](#).

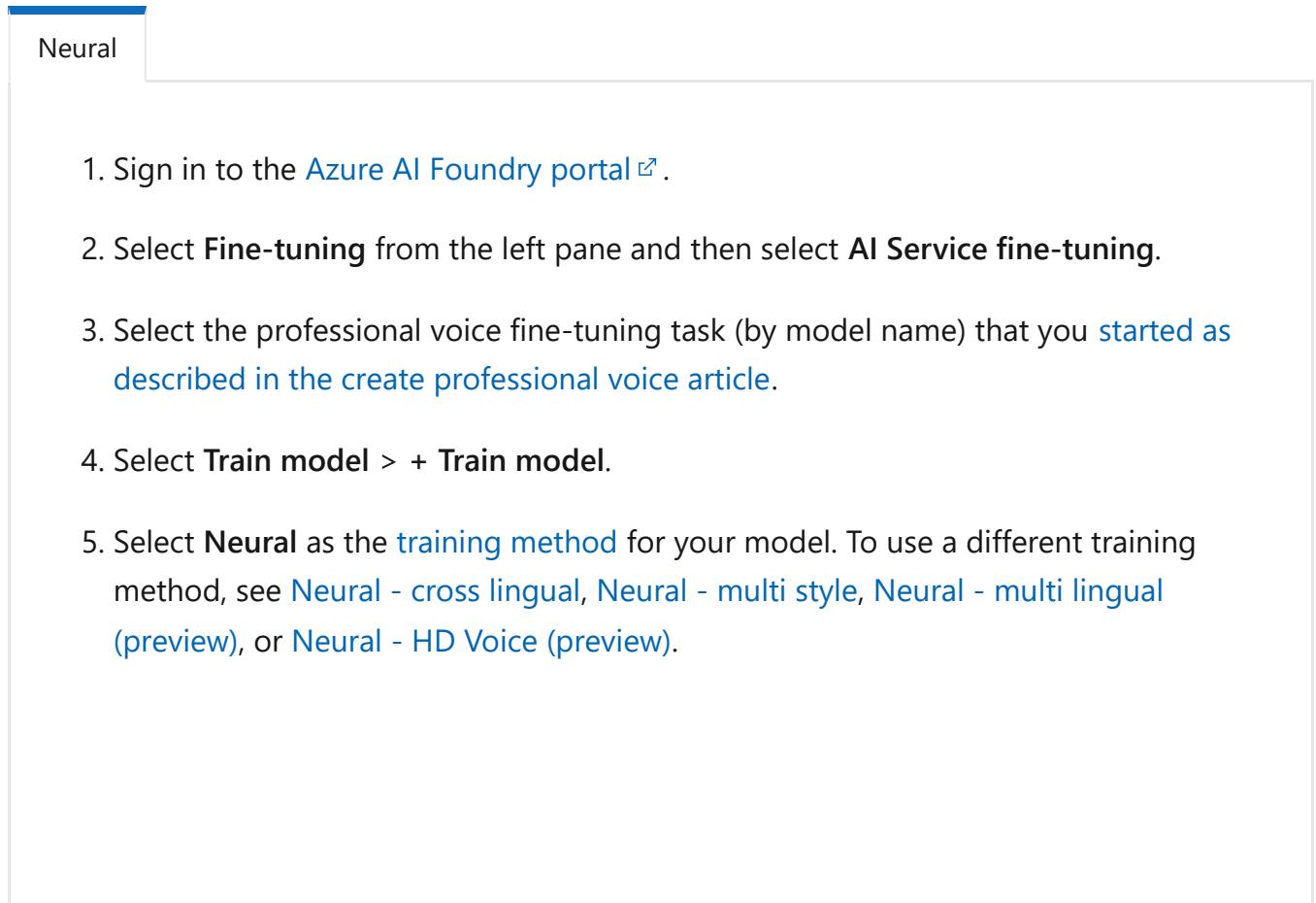
- [Neural - multi lingual \(preview\)](#): Create a voice that speaks multiple languages using the single-language training data. For example, with the `en-US` primary training data, you can create a voice that speaks `en-US`, `de-DE`, `zh-CN` etc. secondary languages.

The primary language of the training data and the secondary languages must be in the [languages that are supported](#) for multi lingual voice training. You don't need to prepare training data in the secondary languages.

The language of the training data must be one of the [languages that are supported](#) for custom voice, cross-lingual, or multiple style training.

Train your custom voice model

To create a custom voice in Azure AI Foundry portal, follow these steps for one of the following methods:



The screenshot shows a user interface for training a custom voice model. At the top, there is a horizontal navigation bar with several tabs. The 'Neural' tab is highlighted with a blue background and white text. Below the navigation bar, there is a large, light-gray rectangular area containing a numbered list of steps. The steps are as follows:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the professional voice fine-tuning task (by model name) that you [started as described in the create professional voice article](#).
4. Select **Train model > + Train model**.
5. Select **Neural** as the **training method** for your model. To use a different training method, see [Neural - cross lingual](#), [Neural - multi style](#), [Neural - multi lingual \(preview\)](#), or [Neural - HD Voice \(preview\)](#).

Train a new model

Select the training method for your model

Training method

- Data and voice talent
- Test script
- Basic information
- Review and train

Neural - Default
Create a voice in the same language as your training data.

Neural - Cross lingual
Create a voice that speaks a different language from your training data.

Neural - Multi lingual Preview
Create a voice that speaks multiple languages using training data from a single language.

Neural - Multi style
Create a voice that speaks in multiple styles.

Version * ⓘ
V10.1

Version description: Model updated to understand context and automatically adjust speaking tones with contextual training data. 15 compute hours estimated for each training. Start with 300 utterances or 30 minutes of speech data to

Next Train Cancel

6. Select a version of the training recipe for your model. The latest version is selected by default. The supported features and training time can vary by version. Normally, we recommend the latest version. In some cases, you can choose an earlier version to reduce training time. See [Bilingual training](#) for more information about bilingual training and differences between locales.

7. Select **Next**.

8. Select the data that you want to use for training. Duplicate audio names are removed from the training. Make sure that the data you select doesn't contain the same audio names across multiple .zip files.

You can select only successfully processed datasets for training. If you don't see your training set in the list, check your data processing status.

9. Select a speaker file with the voice talent statement that corresponds to the speaker in your training data.

10. Select **Next**.

11. Select a test script and then select **Next**.

- Each training generates 100 sample audio files automatically to help you test the model with a default script.
- Alternatively, you can select **Add my own test script** and provide your own test script with up to 100 utterances to test the model at no extra cost. The generated audio files are a combination of the automatic test scripts and custom test scripts. For more information, see [test script requirements](#).

12. Enter a **Voice model name**. Choose a name carefully. The model name is used as the voice name in your [speech synthesis request](#) by the SDK and SSML input. Only letters, numbers, and a few punctuation characters are allowed. Use different names for different neural voice models.
13. Optionally, enter the **Description** to help you identify the model. A common use of the description is to record the names of the data that you used to create the model.
14. Select the checkbox to accept the terms of use and then select **Next**.
15. Review the settings and select the box to accept the terms of use.
16. Select **Train** to start training the model.

Bilingual training

If you select the **Neural** training type, you can train a voice to speak in multiple languages. The `zh-CN`, `zh-HK`, and `zh-TW` locales support bilingual training for the voice to speak both Chinese and English. Depending in part on your training data, the synthesized voice can speak English with an English native accent or English with the same accent as the training data.

ⓘ Note

To enable a voice in the `zh-CN` locale to speak English with the same accent as the sample data, you should upload English data to a **Contextual** training set, or choose `Chinese (Mandarin, Simplified)`, `English bilingual` when creating a project or specify the `zh-CN (English bilingual)` locale for the training set data via REST API.

In your contextual training set, include at least 100 sentences or 10 minutes of English content and do not exceed the amount of Chinese content.

The following table shows the differences among the locales:

[] [Expand table](#)

Speech Studio locale	REST API locale	Bilingual support
<code>Chinese (Mandarin, Simplified)</code>	<code>zh-CN</code>	If your sample data includes English, the synthesized voice speaks English with an English native accent, instead of the same accent as the sample data, regardless of the amount of English data.

Speech Studio locale	REST API locale	Bilingual support
Chinese (Mandarin, Simplified), English bilingual	zh-CN (English bilingual)	If you want the synthesized voice to speak English with the same accent as the sample data, we recommend including over 10% English data in your training set. Otherwise, the English speaking accent might not be ideal.
Chinese (Cantonese, Simplified)	zh-HK	If you want to train a synthesized voice capable of speaking English with the same accent as your sample data, make sure to provide over 10% English data in your training set. Otherwise, it defaults to an English native accent. The 10% threshold is calculated based on the data accepted after successful uploading, not the data before uploading. If some uploaded English data is rejected due to defects and doesn't meet the 10% threshold, the synthesized voice defaults to an English native accent.
Chinese (Taiwanese Mandarin, Traditional)	zh-TW	If you want to train a synthesized voice capable of speaking English with the same accent as your sample data, make sure to provide over 10% English data in your training set. Otherwise, it defaults to an English native accent. The 10% threshold is calculated based on the data accepted after successful uploading, not the data before uploading. If some uploaded English data is rejected due to defects and doesn't meet the 10% threshold, the synthesized voice defaults to an English native accent.

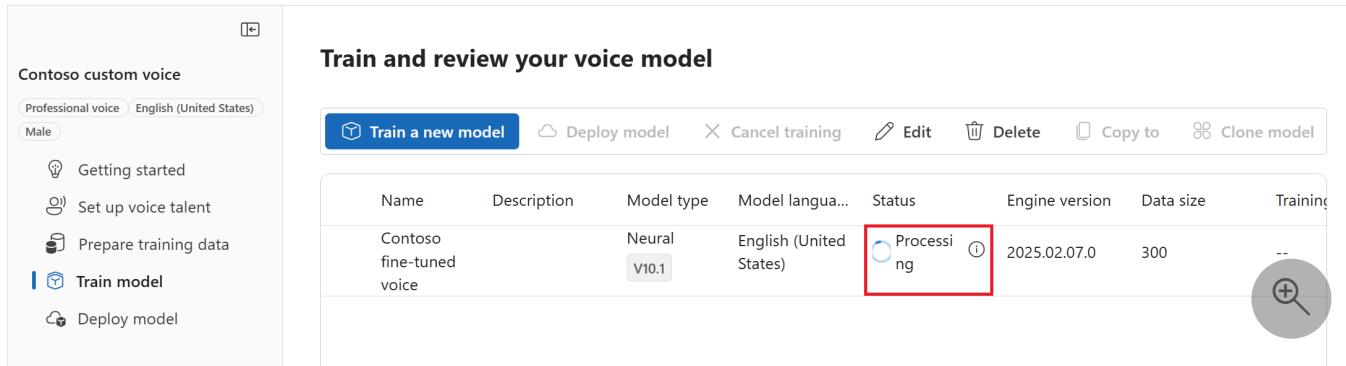
Monitor the training process

The **Train model** table displays a new entry that corresponds to this newly created model. The status reflects the process of converting your data to a voice model, as described in this table:

 Expand table

State	Meaning
Processing	Your voice model is being created.
Succeeded	Your voice model has been created and can be deployed.
Failed	Your voice model has failed in training. The cause of the failure might be, for example, unseen data problems or network issues.
Canceled	The training for your voice model was canceled.

While the model status is **Processing**, you can select the model and then select **Cancel training** to cancel training. You're not charged for this canceled training.



The screenshot shows a user interface for managing voice models. On the left, there's a sidebar with options like 'Getting started', 'Set up voice talent', 'Prepare training data', 'Train model' (which is selected and highlighted in blue), and 'Deploy model'. The main area is titled 'Train and review your voice model' and contains a table of models. The table has columns for Name, Description, Model type, Model langua..., Status, Engine version, Data size, and Training. A single row is visible, representing a 'Contoso fine-tuned voice' model. The 'Status' column for this model is highlighted with a red box and shows the status 'Processing'. There are also other status indicators like a question mark icon next to the status text.

After you finish training the model successfully, you can review the model details and [Test your voice model](#).

Rename your model

You have to clone your model to rename it. You can't rename the model directly.

1. Select the model.
2. Select **Clone model** to create a clone of the model with a new name in the current project.
3. Enter the new name on the **Clone voice model** window.
4. Select **Submit**. The text *Neural* is automatically added as a suffix to your new model name.

Test your voice model

After your voice model is successfully built, you can use the generated sample audio files to test it before you deploy it.

! Note

[Neural - multi lingual \(preview\)](#) and [Neural - HD Voice \(preview\)](#) do not support this type of testing.

The quality of the voice depends on many factors, such as:

- The size of the training data.
- The quality of the recording.
- The accuracy of the transcript file.

- How well the recorded voice in the training data matches the personality of the designed voice for your intended use case.

Select **DefaultTests** under **Testing** to listen to the sample audio files. The default test samples include 100 sample audio files generated automatically during training to help you test the model. In addition to these 100 audio files provided by default, your own test script utterances are also added to **DefaultTests** set. This addition is at most 100 utterances. You're not charged for the testing with **DefaultTests**.

If you want to upload your own test scripts to further test your model, select **Add test scripts** to upload your own test script.

Before you upload test script, check the [Test script requirements](#). You're charged for the extra testing with the batch synthesis based on the number of billable characters. See [Azure AI Speech pricing](#).

Under **Add test scripts**, select **Browse for a file** to select your own script, then select **Add** to upload it.

Test script requirements

The test script must be a `.txt` file that is less than 1 MB. Supported encoding formats include ANSI/ASCII, UTF-8, UTF-8-BOM, UTF-16-LE, or UTF-16-BE.

Unlike the [training transcription files](#), the test script should exclude the utterance ID, which is the filename of each utterance. Otherwise, these IDs are spoken.

Here's an example set of utterances in one `.txt` file:

text

```
This is the waistline, and it's falling.  
We have trouble scoring.  
It was Janet Maslin.
```

Each paragraph of the utterance results in a separate audio. If you want to combine all sentences into one audio, make them a single paragraph.

(!) Note

The generated audio files are a combination of the automatic test scripts and custom test scripts.

Update engine version for your voice model

Azure text to speech engines are updated from time to time to capture the latest language model that defines the pronunciation of the language. After you train your voice, you can apply your voice to the new language model by updating to the latest engine version.

- When a new engine is available, you're prompted to update your neural voice model.
- Go to the model details page and follow the on-screen instructions to install the latest engine.
- Alternatively, select **Install the latest engine** later to update your model to the latest engine version. You're not charged for engine update. The previous versions are still kept.
- You can check all engine versions for the model from the **Engine version** list, or remove one if you don't need it anymore.

The updated version is automatically set as default. But you can change the default version by selecting a version from the drop-down list and selecting **Set as default**.

If you want to test each engine version of your voice model, you can select a version from the list, then select **DefaultTests** under **Testing** to listen to the sample audio files. If you want to upload your own test scripts to further test your current engine version, first make sure the version is set as default, then follow the steps in [Test your voice model](#).

Updating the engine creates a new version of the model at no extra cost. After you update the engine version for your voice model, you need to deploy the new version to [create a new endpoint](#). You can only deploy the default version.

After you create a new endpoint, you need to [transfer the traffic to the new endpoint in your product](#).

To learn more about the capabilities and limits of this feature, and the best practice to improve your model quality, see [Characteristics and limitations for using custom voice](#).

Copy your voice model to another project

Note

In this context "project" refers to a fine-tuning task rather than an Azure AI Foundry project.

After training you can copy your voice model to another project for the same region or another region.

For example, you can copy a professional voice model that was trained in one region, to a project for another region. Professional voice fine-tuning is currently only [available in some regions](#).

To copy your custom voice model to another project:

1. On the **Train model** tab, select a voice model that you want to copy, and then select **Copy to project**.
2. Select the **Subscription**, **Target region**, **Connected AI Service resource** (AI Foundry resource), and **Target fine-tuning task** where you want to copy the model.
3. Select **Copy to** to copy the model.
4. Select **View model** under the notification message for the successful copying.

Navigate to the project where you copied the model to [deploy the model copy](#).

Next steps

[Deploy the professional voice endpoint](#)

Deploy your professional voice model as an endpoint

05/20/2025

After you successfully created and [fine-tuned](#) your professional voice model, you deploy it to a custom voice endpoint.

⚠ Note

You can create up to 50 endpoints with a standard (S0) Speech resource, each with its own custom voice.

To use your fine-tuned professional voice, you must specify the voice model name, use the custom URI directly in an HTTP request, and use the same Speech resource to pass through the authentication of the text to speech service.

Add a deployment endpoint

To create a professional voice endpoint:

To deploy an endpoint, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the professional voice fine-tuning task (by model name) that you [started as described in the create professional voice article](#).
4. Select **Deploy model > Deploy model**.
5. Select a voice model that you want to associate with this endpoint and then select **Next**.
6. Enter a **Endpoint name** and **Description** for your custom endpoint.
7. Select **Endpoint type** according to your scenario. If your resource is in a supported region, the default setting for the endpoint type is *High performance*. Otherwise, if the resource is in an unsupported region, the only available option is *Fast resume*.
 - **High performance:** Optimized for scenarios with real-time and high-volume synthesis requests, such as conversational AI, call-center bots. It takes around 5 minutes to deploy or resume an endpoint. For information about regions where the *High performance* endpoint type is supported, see the footnotes in the [regions](#) table.
 - **Fast-resume:** Optimized for audio content creation scenarios with less frequent synthesis requests. Easy and quick to deploy or resume an endpoint in under a

minute. The fast-resume endpoint type is supported in all [regions](#) where text to speech is available.

8. Select **Next**.
9. Select the checkbox to accept the terms of use and then select **Next**.
10. Review the settings and select the box to accept the model hosting costs.
11. Select **Deploy** to create your endpoint.

After your endpoint is deployed, the endpoint name appears as a link. Select the link to display information specific to your endpoint, such as the endpoint key, endpoint URL, and sample code. When the status of the deployment is **Succeeded**, the endpoint is ready for use.

Use your custom voice

The custom endpoint is functionally identical to the standard endpoint that's used for text to speech requests.

One difference is that the `EndpointId` must be specified to use the custom voice via the Speech SDK. You can start with the [text to speech quickstart](#) and then update the code with the `EndpointId` and `SpeechSynthesisVoiceName`. For more information, see [use a custom endpoint](#).

To use a custom voice via [Speech Synthesis Markup Language \(SSML\)](#), specify the model name as the voice name. This example uses the `YourCustomVoiceName` voice.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice name="YourCustomVoiceName">
    This is the text that is spoken.
  </voice>
</speak>
```

Switch to a new voice model in your product

Once you updated your voice model to the latest engine version, or if you want to switch to a new voice in your product, you need to redeploy the new voice model to a new endpoint. Redeploying new voice model on your existing endpoint isn't supported. After deployment, switch the traffic to the newly created endpoint. We recommend that you transfer the traffic to the new endpoint in a test environment first to ensure that the traffic works well, and then transfer to the new endpoint in the production environment. During the transition, you need to keep the old endpoint. If there are some problems with the new endpoint during transition, you can switch back to your old endpoint. If the traffic has been running well on the new endpoint for about 24 hours (recommended value), you can delete your old endpoint.

! Note

If your voice name is changed and you're using Speech Synthesis Markup Language (SSML), be sure to use the new voice name in SSML.

Suspend and resume an endpoint

You can suspend or resume an endpoint to limit spend and conserve resources that aren't in use. You aren't charged while the endpoint is suspended. When you resume an endpoint, you can continue to use the same endpoint URL in your application to synthesize speech.

! Note

The suspend operation completes almost immediately. The resume operation completes in about the same amount of time as a new deployment.

This section describes how to suspend or resume a custom voice endpoint in the Azure AI Foundry portal.

Suspend endpoint

To suspend and deactivate your endpoint:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the professional voice fine-tuning task (by model name) that you [started as described in the create professional voice article](#).
4. Select **Deploy model**.
5. Select the endpoint you want to suspend and then select **Suspend**.

Name	Description	Voice name	Model...	Model langua...	Created on	Status
Contoso fine-tuned voice	Contoso fine-tuned voice	Neural v10.1	English (United States)	5/15/2025, 7:15 AM	(green circle with checkmark)	(grey circle with magnifying glass)

6. In the dialog box that appears, select **Suspend**. After the endpoint is suspended, the status changes from **Succeeded** to **Suspended**.

Resume endpoint

1. To resume and activate your endpoint, select **Resume** from the **Deploy model** tab in the [Azure AI Foundry portal](#).
2. In the dialog box that appears, select **Submit**. After you successfully reactivate the endpoint, the status will change from **Suspended** to **Succeeded**.

Next steps

- Learn more about custom voice in the [overview](#).
- Learn more about custom avatar in the [overview](#).

Custom voice lite

06/02/2025

Custom voice lite is a project type where you can demo and evaluate custom voice before investing in professional recordings to create a higher-quality voice. No application is required for demo and evaluation purposes. However, Microsoft restricts and selects the recording and testing samples for use with custom voice lite. You must apply for full access to professional voice fine-tuning in order to deploy and use the custom voice lite model for business purpose. In that case, request access on the [intake form](#).

! Note

Custom voice lite is only available in the [Speech Studio](#). It isn't available through the Azure AI Foundry portal, REST API, or SDKs.

With a custom voice lite project, you record your voice online by reading 20-50 pre-defined scripts provided by Microsoft. After you've recorded at least 20 samples, you can start to train a model. Once the model is trained successfully, you can review the model and check out 20 output samples produced with another set of pre-defined scripts.

See the [supported languages](#) for custom voice.

Compare project types

The following table summarizes key differences between custom voice lite and professional voice fine-tuning.

[] [Expand table](#)

Items	Lite	Professional
Target scenarios	Demonstration or evaluation	Professional scenarios like brand and character voices for chat bots, or audio content reading.
Training data	Record online using Speech Studio	Bring your own data. Recording in a professional studio is recommended.
Scripts for recording	Provided in Speech Studio	Use your own scripts that match the use case scenario. Microsoft provides example scripts for reference.

Items	Lite	Professional
Required data size	20-50 utterances	300-2000 utterances
Training time	Less than one compute hour	Approximately 20-40 compute hours
Voice quality	Moderate quality	High quality
Availability	Anyone can record samples online and train a model for demo and evaluation purpose. Full access to custom voice is required if you want to deploy the custom voice lite model for business use.	Data upload isn't restricted, but you can only fine-tune a professional voice after access is approved. Professional voice fine-tuning is limited based on eligibility and usage criteria. Request access on the intake form .
Pricing	Per unit prices apply equally for both custom voice lite and professional voice fine-tuning. Check the pricing details here .	Per unit prices apply equally for both custom voice lite and professional voice fine-tuning. Check the pricing details here .

Create a custom voice lite project

To create a custom voice lite project, follow these steps:

1. Sign in to the [Speech Studio](#).
2. Select the subscription and Speech resource to work with.
3. Select **Custom voice > Create a project**.
4. Select **Custom neural voice lite > Next**. To create a custom voice professional project instead, see the [professional voice fine-tuning documentation](#).
5. Follow the instructions provided by the wizard to create your project.

(i) **Important**

The custom voice lite project expires after 90 days unless the [verbal statement](#) recorded by the voice talent is submitted.

6. Select the new project by name or select **Go to project**. You see these menu items in the left panel: **Record and build**, **Review model**, and **Deploy model**.

The screenshot shows the Microsoft Custom Voice interface. At the top, a banner says "This project expires in 90 days without verbal statement from the voice talent." and "Manage your voice talent". The left sidebar shows "Custom Voice", "My CNV Lite Project Lite (English (United States))", and options for "Record and build", "Review model", and "Deploy model". The main content area is titled "Record and build" with the sub-section "Record and build". It contains instructions: "Speak clearly with a good microphone and no background noise. To create a synthetic voice using Custom Neural Voice Lite, record at least 20 voice samples using the provided scripts. Voice samples recorded here will be used to create a synthetic version of your voice." Below this is a section titled "Custom Neural Voice Lite" with four steps: 1. Record (microphone icon), 2. Train (brain icon), 3. Test (flask icon), and 4. Deploy (up arrow icon). Each step has a brief description and a "Get started" button. Step 4 includes a note: "With the full access approved and the voice talent consent verified, deploy and use the voice in your experience." A link "Apply for full access" is also present.

Record and build a custom voice lite model

Record at least 20 voice samples (up to 50) with provided scripts online. Voice samples recorded here are used to create a synthetic version of your voice.

! Note

Custom voice training is currently only available in some regions. See footnotes in the [regions](#) table for more information.

Here are some tips to help you record your voice samples:

- Use a good microphone. Increase the clarity of your samples by using a high-quality microphone. Speak about 8 inches away from the microphone to avoid mouth noises.
- Avoid background noise. Record in a quiet room without background noise or echoing.
- Relax and speak naturally. Allow yourself to express emotions as you read the sentences.
- Record in one take. To keep a consistent energy level, record all sentences in one session.
- Pronounce each word correctly, and speak clearly.

To record and build a custom voice lite model, follow these steps in [Speech Studio](#):

1. Select **Custom voice** > Your project name > **Record and build**.
2. Select **Get started**.
3. Read the Voice talent terms of use carefully. Select the checkbox to acknowledge the terms of use.
4. Select **Accept**.
5. Press the microphone icon to start the noise check. This noise check takes only a few seconds, and you don't need to speak during it.
6. If noise was detected, you can select **Check again** to repeat the noise check. If no noise was detected, you can select **Done** to proceed to the next step.

Noise check

X

Make sure you're in a quiet room without background noise or echoing. This noise check will take only a few seconds, and you won't need to speak during it.



Check again

⚠️ Noise detected

For the best results, we recommend moving to a quieter area without background noise before recording your voice samples.

Done

Cancel

7. Review the recording tips and select **Got it**. For the best results, go to a quiet area without background noise before recording your voice samples.
8. Press the microphone icon to start recording.

This project expires in 90 days without verbal statement from the voice talent. Manage your voice talent

Speech Studio > Custom Voice > Record and build

Record and build

Speak clearly with a good microphone and no background noise. To create a synthetic voice using Custom Neural Voice Lite, record at least 20 voice samples using the provided scripts. Voice samples recorded here will be used to create a synthetic version of your voice.

Voice talent: Eric Urban

Recording tips | Check noise level

When you're ready to read the following sentence, select Record.

Our differences are what make us beautiful and unique.

Press to start recording or use Enter or Space bar

00:00 00:00

Clearness Pronunciation Volume Omissions Mispronunciations Insertions

20 qualified samples to train or record up to 50

0 qualified/0 recorded

Record next

Recorded samples

A small illustration of a box overflowing with blue speech bubbles.

9. Press the stop icon to stop recording.
10. Review quality metrics. After recording each sample, check its quality metric before continuing to the next one.
11. Record more samples. Although you can create a model with just 20 samples, it's recommended that you record up to 50 to get better quality.
12. Select **Train model** to start the training process.

The training process takes approximately one compute hour. You can check the progress of the training process in the [Review model](#) page.

Review model

To review the custom voice lite model and listen to your own synthetic voice, follow these steps:

1. Select **Custom voice** > Your project name > **Review model**. Here you can review the voice model name, model language, sample data size, and training progress. The voice name is composed of the word "Neural" appended to your project name.
2. Select the voice model name to review the model details and listen to the sample text to speech results.
3. Select the play icon to hear your voice speak each script.

The screenshot shows the 'Speech Studio' interface for a 'Custom Voice' project named 'My CNV Lite Project Neural'. The 'Review model' tab is selected in the sidebar. The main area displays the model details: Gender (Male), Model language (English (United States)), Model type (Neural), Utterances (20), Sampling rate (24000), Status (Succeeded), Model ID (ae6db871-18cd-42ab-856f-90282d9c5fb3), and Created (10/15/2022 8:18 AM). Below this, the 'Sample output' section shows three text inputs with corresponding audio playback icons. The first input is a success message: 'Congratulations! You have successfully created your synthetic voice with Custom Neural Voice Lite!'. The second and third inputs are placeholder text: 'To deploy your voice model and use it in your business applications, you must get the full access to Custom Neural Voice and the explicit consent from your voice talent.' and 'With the full access, you can create an even more natural voice by bringing your own data recorded in professional studios, using Custom Neural Voice Pro.' respectively.

Submit verbal statement

A verbal statement recorded by the voice talent is required before you can [deploy the model](#) for your business use.

To submit the voice talent verbal statement, follow these steps in [Speech Studio](#):

1. Select **Custom voice** > Your project name > **Deploy model** > **Manage your voice talent**.

This project expires in 90 days without full access approval ([Learn more](#)) to Custom Neural Voice. [Apply for access](#) [X](#)

Speech Studio > Custom Voice > Voice talent statement

Deploy your model Manage your voice talent

Before you deploy the model, make sure you get meaningful consent from the voice talent to use the voice in your apps for the specific scenario defined. Learn more about [Microsoft's processing, use and retention of voice talent data](#).

Model * [Select model](#) Voice talent name * [\[State your first and last name\]](#) Company name * [\[State the name of the company\]](#)

When you're ready to read the following sentence, select Record.

I, [your first and last name], understand that the synthetic voice designated to sound like my voice created with Microsoft Speech technology will be used by [the name of the company] for its own business purposes.

Press to start recording or use Enter or Space bar

00:00 00:00s

Clearness Pronunciation Volume Omissions Mispronunciations Insertions

Submit

2. Select the model.

3. Enter the voice talent name and company name.
4. Read and record the statement. Select the microphone icon to start recording. Select the stop icon to stop recording.
5. Select **Submit** to submit the statement.
6. Check the processing status in the script table at the bottom of the dashboard. Once the status is **Succeeded**, you can [deploy the model](#).

Deploy model

To deploy your custom voice lite model and use it in your applications, you must get the full access to custom voice. Request access on the [intake form](#). Within approximately 10 business days, you receive an email with the approval status. A [verbal statement](#) recorded by the voice talent is also required before you can deploy the model for your business use.

To deploy a custom voice lite model, follow these steps in [Speech Studio](#):

1. Select **Custom voice** > Your project name > **Deploy model** > **Deploy model**.
2. Select a voice model name and then select **Next**.
3. Enter a name and description for your endpoint and then select **Next**.
4. Select the checkbox to agree to the terms of use and then select **Next**.
5. Select **Deploy** to deploy the model.

From here, you can use the custom voice lite model similarly as you would use a professional voice model. For example, you can [suspend or resume](#) an endpoint after it's created, to limit spend and conserve resources that aren't in use. You can also access the voice in the [Audio Content Creation](#) tool in the [Speech Studio](#).

Next steps

- Fine-tune a professional voice
- Try the text to speech quickstart
- Learn more about speech synthesis

What is personal voice for text to speech?

With personal voice, you can enable your users to get AI generated replication of their own voices in a few seconds. With a verbal statement and a short speech sample as the audio prompt, you can create a personal voice for your users and allow them to generate speech in any of the more than 90 languages supported across more than 100 locales.

(!) Note

For the current list of regions that support personal voice, see the [Speech service regions table](#). For supported locales, see [personal voice language support](#).

The following table summarizes the difference between personal voice and professional voice.

 Expand table

Comparison	Personal voice	Professional voice
Target scenarios	Business customers to build an app to allow their users to create and use their own personal voice in the app.	Professional scenarios like brand and character voices for chat bots, or audio content reading.
Use cases	Restricted to limited use cases. See the transparency note .	
Training data	Make sure you follow the code of conduct.	Bring your own data. Recording in a professional studio is recommended.
Required data size	One minute of human speech.	300-2000 utterances (about 30 minutes to 3 hours of human speech).
Training time	Less than 5 seconds	Approximately 20-40 compute hours.
Voice quality	Natural	Highly natural
Multilingual support	Yes. The voice is able to speak about 100 languages, with automatic language detection enabled.	Yes. You need to select the "Neural – cross lingual" feature to train a model that speaks a different language from the training data.
Availability	The demo on Speech Studio is available upon registration. Access to the API is restricted to eligible customers and approved use cases. Request access through the intake form.	You can only use professional voice fine-tuning after access is approved. Professional voice fine-tuning access is limited based on eligibility and usage criteria. Request access through the intake form.
Pricing	Check the pricing details here ¹ .	Check the pricing details here .

Comparison	Personal voice	Professional voice
Responsible AI requirements	Speaker's verbal statement required. No unapproved use case allowed.	Speaker's verbal statement required. No unapproved use case allowed.

¹ Note that personal voice pricing will only be visible for service regions where the feature is available. For the current list of supported regions, see the [Speech service regions table](#).

Try the demo

If you have an S0 resource, you can access the personal voice demo in Speech Studio. To use the personal voice API, you can apply for access [here](#).

1. Go to [Speech Studio](#)
2. Select the Personal Voice card.
3. You can record your own voice and try the voice output samples in different languages. The demo includes a subset of the languages supported by personal voice.

The screenshot shows the Microsoft Azure Speech Studio interface. At the top, there's a header with a bell icon, settings, and help. Below it, a section titled "Try your own voice" with a sub-instruction: "Record your voice and create an AI replica in seconds. With each standard Speech resource, you can try up to five voices for free. If you want to use this service for your business, please apply for access through [this link](#)". There's a checkbox for acknowledging resource usage. The main area has two tabs: "Voice list" (containing a "New voice" button and a list item for "Jessa Smith_20231110_161816") and "Detail" (showing details for the selected voice: Jessa Smith, Verbal statement, and Source language English (United States)). Below these are sections for generating audios using sample scripts and a list of generated audio samples with play and download icons.

How to create a personal voice

To get started, here's a summary of the steps to create a personal voice:

1. [Create a project.](#)
2. [Upload consent file](#). With the personal voice feature, it's required that every voice be created with explicit consent from the user. A recorded statement from the user is required acknowledging that the customer (Azure AI Speech resource owner) will create and use their voice.
3. [Get a speaker profile ID](#) for the personal voice. You get a speaker profile ID based on the speaker's verbal consent statement and an audio prompt. The user's voice characteristics are encoded in the `speakerProfileId` property that's used for text to speech.

Once you have a personal voice, you can [use it](#) to synthesize speech in any of the 91 languages supported across 100+ locales. A locale tag isn't required. Personal voice uses automatic language detection at the sentence level. For more information, see [use personal voice in your application](#).

 **Tip**

Check out the code samples in the [Speech SDK repository on GitHub](#) ↗ to see how to use personal voice in your application.

Reference documentation

[Custom voice REST API reference documentation](#)

Responsible AI

We care about the people who use AI and the people who will be affected by it as much as we care about technology. For more information, see the Responsible AI [transparency notes](#).

Next steps

- [Create a project.](#)
- Learn more about custom voice in the [overview](#).
- Learn more about Speech Studio in the [overview](#).

Create a project for personal voice

08/07/2025

Personal voice projects contain the user consent statement and the personal voice ID. You can only create a personal voice project using the custom voice API. You can't create a personal voice project in the Speech Studio.

Create a project

To create a personal voice project, use the [Projects_Create](#) operation of the custom voice API. Construct the request body according to the following instructions:

- Set the required `kind` property to `PersonalVoice`. The kind can't be changed later.
- Optionally, set the `description` property for the project description. The project description can be changed later.

Make an HTTP PUT request using the URI as shown in the following [Projects_Create](#) example.

- Replace `YourResourceKey` with your Speech resource key.
- Replace `YourResourceRegion` with your Speech resource region.
- Replace `ProjectId` with a project ID of your choice. The case sensitive ID must be unique within your Speech resource. The ID will be used in the project's URI and can't be changed later.

Azure CLI

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourResourceKey" -H "Content-Type: application/json" -d '{  
    "description": "Project description",  
    "kind": "PersonalVoice"  
} '  
"https://YourResourceRegion.api.cognitive.microsoft.com/customvoice/projects/ProjectId?api-version=2024-02-01-preview"
```

You should receive a response body in the following format:

JSON

```
{  
    "id": "ProjectId",  
    "description": "Project description",  
    "kind": "PersonalVoice",  
    "createdDateTime": "2024-09-01T05:30:00.000Z"  
}
```

You use the project `id` in subsequent API requests to [add user consent](#) and [get a speaker profile ID](#).

Next steps

[Add user consent to the personal voice project.](#)

Add user consent to the personal voice project

08/07/2025

With the personal voice feature, it's required that every voice be created with explicit consent from the user. A recorded statement from the user is required acknowledging that the customer (Azure AI Speech resource owner) will create and use their voice.

To add user consent to the personal voice project, you provide the prerecorded consent audio file [from a publicly accessible URL \(Consents_Create\)](#) or [upload the audio file \(Consents_Post\)](#).

Consent statement

You need an audio recording of the user speaking the consent statement.

You can get the consent statement text for each locale from the text to speech GitHub repository. See [verbal-statement-all-locales.txt](#) for the consent statement. Below is a sample for the `en-US` locale:

"I [state your first and last name] am aware that recordings of my voice will be used by [state the name of the company] to create and use a synthetic version of my voice."

Supported audio formats for consent audio

See the table below for the supported formats for consent audio files:

[] Expand table

Format	Sample rate	Bit rate	Bit depth
mp3	16 kHz, 24 kHz, 44.1 kHz, 48 kHz	128 kbps, 192 kbps, 256 kbps, 320 kbps	/
wav	16 kHz, 24 kHz, 44.1 kHz, 48 kHz	/	16-bit, 24-bit, 32-bit

Add consent from a file

In this scenario, the audio files must be available locally.

To add consent to a personal voice project from a local audio file, use the `Consents_Post` operation of the custom voice API. Construct the request body according to the following instructions:

- Set the required `projectId` property. See [create a project](#).
- Set the required `voiceTalentName` property. The voice talent name can't be changed later.
- Set the required `companyName` property. The company name can't be changed later.
- Set the required `audiodata` property with the consent audio file.
- Set the required `locale` property. This should be the locale of the consent. The locale can't be changed later. You can find the text to speech locale list [here](#).

Make an HTTP POST request using the URI as shown in the following `Consents_Post` example.

- Replace `YourResourceKey` with your Speech resource key.
- Replace `YourResourceRegion` with your Speech resource region.
- Replace `JessicaConsentId` with a consent ID of your choice. The case sensitive ID will be used in the consent's URI and can't be changed later.

Azure CLI

```
curl -v -X POST -H "Ocp-Apim-Subscription-Key: YourResourceKey" -F
'description="Consent for Jessica voice"' -F 'projectId="ProjectId"' -F
'voiceTalentName="Jessica Smith"' -F 'companyName="Contoso"' -F
'audiodata=@"D:\PersonalVoiceTest\jessica-consent.wav"' -F 'locale="en-US"'
"https://YourResourceRegion.api.cognitive.microsoft.com/customvoice/consents/JessicaConsentId?api-version=2024-02-01-preview"
```

You should receive a response body in the following format:

JSON

```
{
  "id": "JessicaConsentId",
  "description": "Consent for Jessica voice",
  "projectId": "ProjectId",
  "voiceTalentName": "Jessica Smith",
  "companyName": "Contoso",
  "locale": "en-US",
  "status": "NotStarted",
  "createdDateTime": "2024-09-01T05:30:00.000Z",
  "lastActionDateTime": "2024-09-02T10:15:30.000Z"
}
```

The response header contains the `Operation-Location` property. Use this URI to get details about the `Consents_Post` operation. Here's an example of the response header:

HTTP

`Operation-Location:`

`https://eastus.api.cognitive.microsoft.com/customvoice/operations/070f7986-ef17-41d0-ba2b-907f0f28e314?api-version=2024-02-01-preview`

`Operation-Id:` 070f7986-ef17-41d0-ba2b-907f0f28e314

Add consent from a URL

In this scenario, the audio files must already be stored in an Azure Blob Storage container.

To add consent to a personal voice project from the URL of an audio file, use the `Consents_Create` operation of the custom voice API. Construct the request body according to the following instructions:

- Set the required `projectId` property. See [create a project](#).
- Set the required `voiceTalentName` property. The voice talent name can't be changed later.
- Set the required `companyName` property. The company name can't be changed later.
- Set the required `audioUrl` property. The URL of the voice talent consent audio file. Use a URI with the [shared access signatures \(SAS\)](#) token.
- Set the required `locale` property. This should be the locale of the consent. The locale can't be changed later. You can find the text to speech locale list [here](#).

Make an HTTP PUT request using the URI as shown in the following `Consents_Create` example.

- Replace `YourResourceKey` with your Speech resource key.
- Replace `YourResourceRegion` with your Speech resource region.
- Replace `JessicaConsentId` with a consent ID of your choice. The case sensitive ID will be used in the consent's URI and can't be changed later.

Azure CLI

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourResourceKey" -H "Content-Type: application/json" -d '{
  "description": "Consent for Jessica voice",
  "projectId": "ProjectId",
  "voiceTalentName": "Jessica Smith",
  "companyName": "Contoso",
  "audioUrl": "https://contoso.blob.core.windows.net/public/jessica-consent.wav?mySasToken",
  "locale": "en-US"
}'
```

```
"https://YourResourceRegion.api.cognitive.microsoft.com/customvoice/consents/JessicaConsentId?api-version=2024-02-01-preview"
```

You should receive a response body in the following format:

JSON

```
{  
    "id": "JessicaConsentId",  
    "description": "Consent for Jessica voice",  
    "projectId": "ProjectId",  
    "voiceTalentName": "Jessica Smith",  
    "companyName": "Contoso",  
    "locale": "en-US",  
    "status": "NotStarted",  
    "createdDateTime": "2024-09-01T05:30:00.000Z",  
    "lastActionDateTime": "2024-09-02T10:15:30.000Z"  
}
```

The response header contains the `Operation-Location` property. Use this URI to get details about the `Consents_Create` operation. Here's an example of the response header:

HTTP

`Operation-Location:`

`https://eastus.api.cognitive.microsoft.com/customvoice/operations/070f7986-ef17-41d0-ba2b-907f0f28e314?api-version=2024-02-01-preview`

`Operation-Id:` 070f7986-ef17-41d0-ba2b-907f0f28e314

Next steps

[Create a personal voice.](#)

Get a speaker profile ID for the personal voice

08/07/2025

To use personal voice in your application, you need to get a speaker profile ID. The speaker profile ID is used to generate synthesized audio with the text input provided.

You create a speaker profile ID based on the speaker's verbal consent statement and an audio prompt (a clean human voice sample between 5 - 90 seconds). The user's voice characteristics are encoded in the `speakerProfileId` property that's used for text to speech. For more information, see [use personal voice in your application](#).

(!) Note

The personal voice ID and speaker profile ID aren't same. You can choose the personal voice ID, but the speaker profile ID is generated by the service. The personal voice ID is used to manage the personal voice. The speaker profile ID is used for text to speech.

You provide the audio files [from a publicly accessible URL \(PersonalVoices_Create\)](#) or [upload the audio files \(PersonalVoices_Post\)](#).

Prompt audio format

The supported formats for prompt audio files are:

[\[+\] Expand table](#)

Format	Sample rate	Bit rate	Bit depth
mp3	16 kHz, 24 kHz, 44.1 kHz, 48 kHz	128 kbps, 192 kbps, 256 kbps, 320 kbps	/
wav	16 kHz, 24 kHz, 44.1 kHz, 48 kHz	/	16-bit, 24-bit, 32-bit

Create personal voice from a file

In this scenario, the audio files must be available locally.

To create a personal voice and get the speaker profile ID, use the [PersonalVoices_Post](#) operation of the custom voice API. Construct the request body according to the following instructions:

- Set the required `projectId` property. See [create a project](#).
- Set the required `consentId` property. See [add user consent](#).
- Set the required `audiodata` property. You can specify one or more audio files in the same request.

Make an HTTP POST request using the URI as shown in the following [PersonalVoices_Post](#) example.

- Replace `YourResourceKey` with your Speech resource key.
- Replace `YourResourceRegion` with your Speech resource region.
- Replace `JessicaPersonalVoiceId` with a personal voice ID of your choice. The case sensitive ID will be used in the personal voice's URI and can't be changed later.

Azure CLI

```
curl -v -X POST -H "Ocp-Apim-Subscription-Key: YourResourceKey" -F
'projectId="ProjectId"' -F 'consentId="JessicaConsentId"' -F
'audiodata=@"D:\PersonalVoiceTest\CNVSample001.wav"' -F
'audiodata=@"D:\PersonalVoiceTest\CNVSample002.wav"' "
https://YourResourceRegion.api.cognitive.microsoft.com/customvoice/personalvoices/
JessicaPersonalVoiceId?api-version=2024-02-01-preview"
```

You should receive a response body in the following format:

JSON

```
{
  "id": "JessicaPersonalVoiceId",
  "speakerProfileId": "3059912f-a3dc-49e3-bdd0-02e449df1fe3",
  "projectId": "ProjectId",
  "consentId": "JessicaConsentId",
  "status": "NotStarted",
  "createdDateTime": "2024-09-01T05:30:00.000Z",
  "lastActionDateTime": "2024-09-02T10:15:30.000Z"
}
```

Use the `speakerProfileId` property to integrate personal voice in your text to speech application. For more information, see [use personal voice in your application](#).

The response header contains the `Operation-Location` property. Use this URI to get details about the [PersonalVoices_Post](#) operation. Here's an example of the response header:

HTTP

Operation-Location:

```
https://eastus.api.cognitive.microsoft.com/customvoice/operations/1321a2c0-9be4-471d-83bb-bc3be4f96a6f?api-version=2024-02-01-preview
```

```
Operation-Id: 1321a2c0-9be4-471d-83bb-bc3be4f96a6f
```

Create personal voice from a URL

In this scenario, the audio files must already be stored in an Azure Blob Storage container.

To create a personal voice and get the speaker profile ID, use the [PersonalVoices_Create](#) operation of the custom voice API. Construct the request body according to the following instructions:

- Set the required `projectId` property. See [create a project](#).
- Set the required `consentId` property. See [add user consent](#).
- Set the required `audios` property. Within the `audios` property, set the following properties:
 - Set the required `containerUrl` property to the URL of the Azure Blob Storage container that contains the audio files. Use [shared access signatures \(SAS\) SAS for a container](#) with both read and list permissions.
 - Set the required `extensions` property to the extensions of the audio files.
 - Optionally, set the `prefix` property to set a prefix for the blob name.

Make an HTTP PUT request using the URI as shown in the following [PersonalVoices_Create](#) example.

- Replace `YourResourceKey` with your Speech resource key.
- Replace `YourResourceRegion` with your Speech resource region.
- Replace `JessicaPersonalVoiceId` with a personal voice ID of your choice. The case sensitive ID will be used in the personal voice's URI and can't be changed later.

Azure CLI

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourResourceKey" -H "Content-Type: application/json" -d '{
    "projectId": "ProjectId",
    "consentId": "JessicaConsentId",
    "audios": {
        "containerUrl": "https://contoso.blob.core.windows.net/voicecontainer?mySasToken",
        "prefix": "jessica/",
        "extensions": [
            ".wav"
        ]
    }
}'
```

```
        ]
    }
}

"https://YourResourceRegion.api.cognitive.microsoft.com/customvoice/personalvoices
/JessicaPersonalVoiceId?api-version=2024-02-01-preview"

# Ensure the `containerUrl` has both read and list permissions.
# Ensure the `.wav` files are located in the "jessica" folder within the
# container. The `prefix` matches all `.wav` files in the "jessica" folder. If there
# is no such folder, the prefix will match `.wav` files with names starting with
# "jessica".
```

You should receive a response body in the following format:

JSON

```
{
  "id": "JessicaPersonalVoiceId",
  "speakerProfileId": "3059912f-a3dc-49e3-bdd0-02e449df1fe3",
  "projectId": "ProjectId",
  "consentId": "JessicaConsentId",
  "status": "NotStarted",
  "createdDateTime": "2024-09-01T05:30:00.000Z",
  "lastActionDateTime": "2024-09-02T10:15:30.000Z"
}
```

Use the `speakerProfileId` property to integrate personal voice in your text to speech application. For more information, see [use personal voice in your application](#).

The response header contains the `Operation-Location` property. Use this URI to get details about the [PersonalVoices_Create](#) operation. Here's an example of the response header:

HTTP

```
Operation-Location:
https://eastus.api.cognitive.microsoft.com/customvoice/operations/1321a2c0-9be4-
471d-83bb-bc3be4f96a6f?api-version=2024-02-01-preview
Operation-Id: 1321a2c0-9be4-471d-83bb-bc3be4f96a6f
```

Next steps

[Use personal voice in your application.](#)

Use personal voice in your application

08/07/2025

You can use the [speaker profile ID](#) for your personal voice to synthesize speech in any of the 91 languages supported across 100+ locales. A locale tag isn't required. Personal voice uses automatic language detection at the sentence level.

Integrate personal voice in your application

You need to use [speech synthesis markup language \(SSML\)](#) to use personal voice in your application. SSML is an XML-based markup language that provides a standard way to mark up text for the generation of synthetic speech. SSML tags are used to control the pronunciation, volume, pitch, rate, and other attributes of the speech synthesis output.

- The `speakerProfileId` property in SSML is used to specify the [speaker profile ID](#) for the personal voice.
- The voice name is specified in the `name` property in SSML. For personal voice, the voice name must be one of the supported base model voice names. To get a list of supported base model voice names, use the [BaseModels_List](#) operation of the custom voice API.

ⓘ Note

The voice names labeled with the `Latest`, such as `DragonLatestNeural` or `PhoenixLatestNeural`, will be updated from time to time; its performance may vary with updates for ongoing improvements. If you would like to use a fixed version, select one labeled with a version number, such as `PhoenixV2Neural`.

- `DragonLatestNeural` is a base model with superior voice cloning similarity compared to `PhoenixLatestNeural`. `PhoenixLatestNeural` is a base model with more accurate pronunciation and lower latency than `DragonLatestNeural`.
- For personal voice, you can use the `<lang xml:lang>` element to adjust the speaking language. It's the same as with multilingual voices. See [how to use the lang element to speak different languages](#).

Here's example SSML in a request for text to speech with the voice name and the speaker profile ID. The sample also demonstrates how to switch languages from `en-US` to `zh-HK` using the `<lang xml:lang>` element.

XML

```
<speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
      xmlns:mstts='http://www.w3.org/2001/mstts' xml:lang='en-US'>
  <voice name='DragonLatestNeural'>
    <mstts:ttsembedding speakerProfileId='your speaker profile ID here'>
      I'm happy to hear that you find me amazing and that I have made your
      trip planning easier and more fun.
      <lang xml:lang='zh-HK'>我很高興聽到你覺得我很了不起，我讓你的旅行計劃更輕
      鬆、更有趣。</lang>
    </mstts:ttsembedding>
  </voice>
</speak>
```

You can use the SSML via the [Speech SDK](#) or [REST API](#).

- **Real-time speech synthesis:** Use the [Speech SDK](#) or [REST API](#) to convert text to speech.
 - When you use Speech SDK, don't set Endpoint ID, just like prebuild voice.
 - When you use REST API, please use standard voices endpoint.

Supported and unsupported SSML elements for personal voice

For detailed information on the supported and unsupported SSML elements for Phoenix and Dragon models, refer to the following table. For instructions on how to use SSML elements, refer to the [SSML document structure and events](#).

[] Expand table

Element	Description	Supported in Phoenix	Supported in Dragon
<code><voice></code>	Specifies the voice and optional effects (<code>eq_car</code> and <code>eq_telecomhp8k</code>).	Yes	Yes
<code><mstts:express-as></code>	Specifies speaking styles and roles.	No	No
<code><mstts:ttsembedding></code>	Specifies the <code>speakerProfileId</code> property for a personal voice.	Yes	Yes
<code><lang xml:lang></code>	Specifies the speaking language.	Yes	Yes
<code><prosody></code>	Adjusts pitch, contour, range, rate, and volume.		
<code>pitch</code>	Indicates the baseline pitch for the text.	No	No

Element	Description	Supported in Phoenix	Supported in Dragon
<code>contour</code>	Represents changes in pitch.	No	No
<code>range</code>	Represents the range of pitch for the text.	No	No
<code>rate</code>	Indicates the speaking rate of the text.	Yes	Yes
<code>volume</code>	Indicates the volume level of the speaking voice.	No	No
<code><emphasis></code>	Adds or removes word-level stress for the text.	No	No
<code><audio></code>	Embeds prerecorded audio into an SSML document.	Yes	No
<code><mstts:audioduration></code>	Specifies the duration of the output audio.	No	No
<code><mstts:backgroundaudio></code>	Adds background audio to your SSML documents or mixes an audio file with text to speech.	Yes	No
<code><phoneme></code>	Specifies phonetic pronunciation in SSML documents.		
<code>ipa</code>	One of the phonetic alphabets.	Yes	No
<code>sapi</code>	One of the phonetic alphabets.	No	No
<code>ups</code>	One of the phonetic alphabets.	Yes	No
<code>x-sampa</code>	One of the phonetic alphabets.	Yes	No
<code><lexicon></code>	Defines how multiple entities are read in SSML.	Yes	Yes (only support alias)
<code><say-as></code>	Indicates the content type, such as number or date, of the element's text.	Yes	Yes
<code><sub></code>	Indicates that the alias attribute's text value should be pronounced instead of the element's enclosed text.	Yes	Yes
<code><math></code>	Uses the MathML as input text to properly pronounce mathematical notations in the output audio.	Yes	No
<code><bookmark></code>	Gets the offset of each marker in the audio stream.	Yes	No

Element	Description	Supported in Phoenix	Supported in Dragon
<break>	Overrides the default behavior of breaks or pauses between words.	Yes	Yes
<mstts:silence>	Inserts pauses before or after text, or between two adjacent sentences.	Yes	No
<mstts:viseme>	Defines the position of the face and mouth while a person is speaking.	Yes	No
<p>	Denotes paragraphs in SSML documents.	Yes	Yes
<s>	Denotes sentences in SSML documents.	Yes	Yes

Supported and unsupported SDK features for personal voice

The following table outlines which SDK features are supported for Phoenix and Dragon models. For details on how to utilize these SDK features in your applications, refer to [Subscribe to synthesizer events](#).

[+] [Expand table](#)

SDK features	Description	Supported in Phoenix	Supported in Dragon
Word boundary	Signals that a word boundary was received during synthesis, providing precise word timing during the speech synthesis process.	Yes	No
Viseme events	Provides viseme (lips, jaw, and tongue movement) information during synthesis, allowing visual synchronization.	Yes	No

Reference documentation

[Custom voice REST API reference documentation](#)

Next steps

- Learn more about custom voice in the [overview](#).

- Learn more about Speech Studio in the [overview](#).

What is Text to speech avatar?

Text to speech avatar converts text into a digital video of a photorealistic human (either a standard avatar or a [custom text to speech avatar](#)) speaking with a natural-sounding voice. The text to speech avatar video can be synthesized asynchronously or in real time. Developers can build applications integrated with text to speech avatar through an API, or use a content creation tool on Speech Studio to create video content without coding.

With text to speech avatar's advanced neural network models, the feature empowers users to deliver life-like and high-quality synthetic talking avatar videos for various applications while adhering to [responsible AI practices](#).

Tip

To convert text to speech with a no-code approach, try the [Text to speech avatar tool in Speech Studio](#).

Avatar capabilities

Text to speech avatar capabilities include:

- Converts text into a digital video of a photorealistic human speaking with natural-sounding voices powered by Azure AI text to speech.
- Provides a collection of standard avatars.
- Azure AI text to speech generates the voice of the avatar. For more information, see [Avatar voice and language](#).
- Synthesizes text to speech avatar video asynchronously with the [batch synthesis API](#) or in [real-time](#).
- Provides a [content creation tool](#) in Speech Studio for creating video content without coding.
- Enables real-time avatar conversations through the [live chat avatar tool](#) in Speech Studio.

With text to speech avatar's advanced neural network models, the feature empowers you to deliver lifelike and high-quality synthetic talking avatar videos for various applications while adhering to responsible AI practices.

Avatar voice and language

You can choose from a range of standard voices for the avatar. The language support for text to speech avatar is the same as the language support for text to speech. For details, see [Language and voice support for the Speech service](#). Standard text to speech avatars can be accessed through the [Speech Studio portal](#) or via API.

The voice in the synthetic video could be an Azure AI Speech standard voice or the [custom voice](#) of voice talent selected by you.

Avatar video output

Both batch synthesis and real-time synthesis resolution are 1920 x 1080, and the frames per second (FPS) are 25. Batch synthesis codec can be h264, hevc, or av1 if the format is mp4 and can set codec as vp9 or av1 if the format is webm; only vp9 can contain an alpha channel. Real-time synthesis codec is h264. Video bitrate can be configured for both batch synthesis and real-time synthesis in the request; the default value is 2000000; more detailed configurations can be found in the sample code.

[] [Expand table](#)

	Batch synthesis	Real-time synthesis
Resolution	1920 x 1080	1920 x 1080
FPS	25	25
Codec	h264/hevc/vp9/av1	h264

Custom text to speech avatar

You can create custom text to speech avatars that are unique to your product or brand. All it takes to get started is taking 10 minutes of video recordings. If you're also fine-tuning a professional voice for the actor, the avatar can be highly realistic.

Voice sync for avatar is trained alongside the custom avatar utilizing audio from the training video. The voice is exclusively associated with the custom avatar and can't be independently used.

[Professional voice fine-tuning](#) and [custom text to speech avatar](#) are separate features. You can use them independently or together. If you plan to also use [professional voice fine-tuning](#) with a text to speech avatar, you need to deploy or [copy](#) your fine-tuned professional voice model to one of the [avatar supported regions](#).

For more information, see [What is custom text to speech avatar](#).

Sample code

Sample code for text to speech avatar is available on [GitHub ↗](#). These samples cover the most popular scenarios:

- [Batch synthesis \(REST\) ↗](#)
- [Real-time synthesis \(SDK\) ↗](#)
- [Live chat with Azure OpenAI in behind \(SDK\) ↗](#)
- To create a live chat APP with Azure OpenAI [On Your Data](#), you can refer to [this sample code ↗](#) (search "On Your Data")

Pricing

- Throughout an avatar real-time session or batch content creation, the text to speech, speech to text, Azure OpenAI, or other Azure services are charged separately.
- Voice sync for avatar (via custom avatar training) is charged the same as a personal voice in terms of voice creation and synthesis. The storage of the voice is free.
- Refer to [text to speech avatar pricing note](#) to learn how billing works for the text-to-speech avatar feature.
- For the detailed pricing, see [Speech service pricing ↗](#). Avatar pricing is only visible for service regions where the feature is available. For the current list of supported regions, see the [Speech service regions table](#).

Available locations

For the current list of regions that support text to speech avatar, see the [Speech service regions table](#).

Responsible AI

We care about the people who use AI and the people who will be affected by it as much as we care about technology. For more information, see the Responsible AI [transparency notes](#) and [disclosure for voice and avatar talent](#).

Next steps

- [Use batch synthesis for text to speech avatar](#)
- [What is custom text to speech avatar](#)

Last updated on 11/09/2025

Content credentials

08/07/2025

Content credentials help you verify that Azure text to speech avatar generated your video content. This transparency feature automatically adds tamper-evident information about your video's origin and creation history, following industry standards from the [Coalition for Content Provenance and Authenticity \(C2PA\)](#).

Use content credentials to:

- Confirm video authenticity for your audiences
- Meet transparency requirements for AI-generated content
- Build trust in your avatar-powered solutions

Content credentials are automatically applied to all supported video formats—no additional setup required.

What are content credentials?

Azure text to speech avatar adds a content credentials manifest to each avatar video, giving you information about the video's origin. A cryptographic signature secures the manifest, linking it back to Azure text to speech avatar.

The manifest has several key pieces of information:

[] [Expand table](#)

Field name	Field content
"generator"	This field has a value of "Microsoft Azure Txt to Speech Avatar Service" for all applicable videos, confirming the AI-generated nature of the video.
"when"	The timestamp of when the content credentials were created.

Content credentials in Azure text to speech avatar can help people understand when video content is generated by the Azure text to speech avatar system. For more information on how to responsibly build solutions with text to speech avatar models, see the [Text to speech transparency note](#).

Limitations

The content credentials are only supported in video files generated by batch synthesis of text to speech avatar, and only `mp4` file format is supported.

Use content credentials in your solution

You can use content credentials by ensuring that your Azure text to speech avatar generated video files have content credentials.

No extra setup is necessary. Content credentials are automatically applied to all applicable videos generated by Azure text to speech avatar.

Check that a video file has content credentials

You can check the content credentials of a text to speech avatar using either of the following methods:

- **Content credentials verify webpage (contentcredentials.org/verify):** This tool lets users inspect the content credentials of content like a video. If an uploaded video of a text to speech avatar is created with the Speech service, the tool will show that and display the issued date and time of the content credentials. This screenshot shows an example avatar video upload result where the content credentials were issued by Microsoft.

Content credentials

Select another file from your device or drag and drop anywhere

 Avatar video generated by Micros...
Oct 30, 2024

 Oct 30, 2024



Process
The app or device used to produce this content recorded the following info:

App or device used
Microsoft Azure Text To Speech Avatar Service 1.0

About this Content Credential

Issued by
Microsoft Corporation 

Issued on
 Oct 30, 2024 at 3:40 PM GMT+8

[Change language](#)  

This page shows that a video generated by Azure text to speech avatar has content credentials issued by Microsoft.

- **Content Authenticity Initiative (CAI) open-source tools:** The CAI provides multiple open-source tools that can be used to validate and display C2PA content credentials. Find the right tool for your application and [get started here](#).

Next steps

- [Use batch synthesis for text to speech avatar](#)

How to use text to speech avatar with real-time synthesis

This guide shows you how to use text to speech avatar with real-time synthesis. The avatar video is generated almost instantly after you enter text.

Prerequisites

You need:

- **Azure subscription:** [Create one for free ↗](#).
- **Speech resource:** [Create a speech resource ↗](#) in the Azure portal. Select the **Standard S0** pricing tier to access avatars.
- **Speech resource key and region:** After deployment, select **Go to resource** to view and manage your keys.

Set up environment

To use real-time avatar synthesis, install the Speech SDK for JavaScript for your webpage. See [Install the Speech SDK](#).

Real-time avatar works on these 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 Doesn'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

Speech service supports many [languages and voices](#). See the full list or try them in the [Voice Gallery ↗](#).

To match your input text and use a specific voice, set the `SpeechSynthesisLanguage` or `SpeechSynthesisVoiceName` properties in the `SpeechConfig` object:

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 you select **es-ES-ElviraNeural** and enter English text, the avatar speaks English with a Spanish accent.

If the voice doesn't support the input language, the Speech service doesn't create audio. See [Language and voice support](#) for the full list.

Default voice selection:

- If you don't set `SpeechSynthesisVoiceName` or `SpeechSynthesisLanguage`, the default voice in `en-US` is used.
- If you set only `SpeechSynthesisLanguage`, the default voice in that locale is used.
- If you set both, `SpeechSynthesisVoiceName` takes priority.
- If you use SSML to set the voice, both properties are ignored.

Select avatar character and style

See [supported avatar characters and styles](#).

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 the WebRTC protocol to stream video. Set up the connection with the avatar service using a WebRTC peer connection.

First, create a WebRTC peer connection object. WebRTC is peer-to-peer and relies on an ICE server for network relay. Speech service provides a REST API to get ICE server info. We recommend fetching ICE server details from Speech service, but you can use your own.

Sample request to fetch ICE info:

HTTP

```
GET /cognitiveservices/avatar/relay/token/v1 HTTP/1.1
```

```
Host: westus2.tts.speech.microsoft.com  
Ocp-Apim-Subscription-Key: YOUR_RESOURCE_KEY
```

Create the WebRTC peer connection using the ICE server URL, username, and credential from the previous response:

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 can start with `turn` (for example, `turn:relay.communication.microsoft.com:3478`) or `stun` (for example, `stun:relay.communication.microsoft.com:3478`). For `urls`, include only the `turn` URL.

Next, set up the video and audio player elements in the `ontrack` callback of the peer connection. This callback runs twice—once for video, once for audio. Create both player elements in the callback:

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' })

```

Then, use Speech SDK to create an avatar synthesizer and connect to the avatar service with the peer connection:

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) }
);

```

The real-time API disconnects after 5 minutes of idle or after 30 minutes of connection. To keep the avatar running longer, enable automatic reconnect. See this [JavaScript sample code](#) (search "auto reconnect").

Synthesize talking avatar video from text input

After setup, the avatar video plays in your browser. The avatar blinks and moves slightly, but doesn't speak until you send text input.

Send text to the avatar synthesizer to make 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()
});
```

Close the real-time avatar connection

To avoid extra costs, close the connection when you're done:

- Closing the browser releases the WebRTC peer connection and closes the avatar connection after a few seconds.
- The connection closes automatically if the avatar is idle for 5 minutes.
- You can close the avatar connection manually:

JavaScript

```
avatarSynthesizer.close()
```

Edit background

Set background color

Set the background color of the avatar video using the `backgroundColor` property of `AvatarConfig`:

JavaScript

```
const avatarConfig = new SpeechSDK.AvatarConfig(
    "lisa", // Set avatar character here.
    "casual-sitting", // Set avatar style here.
)
avatarConfig.backgroundColor = '#00FF00FF' // Set background color to green
```

ⓘ Note

The color string should be in the format `#RRGGBBAA`. The alpha channel (`AA`) is ignored—transparent backgrounds aren't supported for real-time avatar.

Set background image

Set the background image using the `backgroundImage` property of `AvatarConfig`. Upload your image to a public URL and assign it to `backgroundImage`:

JavaScript

```
const avatarConfig = new SpeechSDK.AvatarConfig(  
    "lisa", // Set avatar character here.  
    "casual-sitting", // Set avatar style here.  
)  
avatarConfig.backgroundImage = "https://www.example.com/1920-1080-image.jpg" // A  
public accessible URL of the image.
```

Set background video

The API doesn't support background video directly, but you can customize the background on the client side:

- Set the background color to green (for easy matting).
- Create a canvas element the same size as the avatar video.
- For each frame, set green pixels to transparent and draw the frame to the canvas.
- Hide the original video.

This gives you a transparent avatar on a canvas. See [JavaScript sample code ↗](#).

You can then place any dynamic content (like a video) behind the canvas.

Crop video

The avatar video is 16:9 by default. To crop to a different aspect ratio, specify the rectangle area using the coordinates of the top-left and bottom-right corners:

JavaScript

```
const videoFormat = new SpeechSDK.AvatarVideoFormat()  
const topLeftCoordinate = new SpeechSDK.Coordinate(640, 0) // coordinate of top-left  
vertex, with X=640, Y=0
```

```
const bottomRightCoordinate = new SpeechSDK.Coordinate(1320, 1080) // coordinate of  
bottom-right vertex, with X=1320, Y=1080  
videoFormat.setCropRange(topLeftCoordinate, bottomRightCoordinate)  
const avatarConfig = new SpeechSDK.AvatarConfig(  
    "lisa", // Set avatar character here.  
    "casual-sitting", // Set avatar style here.  
    videoFormat, // Set video format here.  
)
```

For a full sample, see our [code example ↗](#) and search for `crop`.

Code samples

Find text to speech avatar code samples in the Speech SDK GitHub repository. These samples show how to use real-time avatars in web and mobile apps:

- **Server + client**
 - [Python \(server\) + JavaScript \(client\) ↗](#)
 - [C# \(server\) + JavaScript \(client\) ↗](#)
- **Client only**
 - [JavaScript ↗](#)
 - [Android ↗](#)
 - [iOS ↗](#)

Next steps

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

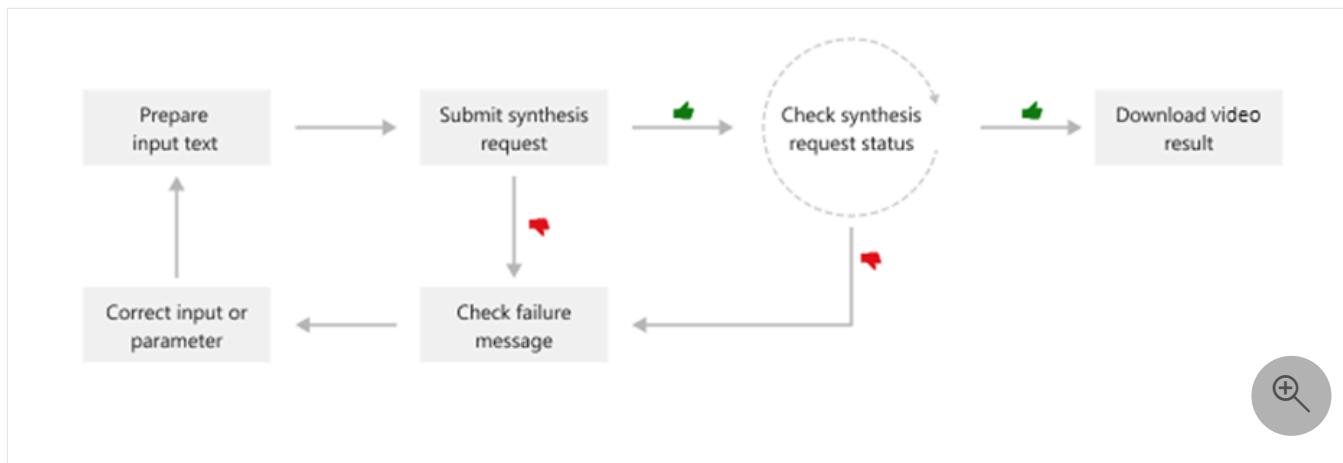
Last updated on 11/09/2025

Use batch synthesis for text to speech avatar

The batch synthesis API for text to speech avatar lets you synthesize text asynchronously into a talking avatar as a video file. Publishers and video content platforms can use this API to create avatar video content in a batch. That approach can be suitable for different use cases like training materials, presentations, or advertisements.

The synthetic avatar video will be generated asynchronously after the system receives text input. The generated video output can be downloaded in batch mode synthesis. You submit text for synthesis, poll for the synthesis status, and download the video output when the status shows success. The text input formats must be plain text or Speech Synthesis Markup Language (SSML) text.

This diagram provides a high-level overview of the workflow.



To run batch synthesis, you can use the following REST API operations.

[] Expand table

Operation	Method	REST API call
Create batch synthesis	PUT	avatar/batchsyntheses/{SynthesisId}?api-version=2024-08-01
Get batch synthesis	GET	avatar/batchsyntheses/{SynthesisId}?api-version=2024-08-01
List batch synthesis	GET	avatar/batchsyntheses/?api-version=2024-08-01
Delete batch synthesis	DELETE	avatar/batchsyntheses/{SynthesisId}?api-version=2024-08-01

You can refer to the code samples on [GitHub](#).

Create a batch synthesis request

Some properties in JSON format are required when you create a new batch synthesis job. Other properties are optional. The [batch synthesis response](#) includes other properties to provide information about the synthesis status and results. For example, the `outputs.result` property has the location from [where you can download a video file](#) containing the avatar video. From `outputs.summary`, you can get the summary and debug details.

To submit a batch synthesis request, construct the HTTP POST request body following these instructions:

- Set the required `inputKind` property.
- If the `inputKind` property is set to `PlainText`, you must also set the `voice` property in the `synthesisConfig`. In the following example, the `inputKind` is set to `SSML`, so the `speechSynthesis` isn't set.
- Set the required `SynthesisId` property. Choose a unique `SynthesisId` for the same speech resource. The `SynthesisId` can be a string of 3 to 64 characters, including letters, numbers, '-', or '_', with the condition that it must start and end with a letter or number.
- Set the required `talkingAvatarCharacter` and `talkingAvatarStyle` properties. You can find supported avatar characters and styles [here](#).
- Optionally, you can set the `videoFormat`, `backgroundColor`, and other properties. For more information, see [batch synthesis properties](#).

! Note

The maximum JSON payload size accepted is 500 kilobytes.

Each Speech resource can have up to 200 batch synthesis jobs running concurrently.

The maximum length for the output video is currently 20 minutes, with potential increases in the future.

To make an HTTP PUT request, use the URI format shown in the following example. Replace `YourSpeechKey` with your Speech resource key, `YourSpeechRegion` with your Speech resource region, and set the request body properties as described previously.

Azure CLI

```
curl -v -X PUT -H "Ocp-Apim-Subscription-Key: YourSpeechKey" -H "Content-Type: application/json" -d '{  
    "inputKind": "SSML",  
    "inputs": [  
        {  
            "content": "<speak version='1.0'><xml:lang='en-US'><voice name='en-US-AvaMultilingualNeural'>The rainbow has seven colors.</voice>"  
        }  
    ]  
}'
```

```
</speak>
    }
],
"avatarConfig": {
    "talkingAvatarCharacter": "lisa",
    "talkingAvatarStyle": "graceful-sitting"
}
}' "https://YourSpeechRegion.api.cognitive.microsoft.com/avatar/batchsyntheses/my-job-01?api-version=2024-08-01"
```

You should receive a response body in the following format:

JSON

```
{
    "id": "my-job-01",
    "internalId": "5a25b929-1358-4e81-a036-33000e788c46",
    "status": "NotStarted",
    "createdDateTime": "2024-03-06T07:34:08.9487009Z",
    "lastActionDateTime": "2024-03-06T07:34:08.9487012Z",
    "inputKind": "SSML",
    "customVoices": {},
    "properties": {
        "timeToLiveInHours": 744,
    },
    "avatarConfig": {
        "talkingAvatarCharacter": "lisa",
        "talkingAvatarStyle": "graceful-sitting",
        "videoFormat": "Mp4",
        "videoCodec": "hevc",
        "subtitleType": "soft_embedded",
        "bitrateKbps": 2000,
        "customized": false
    }
}
```

The `status` property should progress from `NotStarted` status to `Running` and finally to `Succeeded` or `Failed`. You can periodically call the [GET batch synthesis API](#) until the returned status is `Succeeded` or `Failed`.

Get batch synthesis

To get the status of a batch synthesis job, make an HTTP GET request using the URI as shown in the following example.

Replace `YourSynthesisId` with your batch synthesis ID, `YourSpeechKey` with your Speech resource key, and `YourSpeechRegion` with your Speech resource region.

Azure CLI

```
curl -v -X GET  
"https://YourSpeechRegion.api.cognitive.microsoft.com/avatar/batchsyntheses/YourSynt  
hesisId?api-version=2024-08-01" -H "Ocp-Apim-Subscription-Key: YourSpeechKey"
```

You should receive a response body in the following format:

JSON

```
{  
    "id": "my-job-01",  
    "internalId": "5a25b929-1358-4e81-a036-33000e788c46",  
    "status": "Succeeded",  
    "createdDateTime": "2024-03-06T07:34:08.9487009Z",  
    "lastActionDateTime": "2024-03-06T07:34:12.5698769",  
    "inputKind": "SSML",  
    "customVoices": {},  
    "properties": {  
        "timeToLiveInHours": 744,  
        "sizeInBytes": 344460,  
        "durationInMilliseconds": 2520,  
        "succeededCount": 1,  
        "failedCount": 0,  
        "billingDetails": {  
            "neuralCharacters": 29,  
            "talkingAvatarDurationSeconds": 2  
        }  
    },  
    "avatarConfig": {  
        "talkingAvatarCharacter": "lisa",  
        "talkingAvatarStyle": "graceful-sitting",  
        "videoFormat": "Mp4",  
        "videoCodec": "hevc",  
        "subtitleType": "soft_embedded",  
        "bitrateKbps": 2000,  
        "customized": false  
    },  
    "outputs": {  
        "result": "https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-  
output/xxxxx/xxxxx/0001.mp4?SAS_Token",  
        "summary": "https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-  
output/xxxxx/xxxxx/summary.json?SAS_Token"  
    }  
}
```

From the `outputs.result` field, you can download a video file containing the avatar video. The `outputs.summary` field lets you download the summary and debug details. For more information on batch synthesis results, see [batch synthesis results](#).

List batch synthesis

To list all batch synthesis jobs for your Speech resource, make an HTTP GET request using the URI as shown in the following example.

Replace `YourSpeechKey` with your Speech resource key and `YourSpeechRegion` with your Speech resource region. Optionally, you can set the `skip` and `top` (page size) query parameters in the URL. The default value for `skip` is 0, and the default value for `maxpagesize` is 100.

Azure CLI

```
curl -v -X GET  
"https://YourSpeechRegion.api.cognitive.microsoft.com/avatar/batchsyntheses?  
skip=0&maxpagesize=2&api-version=2024-08-01" -H "Ocp-Apim-Subscription-Key:  
YourSpeechKey"
```

You receive a response body in the following format:

JSON

```

    "outputs": {
        "result":
"https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-
output/xxxxx/xxxxx/0001.mp4?SAS_Token",
        "summary":
"https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-
output/xxxxx/xxxxx/summary.json?SAS_Token"
    },
    {
        "id": "my-job-01",
        "internalId": "5a25b929-1358-4e81-a036-33000e788c46",
        "status": "Succeeded",
        "createdDateTime": "2024-03-06T07:34:08.9487009Z",
        "lastActionDateTime": "2024-03-06T07:34:12.5698769",
        "inputKind": "SSML",
        "customVoices": {},
        "properties": {
            "timeToLiveInHours": 744,
            "sizeInBytes": 344460,
            "durationInMilliseconds": 2520,
            "succeededCount": 1,
            "failedCount": 0,
            "billingDetails": {
                "neuralCharacters": 29,
                "talkingAvatarDurationSeconds": 2
            }
        },
        "avatarConfig": {
            "talkingAvatarCharacter": "lisa",
            "talkingAvatarStyle": "graceful-sitting",
            "videoFormat": "Mp4",
            "videoCodec": "hevc",
            "subtitleType": "soft_embedded",
            "bitrateKbps": 2000,
            "customized": false
        },
        "outputs": {
            "result":
"https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-
output/xxxxx/xxxxx/0001.mp4?SAS_Token",
            "summary":
"https://stttsvcprodusw2.blob.core.windows.net/batchsynthesis-
output/xxxxx/xxxxx/summary.json?SAS_Token"
        }
    }
],
"nextLink":
"https://YourSpeechRegion.api.cognitive.microsoft.com/avatar/batchsyntheses/?api-
version=2024-08-01&skip=2&maxpagesize=2"
}

```

From `outputs.result`, you can download a video file containing the avatar video. From `outputs.summary`, you can get the summary and debug details. For more information, see [batch](#)

synthesis results.

The `value` property in the JSON response lists your synthesis requests. The list is paginated, with a maximum page size of 100. The `nextLink` property is provided as needed to get the next page of the paginated list.

Get batch synthesis results file

Once you get a batch synthesis job with `status` of "Succeeded", you can download the video output results. Use the URL from the `outputs.result` property of the [get batch synthesis](#) response.

To get the batch synthesis results file, make an HTTP GET request using the URI as shown in the following example. Replace `YourOutputsResultUrl` with the URL from the `outputs.result` property of the [get batch synthesis](#) response. Replace `YourSpeechKey` with your Speech resource key.

Azure CLI

```
curl -v -X GET "YourOutputsResultUrl" -H "Ocp-Apim-Subscription-Key: YourSpeechKey"  
> output.mp4
```

To get the batch synthesis summary file, make an HTTP GET request using the URI as shown in the following example. Replace `YourOutputsResultUrl` with the URL from the `outputs.summary` property of the [get batch synthesis](#) response. Replace `YourSpeechKey` with your Speech resource key.

Azure CLI

```
curl -v -X GET "YourOutputsSummaryUrl" -H "Ocp-Apim-Subscription-Key: YourSpeechKey"  
> summary.json
```

The summary file has the synthesis results for each text input. Here's an example `summary.json` file:

JSON

```
{  
  "jobID": "5a25b929-1358-4e81-a036-33000e788c46",  
  "status": "Succeeded",  
  "results": [  
    {  
      "texts": [  
        "<speak version='1.0' xml:lang='en-US'><voice name='en-US-AvaMultilingualNeural'>The rainbow has seven colors.</voice></speak>"
```

```
        ],
        "status": "Succeeded",
        "videoFileName": "244a87c294b94ddeb3dbaccee8ffa7eb/5a25b929-1358-4e81-a036-  
33000e788c46/0001.mp4",
        "TalkingAvatarCharacter": "lisa",
        "TalkingAvatarStyle": "graceful-sitting"
    }
]
}
```

Delete batch synthesis

After you get the audio output results and no longer need the batch synthesis job history, you can delete it. The Speech service keeps each synthesis history for up to 31 days or the duration specified by the request's `timeToLiveInHours` property, whichever comes sooner. The date and time of automatic deletion for synthesis jobs with a status of "Succeeded" or "Failed" is calculated as the sum of the `lastActionDateTime` and `timeToLive` properties.

To delete a batch synthesis job, make an HTTP DELETE request using the following URI format. Replace `YourSynthesisId` with your batch synthesis ID, `YourSpeechKey` with your Speech resource key, and `YourSpeechRegion` with your Speech resource region.

Azure CLI

```
curl -v -X DELETE  
"https://YourSpeechRegion.api.cognitive.microsoft.com/avatar/batchsyntheses/YourSynt  
hesisId?api-version=2024-08-01" -H "Ocp-Apim-Subscription-Key: YourSpeechKey"
```

The response headers include `HTTP/1.1 204 No Content` if the delete request was successful.

Next steps

- [Batch synthesis properties](#)
- [Use batch synthesis for text to speech avatar](#)
- [What is text to speech avatar](#)

Batch synthesis properties for text to speech avatar

08/07/2025

Batch synthesis properties can be grouped as: avatar related properties, batch job related properties, and text to speech related properties, which are described in the following tables.

Some properties in JSON format are required when you create a new batch synthesis job. Other properties are optional. The batch synthesis response includes other properties to provide information about the synthesis status and results. For example, the `outputs.result` property contains the location from where you can download a video file containing the avatar video. From `outputs.summary`, you can access the summary and debug details.

Avatar properties

The following table describes the avatar properties.

 Expand table

Property	Description
avatarConfig.talkingAvatarCharacter	<p>The character name of the talking avatar.</p> <p>For standard avatar, the supported avatar characters can be found here.</p> <p>For custom avatar, specify the avatar model name.</p> <p>This property is required.</p>
avatarConfig.talkingAvatarStyle	<p>The style name of the talking avatar.</p> <p>For standard avatar, the supported avatar styles can be found here.</p> <p>For custom avatar, this property should be omitted.</p> <p>This property is required for standard avatar.</p>
avatarConfig.customized	<p>A bool value indicating whether the avatar to be used is customized avatar or not. True for customized avatar, and false for standard avatar.</p> <p>This property is optional, and the default value is <code>false</code>.</p>

Property	Description
avatarConfig.videoFormat	<p>The format for output video file could be mp4 or webm.</p> <p>The <code>webm</code> format is required for a transparent background.</p> <p>This property is optional, and the default value is mp4.</p>
avatarConfig.videoCodec	<p>The codec for output video, could be h264, hevc, vp9 or av1.</p> <p>Vp9 is required for a transparent background. The synthesis speed is slower with the vp9 codec because vp9 encoding is slower.</p> <p>This property is optional, and the default value is hevc.</p>
avatarConfig.bitrateKbps	<p>The bitrate for output video, which is integer value, with unit kbps.</p> <p>This property is optional, and the default value is 2000.</p>
avatarConfig.videoCrop	<p>This property lets you crop the video output, which means to output a rectangle subarea of the original video. This property has two fields, which define the top-left vertex and bottom-right vertex of the rectangle.</p> <p>This property is optional, and the default behavior is to output the full video.</p>
avatarConfig.videoCrop.topLeft	<p>The top-left vertex of the rectangle for video crop. This property has two fields x and y, to define the horizontal and vertical position of the vertex.</p> <p>This property is required when properties.videoCrop is set.</p>
avatarConfig.videoCrop.bottomRight	<p>The bottom-right vertex of the rectangle for video crop. This property has two fields x and y, to define the horizontal and vertical position of the vertex.</p> <p>This property is required when properties.videoCrop is set.</p>
avatarConfig.subtitleType	<p>Type of subtitle for the avatar video file could be <code>external_file</code>, <code>soft_embedded</code>, <code>hard_embedded</code>, or <code>none</code>.</p> <p>This property is optional, and the default value is <code>soft_embedded</code>.</p>
avatarConfig.backgroundImage	<p>Add a background image using the <code>avatarConfig.backgroundImage</code> property. The value of the property should be a URL pointing to the desired image. This property is optional.</p>
avatarConfig.backgroundColor	<p>Background color of the avatar video, which is a string in <code>#RRGGBBAA</code> format. In this string: RR, GG, BB and AA mean the red, green, blue, and alpha channels, with hexadecimal value</p>

Property	Description
	range 00~FF. Alpha channel controls the transparency, with value 00 for transparent, value FF for non-transparent, and value between 00 and FF for semi-transparent.
	This property is optional, and the default value is #FFFFFF (white).
avatarConfig.useBuiltInVoice	A boolean value indicating whether to use the voice sync for avatar as the synthesis voice. This property can only be used when using custom avatar trained with voice sync for avatar. When set to <code>true</code> , other voices specified in <code>synthesisConfig</code> or in SSML will be ignored.
	This property is optional, and the default value is <code>false</code> .
outputs.result	The location of the batch synthesis result file, which is a video file containing the synthesized avatar.
	This property is read-only.
properties.DurationInMilliseconds	The video output duration in milliseconds.
	This property is read-only.

Batch synthesis job properties

The following table describes the batch synthesis job properties.

[Expand table](#)

Property	Description
createdDateTime	The date and time when the batch synthesis job was created.
	This property is read-only.
description	The description of the batch synthesis.
	This property is optional.
ID	The batch synthesis job ID.
	This property is read-only.
lastActionDateTime	The most recent date and time when the status property value changed.

Property	Description
	This property is read-only.
properties	A defined set of optional batch synthesis configuration settings.
properties.destinationContainerUrl	The batch synthesis results can be stored in a writable Azure Blob Storage container. If you don't specify a container URI with shared access signatures (SAS) token, the Speech service stores the results in a container managed by Microsoft. SAS with stored access policies isn't supported. When the synthesis job is deleted, the result data is also deleted.
	This property is required when generating multiple videos in one job. For single video generation, this property is optional.
	This property is not included in the response when you get the synthesis job.
properties.destinationPath	The prefix path for storing batch synthesis results. If no prefix path is provided, a system-generated path will be used.
	This property is optional and can only be set when the <code>destinationContainerUrl</code> property is specified.
properties.timeToLiveInHours	A duration in hours after the synthesis job is created, when the synthesis results will be automatically deleted. The maximum time to live is 744 hours. The date and time of automatic deletion for synthesis jobs with a status of "Succeeded" or "Failed" is calculated as the sum of the <code>lastActionDateTime</code> and <code>timeToLive</code> properties.
	Otherwise, you can call the delete synthesis method to remove the job sooner.
status	The batch synthesis processing status.
	The status should progress from "NotStarted" to "Running", and finally to either "Succeeded" or "Failed".
	This property is read-only.

Text to speech properties

The following table describes the text to speech properties.

[] Expand table

Property	Description
customVoices	<p>A custom voice is associated with a name and its deployment ID, like this:</p> <pre>"customVoices": {"your-custom-voice-name": "502ac834-6537-4bc3-9fd6-140114daa66d"}</pre> <p>You can use the voice name in your <code>synthesisConfig.voice</code> when <code>inputKind</code> is set to "PlainText", or within SSML text of inputs when <code>inputKind</code> is set to "SSML".</p> <p>This property is required to use a custom voice. If you try to use a custom voice that isn't defined here, the service returns an error.</p>
inputs	<p>The plain text or SSML to be synthesized.</p> <p>When the <code>inputKind</code> is set to "PlainText", provide plain text as shown here:</p> <pre>"inputs": [{"content": "The rainbow has seven colors."}].</pre> <p>When the <code>inputKind</code> is set to "SSML", provide text in the Speech Synthesis Markup Language (SSML) as shown here:</p> <pre>"inputs": [{"content": "<speak version='1.0' xml:lang='en-US'><voice xml:lang='en-US' xml:gender='Female' name='en-US-AvaMultilingualNeural'>The rainbow has seven colors.</voice></speak>"}].</pre> <p>Include up to 1,000 text objects if you want multiple video output files. Here's example input text that should be synthesized to two video output files:</p> <pre>"inputs": [{"content": "synthesize this to a file"}, {"content": "synthesize this to another file"}].</pre> <p>To generate multiple videos, the output must be stored in an Azure Blob Storage container by specifying <code>properties.destinationContainerUrl</code>.</p> <p>You don't need separate text inputs for new paragraphs. Within any of the (up to 1,000) text inputs, you can specify new paragraphs using the "\r\n" (newline) string. Here's example input text with two paragraphs that should be synthesized to the same audio output file:</p> <pre>"inputs": [{"content": "synthesize this to a file\r\nsynthesize this to another paragraph in the same file"}].</pre> <p>This property is required when you create a new batch synthesis job. This property isn't included in the response when you get the synthesis job.</p>
properties.billingDetails	<p>The number of words that were processed and billed by <code>customNeural</code> (custom voice) versus <code>neural</code> (standard voice).</p> <p>This property is read-only.</p>
synthesisConfig	<p>The configuration settings to use for batch synthesis of plain text.</p> <p>This property is only applicable when <code>inputKind</code> is set to "PlainText".</p>

Property	Description
synthesisConfig.pitch	<p>The pitch of the audio output.</p> <p>For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p> <p>This optional property is only applicable when inputKind is set to "PlainText".</p>
synthesisConfig.rate	<p>The rate of the audio output.</p> <p>For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p> <p>This optional property is only applicable when inputKind is set to "PlainText".</p>
synthesisConfig.style	<p>For some voices, you can adjust the speaking style to express different emotions like cheerfulness, empathy, and calm. You can optimize the voice for different scenarios like customer service, newscast, and voice assistant.</p> <p>For information about the available styles per voice, see voice styles and roles.</p> <p>This optional property is only applicable when inputKind is set to "PlainText".</p>
synthesisConfig.voice	<p>The voice that speaks the audio output.</p> <p>For information about the available standard voices, see language and voice support. To use a custom voice, you must specify a valid custom voice and deployment ID mapping in the customVoices property.</p> <p>This property is required when inputKind is set to "PlainText".</p>
synthesisConfig.volume	<p>The volume of the audio output.</p> <p>For information about the accepted values, see the adjust prosody table in the Speech Synthesis Markup Language (SSML) documentation. Invalid values are ignored.</p> <p>This optional property is only applicable when inputKind is set to "PlainText".</p>
inputKind	<p>Indicates whether the inputs text property should be plain text or SSML. The possible case-insensitive values are "PlainText" and "SSML". When the inputKind is set to "PlainText", you must also set the synthesisConfig voice property.</p> <p>This property is required.</p>

Edit the background

The avatar batch synthesis API currently doesn't support setting background videos; it only supports static background images. But if you want to add a background for your video during post-production, you can generate videos with a transparent background.

To set a static background image, use the `avatarConfig.backgroundImage` property and specify a URL pointing to the desired image. Additionally, you can set the background color of the avatar video using the `avatarConfig.backgroundColor` property.

To generate a transparent background video, you must set the following properties to the required values in the batch synthesis request:

 Expand table

Property	Required values for background transparency
<code>properties.videoFormat</code>	<code>webm</code>
<code>properties.videoCodec</code>	<code>vp9</code>
<code>properties.backgroundColor</code>	<code>#00000000</code> (or <code>transparent</code>)

Clipchamp is one example of a video editing tool that supports the transparent background video generated by the batch synthesis API.

Some video editing software doesn't support the `webm` format directly and only supports `.mov` format transparent background video input, like Adobe Premiere Pro. In such cases, you first need to convert the video format from `webm` to `.mov` with a tool like FFmpeg.

FFmpeg command line:

Bash

```
ffmpeg -vcodec libvpx-vp9 -i <input.webm> -vcodec png -pix_fmt rgba metadata:s:v:0  
alpha_mode="1" <output.mov>
```

FFmpeg can be downloaded from ffmpeg.org. Replace `<input.webm>` and `<output.mov>` with your local path and file name in the command line.

Next steps

- [Create an avatar with batch synthesis](#)
- [What is text to speech avatar](#)

Customize text to speech avatar gestures with SSML

The [Speech Synthesis Markup Language \(SSML\)](#) with input text determines the structure, content, and other characteristics of the text to speech output. Most SSML tags also work in text to speech avatar. Furthermore, text to speech avatar batch mode provides avatar gesture insertion by using the SSML bookmark element with the format `<bookmark mark='gesture.*' />`.

A gesture starts at the insertion point in time. If the gesture takes more time than the audio, the gesture is cut at the point in time when the audio is finished.

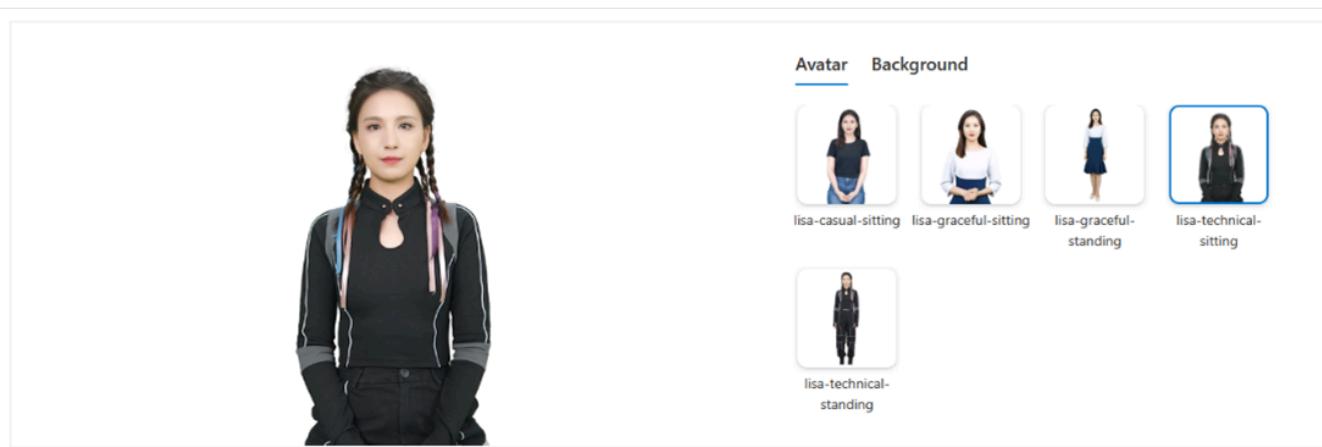
Bookmark example

The following example shows how to insert a gesture in the text to speech avatar batch synthesis with SSML.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
<voice name="en-US-AvaMultilingualNeural">
Hello <bookmark mark='gesture.wave-left-1' />, my name is Ava, nice to meet you!
</voice>
</speak>
```

In this example, the avatar starts waving their hand at the left after the word "Hello".



Preview video

Play audio Insert break Insert gesture Speaking speed Switch to SSML

Language English (United States)

Hello.

wave-left-1 wave-left-2

show-left-1 show-left-2

🔍

ⓘ Note

Gesture feature isn't currently supported when a voice sync for avatar is selected in a custom text to speech avatar.

Supported standard avatar characters, styles, and gestures

The full list of standard avatar supported gestures provided here can also be found in the text to speech avatar portal.

-expand Expand table

Characters	Styles	Gestures
Harry	business	123 calm-down come-on five-star-reviews good hello introduce invite

Characters	Styles	Gestures
		thanks welcome
Harry	casual	123 come-on five-star-reviews gong-xi-fa-cai good happy-new-year hello please welcome
Harry	youthful	123 come-on down five-star good hello invite show-right-up-down welcome
Jeff	business	123 come-on five-star-reviews hands-up here meddle please2 show silence thanks
Jeff	formal	123 come-on five-star-reviews lift please silence thanks very-good
Lisa	casual-sitting	numeric1-left-1 numeric2-left-1 numeric3-left-1 thumbsup-left-1 show-front-1 show-front-2

Characters	Styles	Gestures
		show-front-3 show-front-4 show-front-5 think-twice-1 show-front-6 show-front-7 show-front-8 show-front-9
Lisa	graceful-sitting	wave-left-1 wave-left-2 thumbsup-left show-left-1 show-left-2 show-left-3 show-left-4 show-left-5 show-right-1 show-right-2 show-right-3 show-right-4 show-right-5
Lisa	graceful-standing	
Lisa	technical-sitting	wave-left-1 wave-left-2 show-left-1 show-left-2 point-left-1 point-left-2 point-left-3 point-left-4 point-left-5 point-left-6 show-right-1 show-right-2 show-right-3 point-right-1 point-right-2 point-right-3 point-right-4 point-right-5 point-right-6
Lisa	technical-standing	
Lori	casual	123-left a-little

Characters	Styles	Gestures
		beg calm-down come-on five-star-reviews good hello open please thanks
Lori	graceful	123-left applaud come-on introduce nod please show-left show-right thanks welcome
Lori	formal	123 come-on come-on-left down five-star good hands-triangle hands-up hi hopeful thanks
Max	business	a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-right good-01 good-02 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left lower-left number-one

Characters	Styles	Gestures
		press-both-hands-down-1 press-both-hands-down-2 push-forward raise-ones-hand right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left thanks the-front top-middle-and-bottom-left top-middle-and-bottom-right upper-left upper-right welcome
Max	casual	a-little-bit applaud click-the-link display-number encourage-1 encourage-2 five-star-praise front-left good-1 good-2 hello introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 introduction-to-products-4 left length nodding number-one press-both-hands-down raise-ones-hand right right-front shrug-ones-shoulders slide-from-left-to-right slide-to-the-left thanks the-front upper-left

Characters	Styles	Gestures
		upper-right welcome
Max	formal	a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left lower-left lower-right press-both-hands-down push-forward right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left the-front top-middle-and-bottom-right upper-left upper-right
Meg	formal	a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 hands-forward introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left number-one press-both-hands-down-1

Characters	Styles	Gestures
		press-both-hands-down-2 right say-hi shrug-ones-shoulders slide-from-left-to-right the-front upper-left upper-right
Meg	casual	a-little-bit click-the-link cross-hand display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 handclap introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left length lower-left lower-right number-one press-both-hands-down right say-hi shrug-ones-shoulders slide-from-right-to-left slide-to-the-left spread-hands the-front top-middle-and-bottom-left top-middle-and-bottom-right upper-left upper-right
Meg	business	a-little-bit encourage-1 encourage-2 five-star-praise front-left front-right

Characters	Styles	Gestures
		good-1 good-2 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left length number-one press-both-hands-down-1 press-both-hands-down-2 raise-ones-hand right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left spread-hands thanks the-front upper-left

All styles except `lisa-graceful-sitting`, `lisa-graceful-standing`, `lisa-technical-sitting`, and `lisa-technical-standing` are supported via the real-time text to speech API. Gestures are only supported with the batch synthesis API and aren't supported via the real-time API.

Next steps

- [What is text to speech avatar](#)
- [Real-time synthesis](#)
- [Use batch synthesis for text to speech avatar](#)

Last updated on 11/09/2025

Supported standard text to speech avatar

This article contains the full list of standard avatars with their preview images.

- Standard video avatars
- Standard photo avatars

Standard video avatars

Avatars created from video typically include distinct body parts and can display various styles based on different clothing or poses. Many avatars in this group also allow users to add gestures when processing a batch of videos.

[] Expand table

Characters	Styles	Preview Image	Gestures
Harry	business		123 calm-down come-on five-star-reviews good hello introduce invite thanks welcome

Characters	Styles	Preview Image	Gestures
Harry	casual		123 come-on five-star-reviews good happy-new-year hello please welcome
Harry	youthful		123 come-on down five-star good hello invite show-right-up-down welcome

Characters	Styles	Preview Image	Gestures
Jeff	business		123 come-on five-star-reviews hands-up here meddle please2 show silence thanks
Jeff	formal		123 come-on five-star-reviews lift please silence thanks very-good

Characters	Styles	Preview Image	Gestures
Lisa	casual-sitting	 A woman with long dark hair, wearing a black short-sleeved top and blue jeans, sitting with her hands clasped in her lap.	numeric1-left-1 numeric2-left-1 numeric3-left-1 thumbsup-left-1 show-front-1 show-front-2 show-front-3 show-front-4 show-front-5 think-twice-1 show-front-6 show-front-7 show-front-8 show-front-9
Lisa	graceful-sitting	 A woman with long dark hair, wearing a white long-sleeved blouse over a dark blue skirt, sitting with her hands clasped in front of her.	wave-left-1 wave-left-2 thumbsup-left show-left-1 show-left-2 show-left-3 show-left-4 show-left-5 show-right-1 show-right-2 show-right-3 show-right-4 show-right-5

Characters

Styles

Preview Image

Gestures

Lisa

graceful-
standing



Characters	Styles	Preview Image	Gestures
Lisa	technical-sitting	A photograph of a woman named Lisa sitting down. She is wearing a black long-sleeved top with white stripes on the cuffs and hem, and black pants with white stripes on the sides. She has two braids and is wearing a backpack.	wave-left-1 wave-left-2 show-left-1 show-left-2 point-left-1 point-left-2 point-left-3 point-left-4 point-left-5 point-left-6 show-right-1 show-right-2 show-right-3 point-right-1 point-right-2 point-right-3 point-right-4 point-right-5 point-right-6
Lisa	technical-standing	A photograph of the same woman Lisa standing upright. She is wearing the same black outfit with white stripes. Her hands are at her sides.	

Characters	Styles	Preview Image	Gestures
Lori	casual	 A photograph of a woman from the waist up. She has dark hair pulled back and is wearing a bright yellow short-sleeved ribbed top over a white collared shirt, paired with blue jeans.	123-left a-little beg calm-down come-on five-star-reviews good hello open please thanks
Lori	graceful	 A photograph of the same woman from the waist up, now wearing a light blue wrap-style dress with a white top underneath. Her hands are clasped in front of her.	123-left applaud come-on introduce nod please show-left show-right thanks welcome

Characters	Styles	Preview Image	Gestures
Lori	formal		123 come-on come-on-left down five-star good hands-triangle hands-up hi hopeful thanks
Max	business		a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-right good-01 good-02 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left lower-left number-one press-both-hands-down-1 press-both-hands-down-2 push-forward raise-ones-hand right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left thanks the-front

Characters	Styles	Preview Image	Gestures
			top-middle-and-bottom-left top-middle-and-bottom-right upper-left upper-right welcome
Max	casual		a-little-bit applaud click-the-link display-number encourage-1 encourage-2 five-star-praise front-left good-1 good-2 hello introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 introduction-to-products-4 left length nodding number-one press-both-hands-down raise-ones-hand right right-front shrug-ones-shoulders slide-from-left-to-right slide-to-the-left thanks the-front upper-left upper-right welcome

Characters	Styles	Preview Image	Gestures
Max	formal		a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left lower-left lower-right press-both-hands-down push-forward right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left the-front top-middle-and-bottom-right upper-left upper-right

Characters	Styles	Preview Image	Gestures
Meg	formal		a-little-bit click-the-link display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 hands-forward introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left number-one press-both-hands-down-1 press-both-hands-down-2 right say-hi shrug-ones-shoulders slide-from-left-to-right the-front upper-left upper-right
Meg	casual		a-little-bit click-the-link cross-hand display-number encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 handclap introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left length lower-left lower-right number-one press-both-hands-down

Characters	Styles	Preview Image	Gestures
			right say-hi shrug-ones-shoulders slide-from-right-to-left slide-to-the-left spread-hands the-front top-middle-and-bottom-left top-middle-and-bottom-right upper-left upper-right
Meg	business		a-little-bit encourage-1 encourage-2 five-star-praise front-left front-right good-1 good-2 introduction-to-products-1 introduction-to-products-2 introduction-to-products-3 left length number-one press-both-hands-down-1 press-both-hands-down-2 raise-ones-hand right say-hi shrug-ones-shoulders slide-from-left-to-right slide-to-the-left spread-hands thanks the-front upper-left

All styles except lisa-graceful-sitting, lisa-graceful-standing, lisa-technical-sitting, and lisa-technical-standing are supported via the real-time text to speech API. Gestures are only supported with the batch synthesis API and aren't supported via the real-time API.

Standard photo avatars

[] Expand table

Characters	Preview Image
Adrian	 A portrait of a man with long, wavy brown hair and a beard, smiling at the camera. He is wearing a dark grey t-shirt.
Amara	 A portrait of a woman with short, curly black hair, smiling at the camera. She is wearing a dark blazer over a black top.
Amira	 A portrait of a woman with dark hair pulled back, smiling at the camera. She is wearing a dark blazer over a light-colored top.

Characters**Preview Image**

Anika



Bianca



Camila



Characters**Preview Image**

Carlos



Clara



Darius



Characters**Preview Image**

Diego



Elise



Farhan

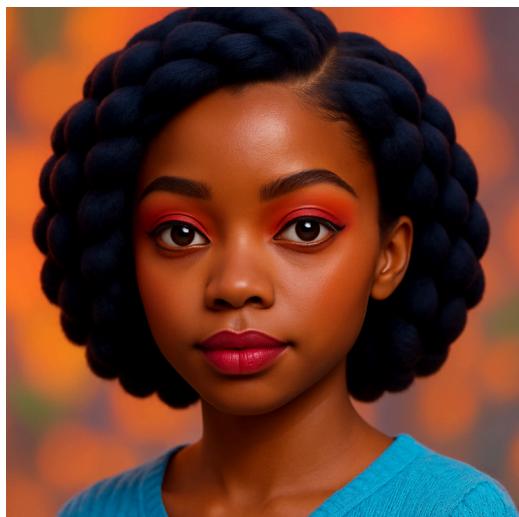


Characters**Preview Image**

Faris



Gabrielle



Hyejin



Characters**Preview Image**

Imran



Isabella



Layla



Characters**Preview Image**

Liwei



Ling



Marcus



Characters**Preview Image**

Matteo



Rahul



Rana



Characters**Preview Image**

Ren



Riya



Sakura



Characters**Preview Image**

Simone



Zayd



Zoe



What is custom text to speech avatar?

Custom text to speech avatar allows you to create a customized, one-of-a-kind synthetic talking avatar for your application. With custom text to speech avatar, you can build a unique and natural-looking avatar for your product or brand by providing video recording data of your selected actors. The avatar is even more realistic if you also use a [professional voice](#) or [voice sync for avatar](#) for the same actor.

Important

Custom text to speech avatar access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

How does it work?

Creating a custom text to speech avatar requires at least 10 minutes of video recording of the avatar talent as training data, and you must first get consent from the actor talent.

The custom avatar model can support:

- Video generation via the [batch synthesis API](#).
- Live chat via the [streaming synthesis API](#).

Before you get started, here are some considerations:

Your use case: Do you want to use the avatar to create video content such as training material or a product introduction? Do you want to use the avatar as a virtual salesperson in a real-time conversation with your customers? There are some recording requirements for different use cases.

The look of the avatar: The custom text to speech avatar looks the same as the avatar talent in the training data, and we don't support customizing the appearance of the avatar model, such as clothes, hairstyle, etc. So if your application requires multiple styles of the same avatar, you should prepare training data for each style, as each style of an avatar is considered as a single avatar model.

The voice of the avatar: The custom text to speech avatar can work with standard voice, professional voice, and voice sync for avatar.

- Voice sync for avatar: A synthetic voice resembling the avatar talent's voice is trained alongside the custom avatar utilizing audio from the training video.

- Professional voice: Fine-tune a professional voice with more training data, providing a premium voice experience for your avatar, including natural conversations, multi-style, and multilingual support.

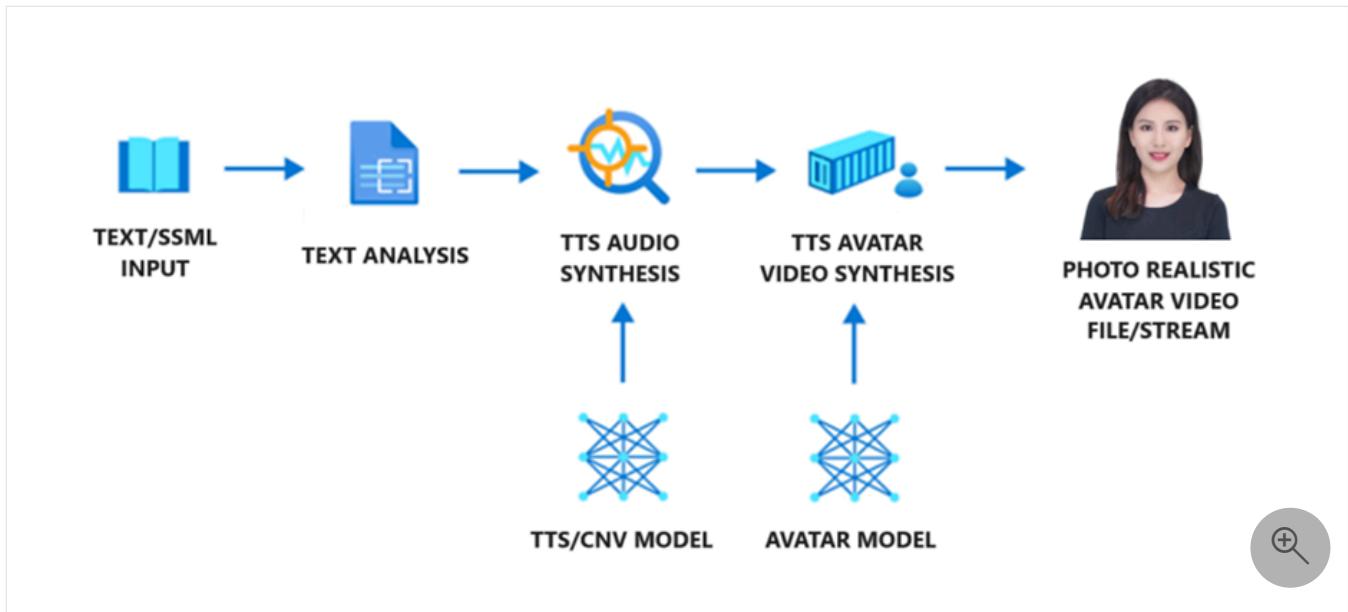
Here's an overview of the steps to create a custom text to speech avatar:

- 1. Get consent video.** Obtain a video recording of the talent reading a consent statement. They must consent to the usage of their image and voice data to train a custom text to speech avatar model and a synthetic version of their voice.
- 2. Prepare training data.** Ensure that the video recording is in the right format. It's a good idea to shoot the video recording in a professional-quality video shooting studio to get a clean background image. The quality of the resulting avatar heavily depends on the recorded video used for training. Factors like speaking rate, body posture, facial expression, hand gestures, consistency in the actor's position, and lighting of the video recording are essential to create an engaging custom text to speech avatar. See [how to prepare training data](#) for more details.
- 3. Train the avatar model.** Once you have the data ready, upload your data to the [custom avatar portal](#) and start to train your model. Consent verification is conducted during the training. Make sure that you have access to the custom text to speech avatar feature before you can create a project.
- 4. Deploy and use your avatar model in your applications.**

Components sequence

The custom text to speech avatar model contains three components: text analyzer, the text to speech audio synthesizer, and text to speech avatar video renderer.

- To generate an avatar video file or stream with the avatar model, text is first input into the text analyzer, which provides the output in the form of a phoneme sequence.
- The audio synthesizer synthesizes the speech audio for input text, and these two parts are provided by standard or custom voice models.
- Finally, the text to speech avatar model predicts the image of lip sync with the speech audio, so that the synthetic video is generated.



The text to speech avatar models are trained using deep neural networks based on the recording samples of human videos in different languages. All languages of standard voices and custom voices can be supported.

Available locations

For the current list of regions that support custom avatar training and usage, see the [Speech service regions table](#).

Custom voice and custom text to speech avatar

[Custom voice](#) and custom text to speech avatar are separate features. You can use them independently or together. If you're also creating a professional voice for the actor, the avatar can be highly realistic.

The custom text to speech avatar can work with a standard voice or custom voice as the avatar's voice. For more information, see [Avatar voice and language](#).

There are two kinds of custom voice for a custom avatar:

- **Voice sync for avatar:** When you enable the voice sync for avatar option during custom avatar training, a synthetic voice model using the likeness of the avatar talent is simultaneously trained with the avatar. This voice is exclusively associated with the custom avatar and can't be independently used. For supported regions, see the [Speech service regions table](#).
- **Professional voice:** You can fine-tune a professional voice. [Professional voice fine-tuning](#) and custom text to speech avatar are separate features. You can use them independently or together. If you choose to use them together, you need to apply for [professional voice](#)

[fine-tuning](#) and [custom text to speech avatar](#) separately, and you're charged separately for professional voice fine-tuning and custom text to speech avatar. For more information, see the [pricing page](#). Additionally, if you plan to use [professional voice fine-tuning](#) with a text to speech avatar, you need to deploy or [copy your custom voice model](#) to one of the [avatar supported regions](#).

If you fine-tune a professional voice and want to use it together with the custom avatar, pay attention to the following points:

- Ensure that the custom voice endpoint is created in the same Azure AI Foundry resource as the custom avatar endpoint. As needed, refer to [train your professional voice model](#) to copy the custom voice model to the same Azure AI Foundry resource as the custom avatar endpoint.
- You can see the custom voice option in the voices list of the [avatar content generation page](#) and [live chat voice settings](#).
- If you're using batch synthesis for avatar API, add the `"customVoices"` property to associate the deployment ID of the custom voice model with the voice name in the request. For more information, see the [text to speech properties](#).
- If you're using real-time synthesis for avatar API, refer to our sample code on [GitHub](#) to set the custom voice.

Related content

- [How to create a custom text to speech avatar](#)
- [How to prepare custom text to speech avatar training data](#)
- [Real-time synthesis for live chat avatar](#)
- [Batch synthesis for video creation](#)
- [Transparency note for text to speech](#)

Last updated on 10/24/2025

Record video samples for custom text to speech avatar

08/07/2025

This article shows you how to prepare high-quality video samples for creating a custom text to speech avatar.

Custom text to speech avatar model building requires training on a video recording of a real human speaking. This person is the avatar talent. You must get sufficient consent under all relevant laws and regulations from the avatar talent to create a custom avatar from their talent's image or likeness. To learn about requirements of the consent statement video, see [Get consent file from the avatar talent](#).

Recording environment

Record in a professional video recording studio or well-lit space.

Background requirements

For commercial, multi-scene avatars, use a clean, smooth, solid-colored background. A green screen works best.

If your avatar will only be used in a single scene, you can record in a specific location like your office, but you can't change the background later.

Follow these best practices when using a solid-colored background like a green screen:

- Position the green screen behind the actor. For full-body shots, place a green screen on the floor under the actor's feet. Connect the back and floor green screens seamlessly.
- Keep the green screen flat with uniform color.
- Maintain 0.5-1 meter distance between the actor and background.
- Light the green screen properly to prevent shadows.
- Keep the actor's full outline within the green screen edges.
- Don't let the actor stand too close to the green screen.
- Keep the actor's head and hands within the green screen when speaking.

Lighting requirements

- Use even, bright lighting on the actor's face. Avoid shadows on the face or reflections on glasses and clothing.

- Keep ambient lighting consistent. Turn off projectors, close curtains to avoid daylight changes, and use stable artificial light sources.

Equipment

- Camera: Minimum 1080p resolution and 25 FPS (frames per second).
- Keep lighting and camera positions fixed throughout recording.
- You can use a teleprompter during recording, but make sure it doesn't affect the actor's gaze toward the camera. Provide seating if the avatar needs to be in a sitting position.
- For half-length or seated avatars, provide seating for the actor. Choose an appropriate chair if you don't want it visible in the video.

Appearance of the actor

Custom text to speech avatar doesn't support customization of clothing or appearance. It's essential to carefully design and prepare the avatar's appearance when recording training data. Consider these tips:

[] Expand table

Categories	Dos	Don'ts
Hair	<ul style="list-style-type: none"> - The actor's hair should have a smooth and glossy surface. - Even the actor's bangs or broken hair should have a clear and smooth border. - Choose a hairstyle that is easy to keep consistent during the whole video recording. 	<ul style="list-style-type: none"> - Avoid messy hair or backgrounds showing through the hair. - Don't let hair block the eyes or eyebrows. - Avoid shadows on the face caused by hairstyle. - Avoid hair changes too much during speech and body gesture. For example, the high ponytail of an actor might appear, disappear, and swing during speaking.
Clothing	<ul style="list-style-type: none"> - Pay attention to clothing status and make sure no significant changes on the clothing during speaking. 	<ul style="list-style-type: none"> - Avoid wearing clothing and accessories that are too loose, heavy, or complex, as they might affect the consistency of clothing status during speaking and body gesture. - Avoid wearing clothing that is too similar to the background color or reflective materials like white shirts or translucent materials. - Avoid clothing with obvious lines or items with logos and brand names you don't want to highlight. - Avoid reflective elements such as metal belts, shiny leather shoes, and leather pants.
Face	<ul style="list-style-type: none"> - Ensure the actor's face is clearly visible. 	<ul style="list-style-type: none"> - Avoid face obscured by hair, sunglasses, or accessories.

What video clips to record

You need these types of video clips:

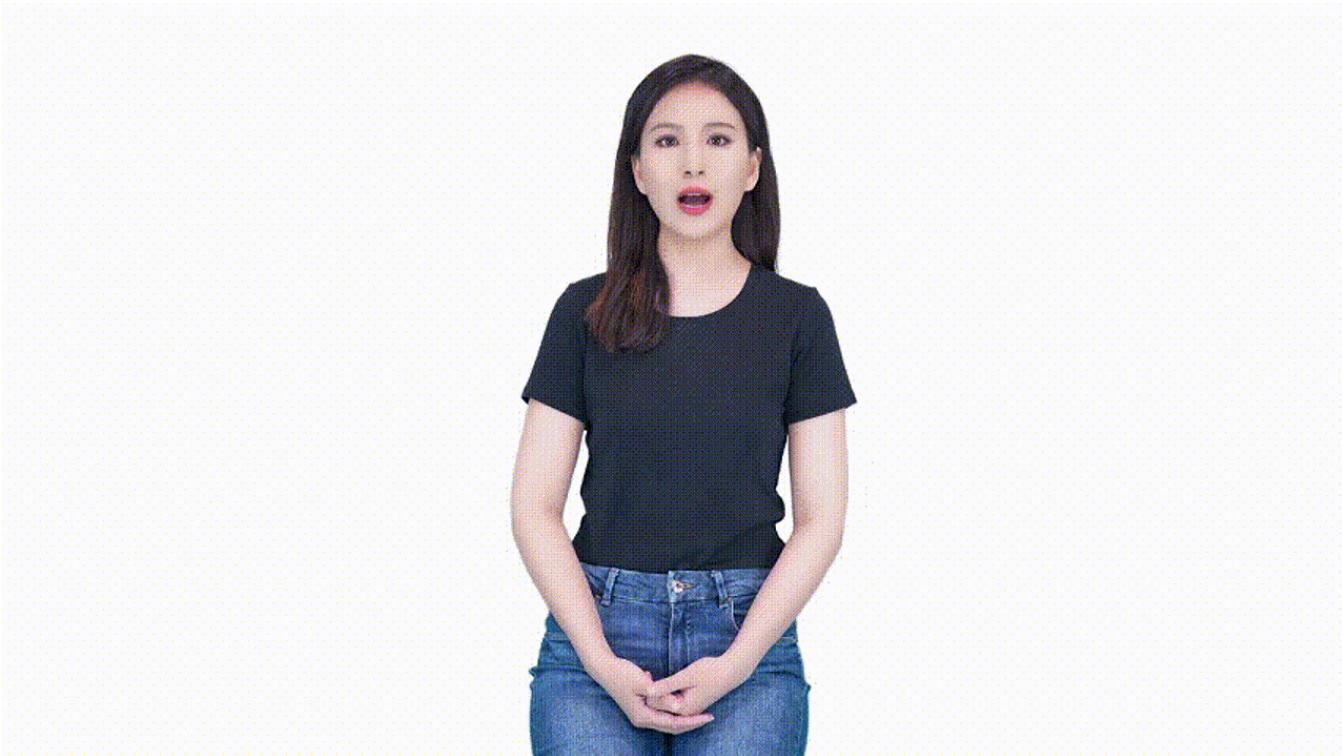
Consent Video (Required) The consent video is required for creating a custom avatar.

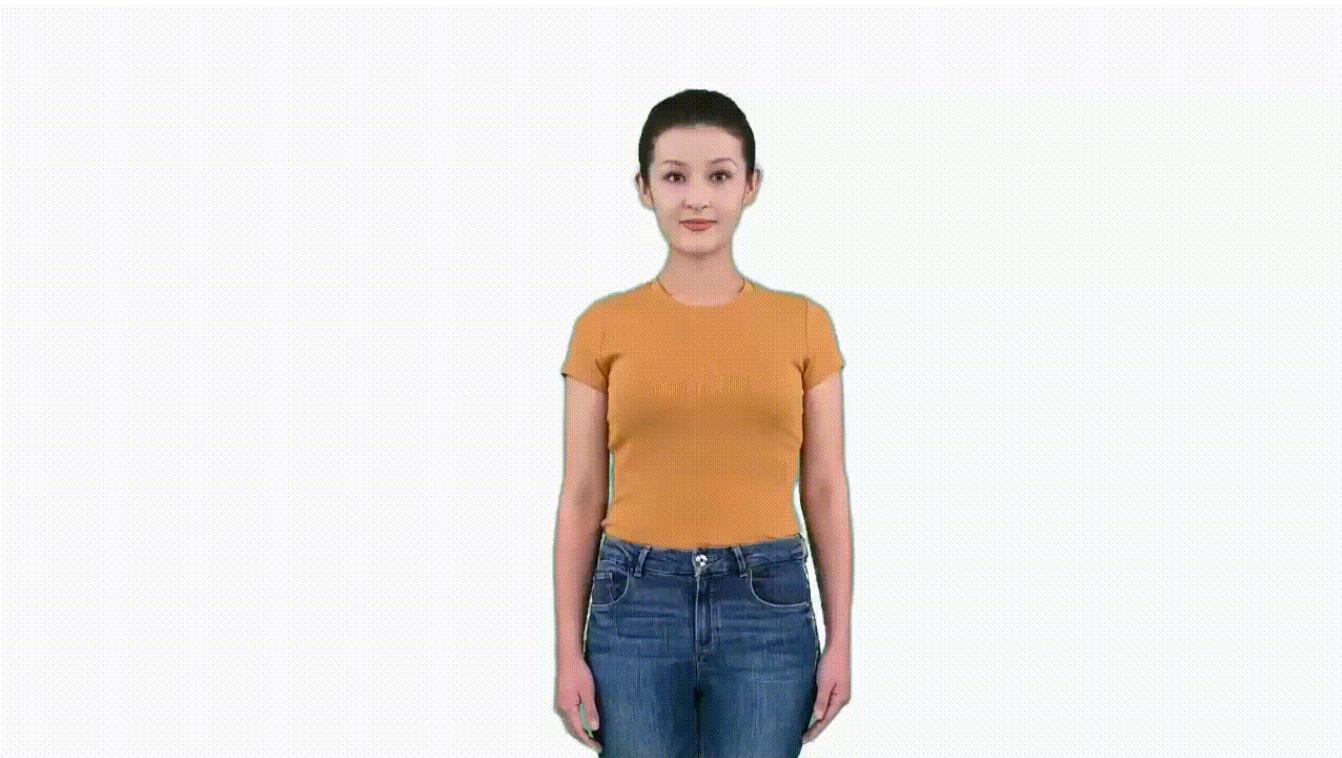
- The consent video must show the same avatar talent speaking and follow the consent statement requirements. Make sure the statement is recorded correctly with each word spoken clearly. You can use any supported language. To learn about consent statement video requirements, see [Get consent file from the avatar talent](#).
- The avatar talent should always face the camera without large movements.
- Record the video in a quiet environment with clear audio at reasonable volume. Keep the signal-to-noise ratio above 20. For voice recording guidance, see the [Recording custom voice samples](#) guide.
- Make sure the actor's head isn't blocked in any frame.
- Keep other objects out of the camera view, including filming equipment and mobile phones.

Status 0 speaking (Required for gestures) The status 0 speaking video clip is required for gestures with the avatar.

- Status 0 represents the posture you can naturally maintain most of the time while speaking. For example, arms crossed in front of the body or hanging naturally at the sides.
- Maintain a front-facing pose. The actor can move slightly to show a relaxed state, like moving the head or shoulder slightly, but don't move the body too much.
- Duration: 3-5 minutes of speaking in status 0.

Samples of status 0 speaking

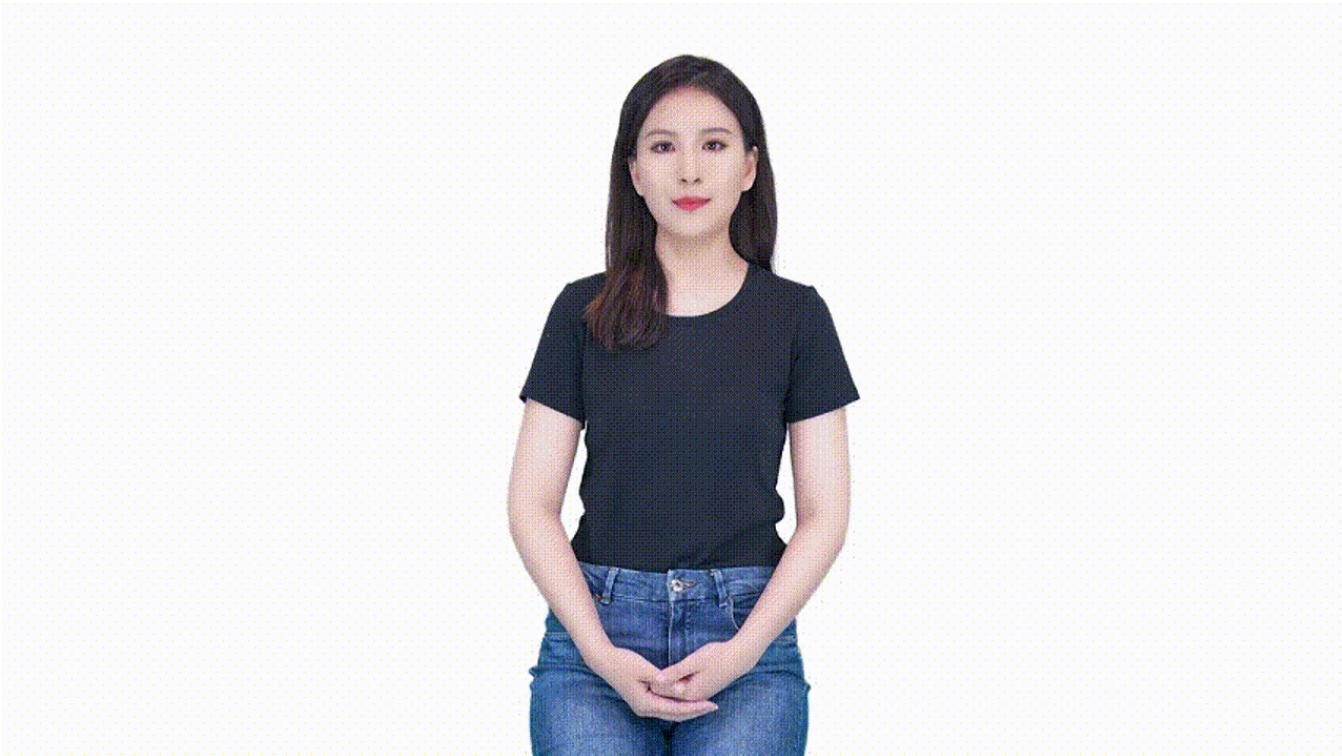


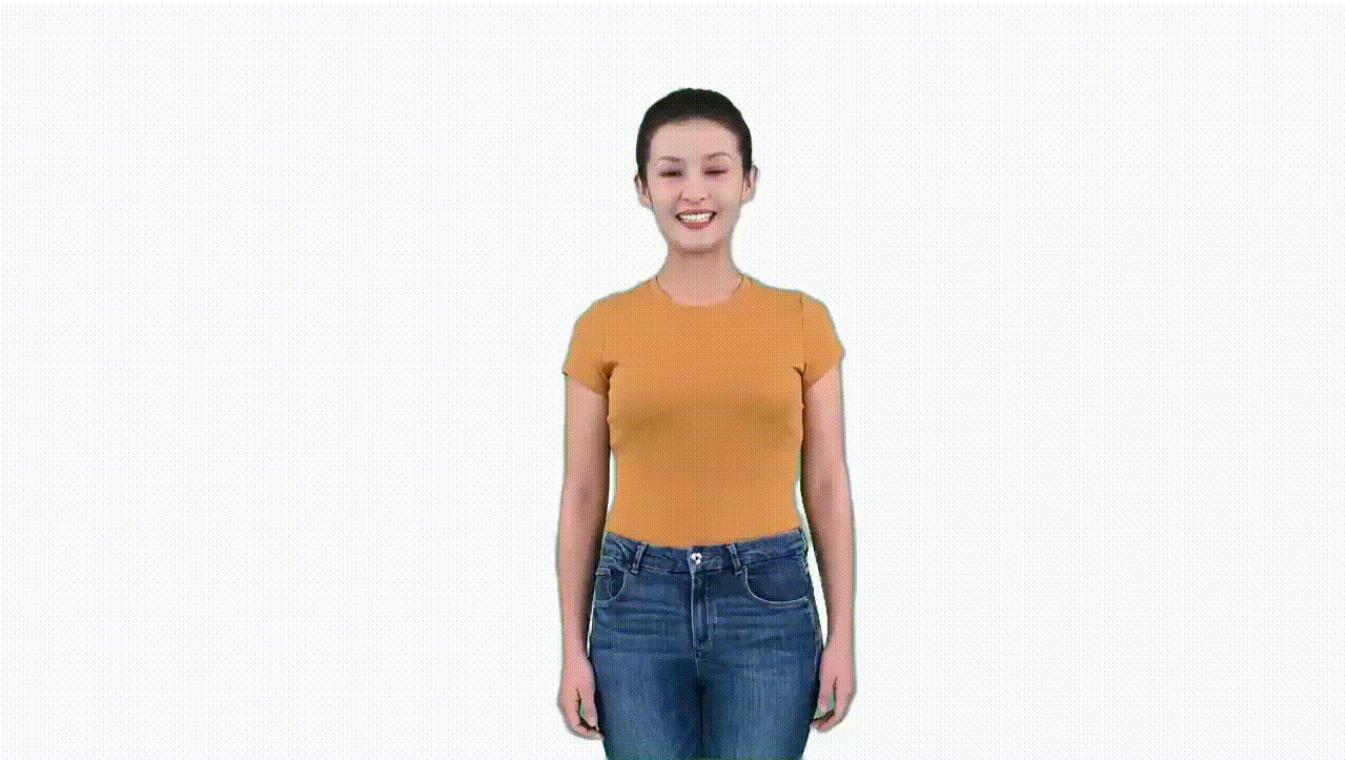


Naturally speaking (Required) The naturally speaking video clip is required for the avatar to speak naturally.

- Actor speaks in status 0 but with natural hand gestures from time to time.
- Hands should start from status 0 and return after making gestures.
- Use natural and common gestures when speaking. Avoid meaningful gestures like pointing, applause, or thumbs up.
- Duration: Minimum 5 minutes, maximum 30 minutes total. At least one 5-minute continuous video recording is required. If recording multiple clips, keep each under 10 minutes.

Samples of natural speaking



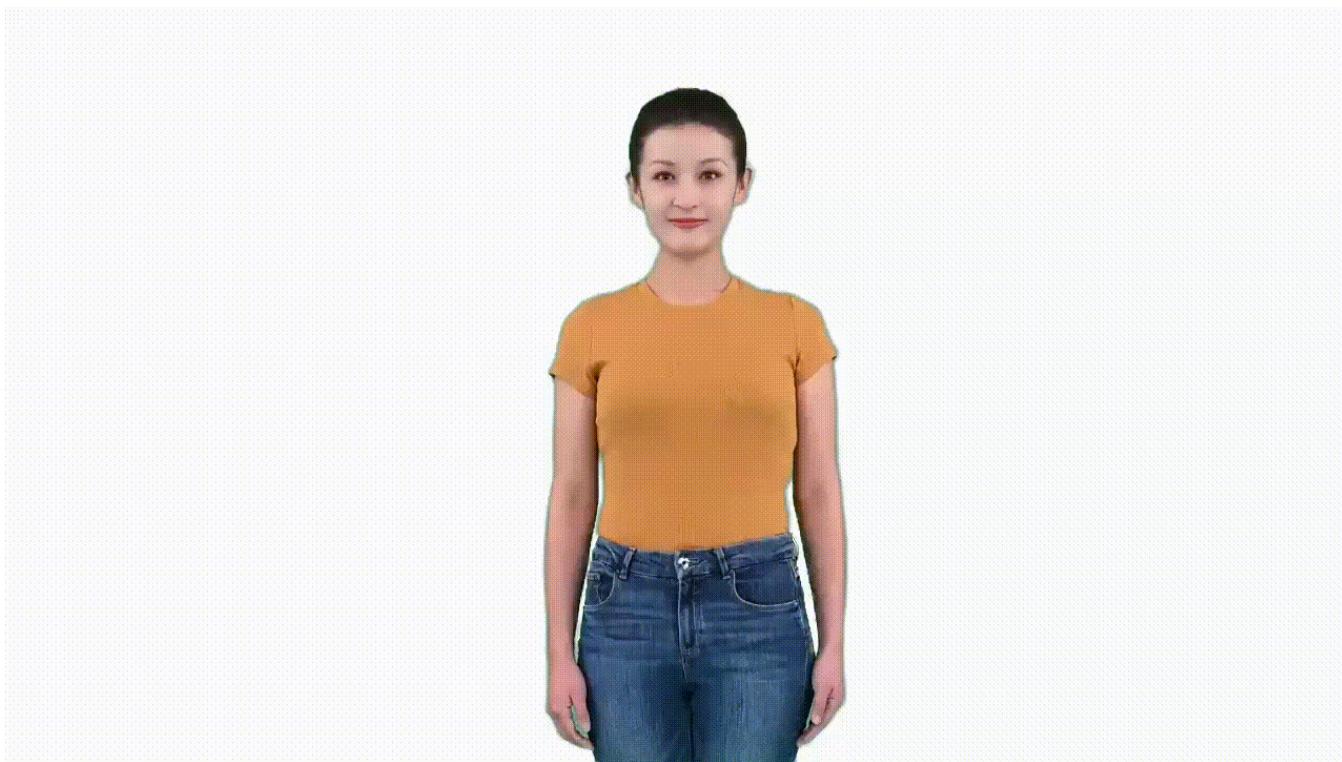


Silent status (Required) The silent status video clip is required. It's important if you build a real-time conversation with the custom avatar. The video clip is used as the main template for both speaking and listening status for a chatbot.

- Maintain status 0, don't speak, but stay relaxed.
- Even while remaining in status 0, don't stay completely still. You can move slightly but not too much. Act like you're waiting.
- Maintain a smile as if listening or waiting patiently.
- Avoid nodding frequently.
- Duration: 1 minute.

Samples of silent status





Gestures (optional)

Gesture video clips are optional. If you need to insert certain gestures in the avatar speaking, follow this guideline to record gesture videos. Gesture insertion is only available for batch mode avatar; real-time avatar doesn't support gesture insertion. Each custom avatar model can support up to 10 gestures.

Gesture tips

- Each gesture clip should be within 10 seconds.
- Gestures should start from status 0 and end with status 0. It's essential that the character maintains the same position as in status 0, which is in the middle of the screen, throughout the gesture. Otherwise, the gesture clip can't be smoothly inserted into the avatar video.
- The gesture clip only captures the body gestures; the actor doesn't have to speak during making gestures.
- Design a list of gestures before recording. Here are some examples:

Samples of gesture

[\[+\] Expand table](#)

Gestures**Samples**

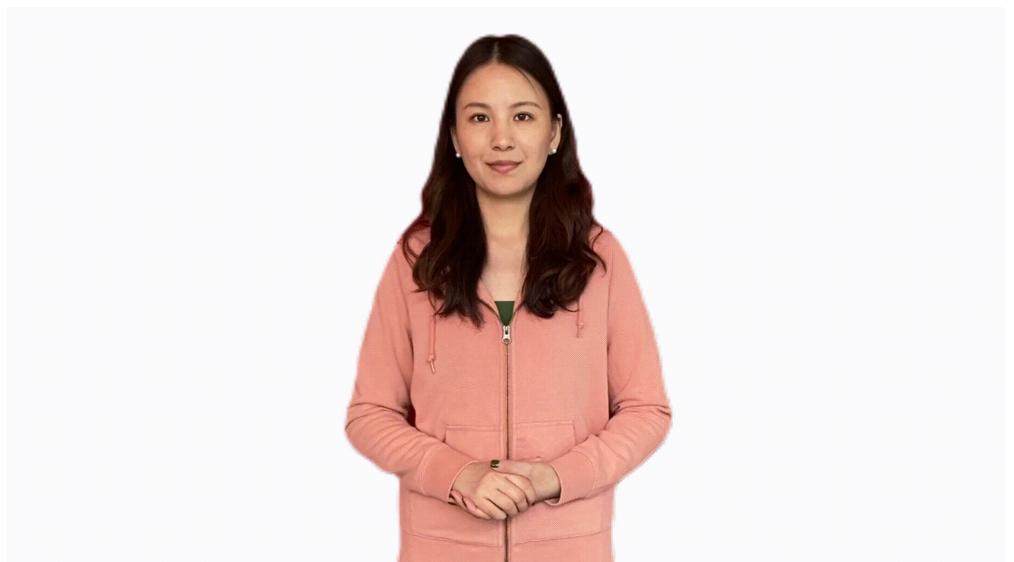
Delivering sell
link/promotion code



Praising the product



Introducing the
product



Gestures	Samples
Displaying the price (number from 1 to 10- fist-number with each hand)	<p>Right hand</p> 
	<p>Left hand</p> 

High-quality avatar models are built from high-quality video recordings, including audio quality. Here are more tips for actor's performance and recording video clips:

 Expand table

Dos	Don'ts
<ul style="list-style-type: none"> - Ensure all video clips are taken in the same conditions. - During the recording process, design the size and display area of the character you need so that the character can be displayed on the screen appropriately. - Actor should be steady during the recording. - Mind facial expressions, which should be suitable for the avatar's use case. For example, look positive and smile if the custom text to speech avatar is used as customer service. Look professionally if the avatar is used 	<ul style="list-style-type: none"> - Don't adjust the camera parameters, focal length, position, angle of view. Don't move the camera; keep the person's position, size, angle, consistent in the camera. - Characters that are too small might lead to a loss of image quality during post-processing. Characters that are too large might cause the screen to overflow during gestures and movements.

Dos	Don'ts
<p>for news reporting.</p> <ul style="list-style-type: none"> - Maintain eye gaze towards the camera, even when using a teleprompter. - Return your body to status 0 when pausing speaking. - Speak on a self-chosen topic, and minor speech mistakes like miss a word or mispronounced are acceptable. If the actor misses a word or mispronounces something, just go back to status 0, pause for 3 seconds, and then continue speaking. - Consciously pause between sentences and paragraphs. When pausing, go back to the status 0 and close your lips. - The audio should be clear and loud enough; bad audio quality impacts training result. - Keep the shooting environment quiet. 	<ul style="list-style-type: none"> - Don't make too long gestures or too much movement for one gesture; for example, actor's hands are always making gestures and forget to go back to status 0. - The actor's movements and gestures must not block the face. - Avoid small movements of the actor like licking lips, touching hair, talking sideways, constant head shaking during speech, and not closing up after speaking. - Avoid background noise; staff should avoid walking and talking during video recording. - Avoid other people's voice recorded during the actor speaking.

How to prepare an interaction video clip

Creating a high-quality interaction video clip is essential if you're building a real-time conversation with a custom avatar. The clip should consist of a question-and-answer format, where a photographer asks a question, and the actor responds. Loop the question-answer pair until the conversation is complete. If you're filming alone, imagine someone else asking the questions during the asking phase.

Here are some tips for each phase:

Asking phase

- Maintain status 0, don't speak, but still feel relaxed.
- Even remaining in status 0, don't keep still. Perform like you're waiting.
- Maintain a smile as if listening or waiting patiently.
- Avoid nodding frequently.
- Length: Each asking slot should last around 3–5 seconds.

Answering phase

- Speak naturally with natural hand gestures from time to time.
- Use natural and common gestures when speaking. Avoid meaningful gestures like pointing, applause, or thumbs up.
- Begin gestures after starting to speak, and stop them before you finish.
- Length: Each answering slot should last around 5 seconds.

Total video length

- Aim for a total video length of 1–5 minutes.

Data requirements

Basic video processing helps improve model training efficiency:

- Keep the character centered on screen with consistent size and position throughout recording. Keep all video processing parameters like brightness and contrast consistent. The output avatar's size, position, brightness, and contrast will directly reflect those in the training data. We don't apply alterations during processing or model building.
- Start and end clips in status 0. Actors should close their mouths, smile, and look ahead. The video should be continuous, not abrupt.

Avatar training video recording file format: .mp4 or .mov.

Resolution: At least 1920x1080.

Frame rate per second: At least 25 FPS.

Related content

- [What is text to speech avatar](#)
- [What is custom text to speech avatar](#)

How to create a custom text to speech avatar

08/07/2025

Getting started with a custom text to speech avatar is a straightforward process. All it takes are a few video clips of your actor. If you'd like to train a [custom voice](#) for the same actor, you can do so separately.

! [Note](#)

Custom avatar access is limited based on eligibility and usage criteria. Request access on the [intake form](#).

Prerequisites

You need an Azure AI Foundry resource in one of the [regions that supports custom avatar training](#). Custom avatar only supports standard (S0) AI Foundry or Speech resources.

You need a video recording of the talent reading a consent statement acknowledging the use of their image and voice. You upload this video when you set up the avatar talent. For more information, see [Add avatar talent consent](#).

You need video recordings of your avatar talent as training data. You upload these videos when you prepare training data. For more information, see [Add training data](#).

Step 1: Start fine-tuning

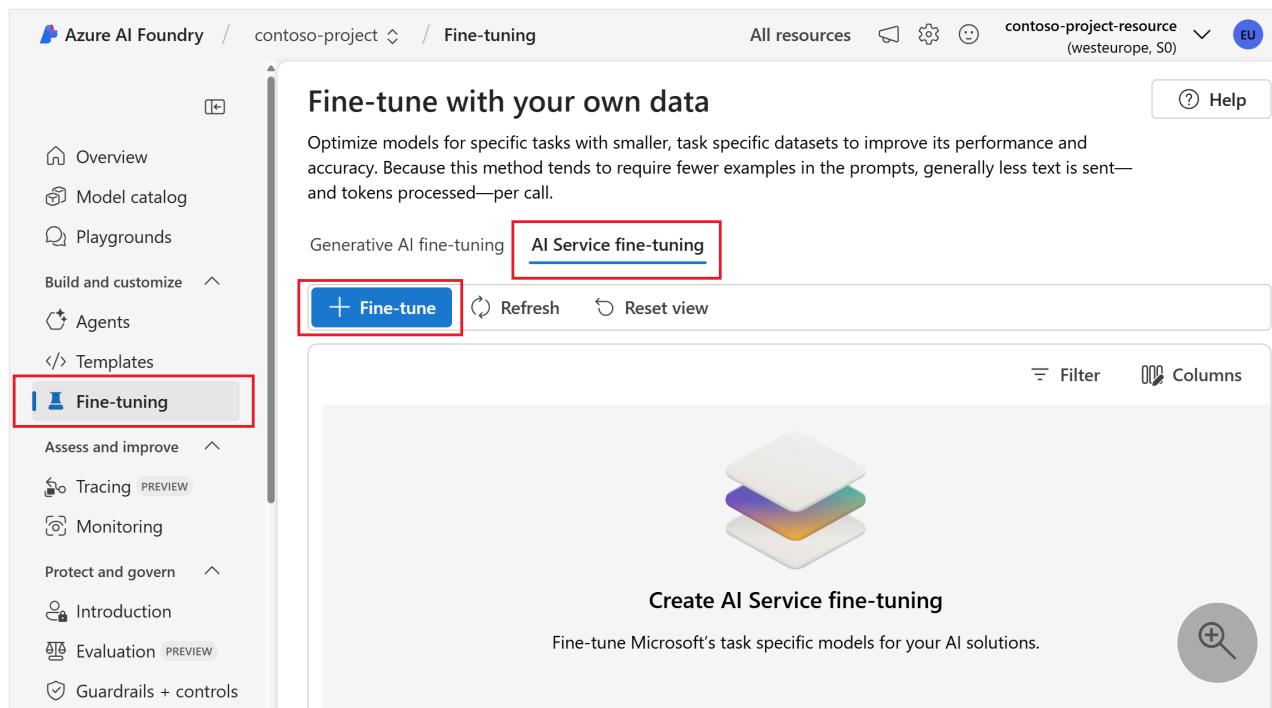
💡 Tip

Don't mix data for different avatars in one fine-tuning workspace. Each avatar must have its own fine-tuning workspace.

To fine-tune a custom avatar, follow these steps:

1. Go to your Azure AI Foundry project in the [Azure AI Foundry portal](#). If you need to create a project, see [Create an Azure AI Foundry project](#).
2. Select **Fine-tuning** from the left pane.

3. Select AI Service fine-tuning > + Fine-tune.



The screenshot shows the Azure AI Foundry interface with the 'contoso-project' project selected. The left sidebar has a 'Fine-tuning' section expanded, with 'AI Service fine-tuning' selected. The main area displays the 'Fine-tune with your own data' page, which includes a brief description of fine-tuning, tabs for 'Generative AI fine-tuning' and 'AI Service fine-tuning' (the latter is selected), and a large blue '+ Fine-tune' button. The '+ Fine-tune' button is highlighted with a red box.

4. In the wizard, select **Custom avatar (text to speech avatar fine-tuning)**.

5. Select **Next**.

6. Follow the instructions provided by the wizard to create your fine-tuning workspace.

Step 2: Add avatar talent consent

An avatar talent is an individual or target actor whose video of speaking is recorded and used to create neural avatar models. You must obtain sufficient consent under all relevant laws and regulations from the avatar talent to use their video to create the custom text to speech avatar.

You must provide a video file with a recorded statement from your avatar talent, acknowledging the use of their image and voice. Microsoft verifies that the content in the recording matches the predefined script provided by Microsoft. Microsoft compares the face of the avatar talent in the recorded video statement file with randomized videos from the training datasets to ensure that the avatar talent in video recordings and the avatar talent in the statement video file are from the same person.

- If you want to create a voice sync for avatar during avatar training, a custom voice resembling your avatar is created alongside the custom avatar. The voice is used exclusively with the specified avatar. Your consent statement must include both the custom avatar and the voice sync for avatar. For an example of the consent statement for custom avatar with voice sync, see the [verbal-statement-voice-sync-for-avatar-all-locales.txt](#) file in the [Azure-Samples/cognitive-services-speech-sdk](#) GitHub repository.

- If you don't create a voice sync for avatar, only the custom avatar is trained, and your consent statement must reflect this scope. For an example of the consent statement for custom avatar only, see the [verbal-statement-all-locales.txt](#) file in the [Azure-Samples/cognitive-services-speech-sdk](#) GitHub repository.

For more information about recording the consent video, see [How to record video samples](#) and [Disclosure for avatar talent](#).

To add an avatar talent profile and upload their consent statement in your project, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom avatar fine-tuning task (by model name) that you [started as described in the previous section](#).
4. Select **Set up avatar talent > Upload consent video**.
5. On the **Upload consent video** page, follow the instructions to upload the avatar talent consent video you recorded beforehand.
 - Select the avatar type to build. Build a voice sync for avatar, which sounds like your avatar talent together with the avatar model, or build avatar without the voice sync for avatar. The option to build a voice sync for avatar is only available in the Southeast Asia, West Europe, and West US 2 regions.
 - Select the speaking language of the verbal consent statement recorded by the avatar talent.
 - Enter the avatar talent name and your company name in the same language as the recorded statement.
 - The avatar talent name must be the name of the person who recorded the consent statement.
 - The company name must match the company name that was spoken in the recorded statement.
 - You can choose to upload your data from local files, or from a shared storage with Azure Blob.
6. Select local files from your computer or enter the Azure Blob storage URL where your data is stored.
7. Select **Next**.
8. Review the upload details, and select **Upload**.

After the avatar talent consent upload is successful, you can proceed to train your custom avatar model.

Step 3: Add training data

The Speech service uses your training data to create a unique avatar tuned to match the look of the person in the recordings. After you train the avatar model, you can start synthesizing avatar videos or use it for live chats in your applications.

All data you upload must meet the requirements for the data type that you choose. To ensure that the Speech service accurately processes your data, it's important to correctly format your data before upload. To confirm that your data is correctly formatted, see [Data requirements](#).

Upload your data

When you're ready to upload your data, go to the **Prepare training data** tab to add your data.

To upload training data, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom avatar fine-tuning task (by model name) that you [started as described in the previous section](#).
4. Select **Prepare training data > Upload data**.
5. In the **Upload data** wizard, choose a data type and then select **Next**. For more information about the data types (including **Naturally Speaking**, **Silent Status**, **Gesture**, and **Status 0 speaking**), see [what video clips to record](#).
6. Select local files from your computer or enter the Azure Blob storage URL where your data is stored.
7. Select **Next**.
8. Review the upload details, and select **Upload**.

Data files are automatically validated when you select **Upload**. Data validation includes series of checks on the video files to verify their file format, size, and total volume. If there are any errors, fix them and submit again.

After you upload the data, you can check the data overview, which indicates whether you provided enough data to start training.

Step 4: Train your avatar model

Important

All the training data in the project is included in the training. The model quality is highly dependent on the data you provided, and you're responsible for the video quality. Make sure you record the training videos according to the [how to record video samples guide](#).

To create a custom avatar in the Azure AI Foundry portal, follow these steps for one of the following methods:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom avatar fine-tuning task (by model name) that you [started as described in the previous section](#).
4. Select **Train model > + Train model**.
5. Enter a Name to help you identify the model. Choose a name carefully. The model name is used as the avatar name in your synthesis request by the SDK and speech synthesis markup language (SSML) input. Only letters, numbers, hyphens, and underscores are allowed. Use a unique name for each model.

Important

The avatar model name must be unique within the same Speech or AI Services resource.

6. Select **Train** to start training the model.

Training duration varies depending on how much data you use. It normally takes 20-40 compute hours on average to train a custom avatar. Check the [pricing note](#) on how training is charged.

Copy your custom avatar model to another project (optional)

Custom avatar training is currently only available in some regions. After your avatar model is trained in a supported region, you can copy it to an AI Services resource for Speech in another region as needed. For more information, see footnotes in the [regions table](#).

Note

You can only copy the voice sync for avatar model to the regions that support the voice sync for avatar feature, which are the same regions that support personal voice.

To copy your custom avatar model to another project:

1. On the **Train model** tab, select an avatar model that you want to copy, and then select **Copy to project**.
2. Select the subscription, region, AI Services resource for Speech, and project where you want to copy the model to. You must have an AI Services resource for Speech and project in the target region, otherwise you need to create them first.
3. Select **Submit** to copy the model.

Once the model is copied, you see a notification in the Azure AI Foundry portal.

Navigate to the project where you copied the model to deploy the model copy.

Step 5: Deploy and use your avatar model

After you successfully created and trained your avatar model, you deploy it to your endpoint.

To deploy your avatar:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom avatar fine-tuning task (by model name) that you [started as described in the previous section](#).
4. Select **Deploy model > Deploy model**.
5. Select a model that you want to deploy.
6. Select **Deploy** to start the deployment.

Important

When a model is deployed, you pay for continuous up time of the endpoint regardless of your interaction with that endpoint. Check the pricing note on how model deployment is charged. You can delete a deployment when the model isn't in use to reduce spending and conserve resources.

After you deploy your custom avatar, it's available to use in the Azure AI Foundry portal or via API:

- The avatar appears in the avatar list of [text to speech avatar on Azure AI Foundry portal](#).
- The avatar appears in the avatar list of [live chat avatars via Azure AI Foundry portal](#).
- You can call the avatar from the SDK and SSML input by specifying the avatar model name. For more information, see the [avatar properties](#).

Remove a deployment

To remove your deployment, follow these steps:

1. Sign in to the [Azure AI Foundry portal](#).
2. Select **Fine-tuning** from the left pane and then select **AI Service fine-tuning**.
3. Select the custom avatar fine-tuning task (by model name) that you [started as described in the previous section](#).
4. Select the deployment on the **Deploy model** page. The model is actively hosted if the status is "Succeeded".
5. You can select the **Delete deployment** button and confirm the deletion to remove the hosting.

Tip

Once a deployment is removed, you no longer pay for its hosting. Deleting a deployment doesn't cause any deletion of your model. If you want to use the model again, create a new deployment.

Next steps

- [What is text to speech avatar](#)
- [How to record video samples](#)

Text to speech with the audio content creation tool

08/05/2025

You can use the audio content creation tool in [Azure AI Foundry portal](#) or [Speech Studio](#) for text to speech without writing any code.

Tip

Select [Foundry portal](#) or [Speech Studio](#) at the top of this article.

Build highly natural audio content for various scenarios, such as audiobooks, news broadcasts, video narrations, and chat bots. With audio content creation, you can efficiently fine-tune text to speech voices and design customized audio experiences.

The tool is based on [Speech Synthesis Markup Language \(SSML\)](#). It allows you to adjust text to speech output attributes in real-time or batch synthesis, such as voice characters, voice styles, speaking speed, pronunciation, and prosody.

- No-code approach: You can use the audio content creation tool for text to speech synthesis without writing any code. The output audio might be the final deliverable that you want. For example, you can use the output audio for a podcast or a video narration.
- Developer-friendly: You can listen to the output audio and adjust the SSML to improve speech synthesis. Then you can use the [Speech SDK](#) or [Speech CLI](#) to integrate the SSML into your applications.

You have easy access to a broad portfolio of [languages and voices](#). These voices include state-of-the-art standard voices and your custom voice, if you built one.

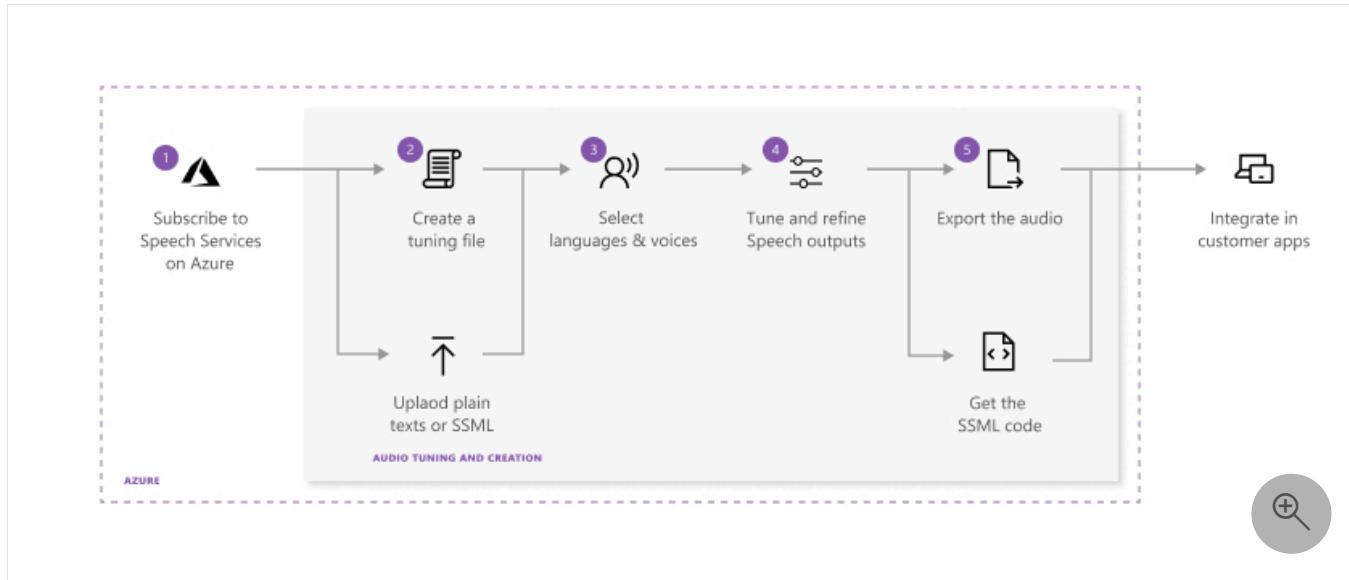
The audio content creation tool is free to access; you pay only for Speech service usage.

Prerequisites

- An active Azure subscription. [Create one for free](#).
- Permission to create resources in your subscription.
- An Azure AI Foundry project. For more information, see [Create an Azure AI Foundry project](#).

Use the audio content creation tool

The following diagram displays the process for fine-tuning the text to speech outputs.



Access the tool

To access the audio content creation tool in Azure AI Foundry, follow these steps:

1. Go to your project in [Azure AI Foundry](#).
2. Select **Playgrounds** from the left pane.
3. In the **Speech** playground tile, select **Try the Speech playground**.
4. Select **Text to speech > Audio content creation**. You might need to scroll to find the tile.

← Speech Playground ▾

/ View code View documentation Fine-tune

All Speech to text Text to speech Speech capabilities by scenario

Video translation

Seamlessly translate and generate videos in multiple languages automatically.

Voice live Preview

Empower your agents with real-time, customized voices, avatars and a whole suite of conversational enhancements, using a single API.

Workflow overview

Once you have access to the tool, follow this general workflow:

1. Create an audio tuning file by using plain text or SSML scripts. Enter or upload your content into audio content creation.
2. Choose the voice and the language for your script content. Audio content creation includes all of the [standard text to speech voices](#). You can use standard voices or a custom voice.

 **Note**

Custom voice access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

3. Select the content you want to preview, and then select **Play** (via the triangle icon) to preview the default synthesis output.

If you make any changes to the text, select the **Stop** icon, and then select **Play** again to regenerate the audio with changed scripts.

Improve the output by adjusting pronunciation, break, pitch, rate, intonation, voice style, and more. For a complete list of options, see [Speech Synthesis Markup Language](#).

4. Save and [export your tuned audio](#).

When you save the tuning track in the system, you can continue to work and iterate on the output. When you're satisfied with the output, you can create an audio creation task with the export feature. You can observe the status of the export task and download the output for use with your apps and products.

Create an audio tuning file

You can get your content into the audio content creation tool in either of two ways:

Option 1: Create a new audio tuning file

1. Select **New > Text file** to create a new audio tuning file.
2. Enter or paste your content into the editing window. The allowable number of characters for each file is 20,000 or fewer. If your script contains more than 20,000 characters, you can use Option 2 to automatically split your content into multiple files.
3. Select **Save**.

Option 2: Upload an audio tuning file

1. Select **Upload > Text file** to import one or more text files. Both plain text and SSML are supported.

If your script file is more than 20,000 characters, split the content by paragraphs, by characters, or by regular expressions.

2. When you upload your text files, make sure that they meet these requirements:

 Expand table

Property	Description
File format	Plain text (.txt) or SSML text (.txt) Zip files aren't supported.
Encoding format	UTF-8
File name	Each file must have a unique name. Duplicate files aren't supported.
Text length	Character limit is 20,000. If your files exceed the limit, split them according to the instructions in the tool.
SSML restrictions	Each SSML file can contain only a single piece of SSML.

Here's a plain text example:

txt

Welcome to use audio content creation to customize audio output for your products.

Here's an SSML example:

XML

```
<speak xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:mstts="http://www.w3.org/2001/mstts" version="1.0" xml:lang="en-US">
    <voice name="en-US-AvaMultilingualNeural">
      Welcome to use audio content creation <break time="10ms" />to customize
      audio output for your products.
    </voice>
</speak>
```

Export tuned audio

After you review your audio output and are satisfied with your tuning and adjustment, you can export the audio.

1. Select **Export** to create an audio creation task.

We recommend **Export to Audio library** to easily store, find, and search audio output in the cloud. You can better integrate with your applications through Azure blob storage. You can also download the audio to your local disk directly.

2. Choose the output format for your tuned audio. The **supported audio formats and sample rates** are listed in the following table:

[+] Expand table

Format	8 kHz sample rate	16 kHz sample rate	24 kHz sample rate	48 kHz sample rate
wav	riff-8khz-16bit-mono-pcm	riff-16khz-16bit-mono-pcm	riff-24khz-16bit-mono-pcm	riff-48khz-16bit-mono-pcm
mp3	N/A	audio-16khz-128kbitrate-mono-mp3	audio-24khz-160kbitrate-mono-mp3	audio-48khz-192kbitrate-mono-mp3

3. To view the status of the task, select the **Task list** tab.

If the task fails, see the detailed information page for a full report.

4. When the task is complete, your audio is available for download on the **Audio library** pane.

5. Select the file you want to download and **Download**.

Now you're ready to use your custom tuned audio in your apps or products.

Related content

- [Speech Synthesis Markup Language \(SSML\)](#)
- [Batch synthesis](#)

What are OpenAI text to speech voices?

Like Azure AI Speech voices, OpenAI text to speech voices deliver high-quality speech synthesis to convert written text into natural sounding spoken audio. This unlocks a wide range of possibilities for immersive and interactive user experiences.

OpenAI text to speech voices are available via two model variants: `Neural1` and `NeuralHD`.

- `Neural1`: Optimized for real-time use cases with the lowest latency, but lower quality than `NeuralHD`.
- `NeuralHD`: Optimized for quality.

Available text to speech voices in Azure AI services

You might ask: If I want to use an OpenAI text to speech voice, should I use it via the Azure OpenAI in Azure AI Foundry Models or via Azure AI Speech? What are the scenarios that guide me to use one or the other?

Each voice model offers distinct features and capabilities, allowing you to choose the one that best suits your specific needs. You want to understand the options and differences between available text to speech voices in Azure AI services.

You can choose from the following text to speech voices in Azure AI services:

- OpenAI text to speech voices in [Azure OpenAI](#). For the current list of supported regions, see the [Speech service regions table](#).
- OpenAI text to speech voices in [Azure AI Speech](#). For the current list of supported regions, see the [Speech service regions table](#).
- Azure AI Speech service [text to speech voices](#). Available in dozens of regions. See the [region list](#).

OpenAI text to speech voices via Azure OpenAI or via Azure AI Speech?

If you want to use OpenAI text to speech voices, you can choose whether to use them via [Azure OpenAI](#) or via [Azure AI Speech](#). You can visit the [Voice Gallery](#) to listen to samples of Azure OpenAI voices or synthesize speech with your own text using the [Audio Content Creation](#). The audio output is identical in both cases, with only a few feature differences between the two services. See the table below for details.

Here's a comparison of features between OpenAI text to speech voices in Azure OpenAI and OpenAI text to speech voices in Azure AI Speech.

[+] [Expand table](#)

Feature	Azure OpenAI (OpenAI voices)	Azure AI Speech (OpenAI voices)	Azure AI Speech voices
Region	North Central US, Sweden Central	North Central US, Sweden Central	Available in dozens of regions. See the region list .
Voice variety	6	12	More than 500
Multilingual voice number	6	12	49
Max multilingual language coverage	57	57	77
Speech Synthesis Markup Language (SSML) support	Not supported	Support for a subset of SSML elements .	Support for the full set of SSML in Azure AI Speech.
Development options	REST API	Speech SDK, Speech CLI, REST API	Speech SDK, Speech CLI, REST API
Deployment option	Cloud only	Cloud only	Cloud, embedded, hybrid, and containers.
Real-time or batch synthesis	Real-time	Real-time	Real-time and batch synthesis
Latency	greater than 500 ms	greater than 500 ms	less than 300 ms
Sample rate of synthesized audio	24 kHz	8, 16, 24, and 48 kHz	8, 16, 24, and 48 kHz
Speech output audio format	opus, mp3, aac, flac	opus, mp3, pcm, truesilk	opus, mp3, pcm, truesilk

There are additional features and capabilities available in Azure AI Speech that aren't available with OpenAI voices. For example:

- OpenAI text to speech voices in Azure AI Speech [only support a subset of SSML elements](#). Azure AI Speech voices support the full set of SSML elements.
- Azure AI Speech supports [word boundary events](#). OpenAI voices don't support word boundary events.

Available OpenAI text to speech voices

The available OpenAI voices in Azure OpenAI are:

- `alloy`
- `echo`
- `fable`
- `onyx`
- `nova`
- `shimmer`

The available OpenAI voices in Azure AI Speech are:

- `en-US-AlloyMultilingualNeural`
- `en-US-EchoMultilingualNeural`
- `en-US-FableMultilingualNeural`
- `en-US-OnyxMultilingualNeural`
- `en-US-NovaMultilingualNeural`
- `en-US-ShimmerMultilingualNeural`
- `en-US-AlloyMultilingualNeuralHD`
- `en-US-EchoMultilingualNeuralHD`
- `en-US-FableMultilingualNeuralHD`
- `en-US-OnyxMultilingualNeuralHD`
- `en-US-NovaMultilingualNeuralHD`
- `en-US-ShimmerMultilingualNeuralHD`

SSML elements supported by OpenAI text to speech voices in Azure AI Speech

The [Speech Synthesis Markup Language \(SSML\)](#) with input text determines the structure, content, and other characteristics of the text to speech output. For example, you can use SSML to define a paragraph, a sentence, a break or a pause, or silence. You can wrap text with event tags such as bookmark or viseme that can be processed later by your application.

The following table outlines the Speech Synthesis Markup Language (SSML) elements supported by OpenAI text to speech voices in Azure AI speech. Only the following subset of SSML tags are supported for OpenAI voices. See [SSML document structure and events](#) for more information.

 Expand table

SSML	Description
element	
name	
<speak>	Encloses the entire content to be spoken. It's the root element of an SSML document.
<voice>	Specifies a voice used for text to speech output.
<sub>	Indicates that the alias attribute's text value should be pronounced instead of the element's enclosed text.
<say-as>	Indicates the content type, such as number or date, of the element's text. All of the <code>interpret-as</code> property values are supported for this element except <code>interpret-as="name"</code> . For example, <code><say-as interpret-as="date" format="dmy">10-12-2016</say-as></code> is supported, but <code><say-as interpret-as="name">ED</say-as></code> isn't supported. For more information, see pronunciation with SSML .
<s>	Denotes sentences.
<lang>	Indicates the default locale for the language that you want the neural voice to speak.
<break>	Use to override the default behavior of breaks or pauses between words.

Related content

- [Try the text to speech quickstart in Azure AI Speech](#)
- [Try the text to speech via Azure OpenAI](#)

Last updated on 10/24/2025

Text to speech FAQ

This article answers commonly asked questions about the text to speech (TTS) capability. If you can't find answers to your questions here, check out [other support options](#).

General

How does the billing work for text to speech?

Text to speech usage is billed per character. Check the definition of billable characters in the [pricing note](#).

What is the rate limit for the text to speech synthesis requests?

The text to speech synthesis rate scales automatically as it receives more requests. A default rate limit is set per speech resource. The rate is adjustable with business justifications and no extra charges are incurred for rate limit increase. Check more details in [Speech service quotas and limits](#).

How would we disclose to the end user that the voice is a synthetic voice?

We recommend that every user should follow our [code of conduct](#) when using the text to speech capability. There are several ways to disclose the synthetic nature of the voice including implicit and explicit byline. Refer to [Disclosure design guidelines](#).

How can I reduce the latency for my voice app?

We provide several tips for you to lower the latency and bring the best performance to your users. See [Lower speech synthesis latency using Speech SDK](#).

What output audio formats does text to speech support?

Azure AI text to speech supports various streaming and non-streaming audio formats, with the commonly used sampling rates. All TTS standard voices are created to support high-fidelity audio outputs with 48 kHz and 24 kHz. The audio can be resampled to support other rates as needed. See [Audio outputs](#).

Can the voice be customized to stress specific words?

Adjusting the emphasis is supported for some voices depending on the locale. See the [emphasis tag](#).

Can we have multiple strength for each emotion, like sad, slightly sad, and so on, in?

Adjusting the style degree is supported for some voices depending on the locale. See the [mstts:express-as tag](#).

Is there a mapping between Viseme IDs and mouth shape?

Yes. See [Get facial position with viseme](#).

Why am I getting HTTP 429 (Too Many Requests) errors when using Text-to-Speech Standard Voice?

The default quota of 200 transactions per second (TPS) for Text-to-Speech (TTS) Standard Voice is designed to accommodate the needs of most customers. In most cases, HTTP 429 errors are not caused by quota limitations, but by insufficient backend service capacity in the selected region for the specified voice. Increasing the quota does not resolve these capacity constraints. To address this issue effectively:

- Use native regions: Deploy the voice in a region where it is natively supported and better resourced (for example, use the Japan region for Japanese voices).
- Select popular voices: Choose a more commonly used voice within your current region to reduce the likelihood of hitting capacity limits.

Audio Content Creation

How can I reference a lexicon file that I created on the Audio Content Creation platform in my code?

First, you can open the lexicon file on the Audio Content Creation and obtain the lexicon file ID, which is located before "?fileKind=CustomLexiconFile" in the file path. For example, if the file

path is

<https://speech.microsoft.com/portal/d391a094f76846acbcd11dc2ba835f4f/audiocontentcreation/file/6cbc2527-8d57-4c1b-b9d9-3ea6d13ca95c?fileKind=CustomLexiconFile>, the lexicon file ID is `6cbc2527-8d57-4c1b-b9d9-3ea6d13ca95c`. Then, switch a file referencing this lexicon to SSML format on the Audio Content Creation. In the SSML file, locate the `<!--ID=FCB` xml node, where you can find the URI of the lexicon file based on the mentioned file ID. Finally, reference the lexicon file URI link using the SSML lexicon element in your code. For instance, if you locate the XML node `<!--ID=FCB5B6FB566-33CA-4B68-BEAF-B013C53B3368;Version=1|{"Files":{ "6cbc2527-8d57-4c1b-b9d9-3ea6d13ca95c":`

`{"FileKind":"CustomLexiconFile","FileSubKind":"CustomLexiconFile","Uri":"https://cvoiceprodwus2.blob.core.windows.net/acc-public-files/d391a094f76846acbcd11dc2ba835f4f/e9a6a5a2-9cef-47f4-b961-d175be75d92f.xml"}}}`, you can obtain the lexicon file URI <https://cvoiceprodwus2.blob.core.windows.net/acc-public-files/d391a094f76846acbcd11dc2ba835f4f/e9a6a5a2-9cef-47f4-b961-d175be75d92f.xml>.

Professional voice fine-tuning

How much data is required for professional voice fine-tuning?

You need training data of at least 300 lines of recordings (or approximately 30 minutes of speech) for professional voice fine-tuning. We recommend 2,000 lines of recordings (or approximately 2-3 hours of speech) to create a voice for production use. For the script selection criteria, see [Record custom voice samples](#).

Can we include duplicate text sentences in the same set of training data?

No. The service will flag the duplicate sentences and just keep the first imported one. For the script selection criteria, see [Record custom voice samples](#).

Can we include multiple styles in the same set of training data?

We recommend that you keep the style consistent in one set of training data. If the styles are different, put them into different training sets. In this case, consider using the multi-style training method of professional voice fine-tuning. For the script selection criteria, see [Record custom voice samples](#).

Does switching styles via SSML work for custom voices?

Switching styles via SSML works for both multi-style standard voices and multi-style custom voices. With multi-style training, you can create a voice that speaks in different styles, and you can also adjust these styles via SSML.

How does cross-lingual voice work with languages that have different pronunciation structure and assembly?

Sentence structure and pronunciation naturally vary across languages such as English and Japanese. Each neural voice is trained with audio data recorded by native speaking voice talent. For [cross lingual](#) voice, we transfer the major features like timbre to sound like the original speaker and preserve the right pronunciation. For example, a cross-lingual voice uses the native way to speak Japanese and still sounds similar (but not exactly) like the original English speaker.

Can I use professional voice fine-tuning to customize pronunciation for my domain?

Professional voice fine-tuning enables you to create a brand voice for your business. You can optimize it for your domain as well. We recommend you include domain-specific samples in your training data for higher naturalness. However, the pronunciation is defined by the Speech service by default. We don't support pronunciation customization with professional voice fine-tuning. If you want to customize pronunciation for your voice, use SSML. See [Pronunciation with Speech Synthesis Markup Language \(SSML\)](#).

After one training can I train my voice again?

You can train again. Each training creates a new voice model. You are charged for each training.

Is the model version the same as the engine version?

No. The model version is different from the engine version. The model version means the version of the training recipe for your model and varies by the features supported and model training time. Azure AI services text to speech engines are updated from time to time to capture the latest language model that defines the pronunciation of the language. After you've trained your voice, you can apply your voice to the new language model by updating to the

latest engine version. When a new engine is available, you're prompted to update your neural voice model. See [Update engine version for your voice model](#).

Can we limit the number of trainings using Azure Policy or other features? Or is there any way to avoid false training?

If you want to limit the permission to training, you can limit the user roles and access. Refer to [Role-based access control for Speech resources](#).

Can Microsoft add a mechanism to prevent unauthorized use or misuse of our voice when it's created?

The voice model can only be used by yourselves using your own token. Microsoft also doesn't use your data. See [Data, privacy, and security](#). You can also request to add watermarks to your voice to protect your model. See [Microsoft Azure Neural TTS introduces the watermark algorithm for synthetic voice identification](#).

Do you have any tips about contracts or negotiation with voice actors?

We have no recommendations on contracts and it's up to the customer and the voice talent to negotiate the terms. However, you should make sure the voice talent understands the capabilities of text to speech, including its potential risks, and provide explicit consent to create a synthetic version of their voice in both the contract and a verbal statement. See [Disclosure for voice talent](#).

Do we need to return the written permission from the voice talent back to Microsoft?

Microsoft doesn't need the written permission, but you must obtain consent from your voice talent. The voice talent will also be required to record the consent statement and it must be uploaded into Speech Studio before training can begin. See [Set up voice talent for professional voice fine-tuning](#).

Next steps

- [Text to speech quickstart](#)
- [What's new](#)

What is speech translation?

09/12/2025

In this article, you learn about the benefits and capabilities of translation with Azure AI Speech. The Speech service supports real-time, multi-language speech to speech and speech to text translation of audio streams.

By using the Speech SDK or Speech CLI, you can give your applications, tools, and devices access to source transcriptions and translation outputs for the provided audio. Interim transcription and translation results are returned as speech is detected, and the final results can be converted into synthesized speech.

For a list of languages supported for speech translation, see [Language and voice support](#).

💡 Tip

Go to the [Speech Studio](#) to quickly test and translate speech into other languages of your choice with low latency.

Core features

The core features of speech translation include:

- [Speech to text translation](#)
- [Speech to speech translation](#)
- [Multi-lingual speech translation](#)
- [Live Interpreter \(preview\)](#)
- [Multiple target languages translation](#)

Speech to text translation

The standard feature offered by the Speech service is the ability to take in an input audio stream in your specified source language, and have it translated and outputted as text in your specified target language.

Speech to speech translation

As a supplement to the above feature, the Speech service also offers the option to read aloud the translated text using our large database of pretrained voices, allowing for a natural output

of the input speech.

Multi-lingual speech translation

Multi-lingual speech translation implements a new level of speech translation technology that unlocks various capabilities, including having no specified input language, handling language switches within the same session, and supporting live streaming translations into English. These features enable a new level of speech translation powers that can be implemented into your products.

- Unspecified input language. Multi-lingual speech translation can receive audio in a wide range of languages, and there's no need to specify what the expected input language is.
- Language switching. Multi-lingual speech translation allows for multiple languages to be spoken during the same session, and have them all translated into the same target language. There's no need to restart a session when the input language changes or any other actions by you.
- Transcription. The service outputs a transcription in the specified target language. Source language transcription isn't available yet.

Some use cases for multi-lingual speech translation include:

- Travel Interpreter. When traveling abroad, multi-lingual speech translation offers the ability to create a solution that allows customers to translate any input audio to and from the local language. This allows them to communicate with the locals and better understand their surroundings.
- Business Meeting. In a meeting with people who speak different languages, multi-lingual speech translation allows the members of the meeting to all communicate with each other naturally as if there was no language barrier.

For a list of the supported input (source) languages, see the [speech to text languages documentation](#). For a list of the supported output (target) languages, see the *Translate to text language* table in the [speech translation languages documentation](#).

For more information on multi-lingual speech translation, see [the speech translation how to guide](#) and [speech translation samples on GitHub ↗](#).

Live interpreter (preview)

Live Interpreter continuously identifies the language being spoken without requiring you to set an input language and delivers low latency speech-to-speech translation in a natural voice that preserves the speaker's style and tone. Live Interpreter helps people communicate clearly and

inclusively in everyday scenarios, like in Teams meetings, customer support centers, international classrooms, or global events.

For a list of the supported input (source) languages, see the [speech to text languages documentation](#).

Please refer to the [Speech translation how-to guide](#) for the Live Interpreter sample code.

Multiple target languages translation

In scenarios where you want output in multiple languages, the Speech service directly offers the ability for you to translate the input language into two target languages. This enables them to receive two outputs and share these translations to a wider audience with a single API call. If more output languages are required, you can create a multi-service resource or use separate translation services.

If you need translation into more than two target languages, you need to either [Create an AI Foundry resource](#) or utilize separate translation services for more languages beyond the second. If you choose to call the speech translation service with a multi-service resource, please note that translation fees apply for each language beyond the second, based on the character count of the translation.

To calculate the applied translation fee, please refer to [Azure AI Translator pricing](#).

Multiple target languages translation pricing

It's important to note that the speech translation service operates in real-time, and the intermediate speech results are translated to generate intermediate translation results. Therefore, the actual translation amount is greater than the input audio's tokens. You're charged for the speech to text transcription and the text translation for each target language.

For example, let's say that you want text translations from a one-hour audio file to three target languages. If the initial speech to text transcription contains 10,000 characters, you might be charged \$2.80.

Warning

The prices in this example are for illustrative purposes only. Please refer to the [Azure AI Speech pricing](#) and [Azure AI Translator pricing](#) for the most up-to-date pricing information.

The previous example price of \$2.80 was calculated by combining the speech to text transcription and the text translation costs. Here's how the calculation was done:

- The speech translation list price is \$2.50 per hour, covering up to 2 target languages. The price is used as an example of how to calculate costs. See **Standard > Speech translation > Standard** in the [Azure AI Speech pricing table](#) for the most up-to-date pricing information.
- The cost for the third language translation is 30 cents in this example. The translation list price is \$10 per million characters. Since the audio file contains 10,000 characters, the translation cost is $\$10 * 10,000 / 1,000,000 * 3 = \0.3 . The number "3" in this equation represents a weighting coefficient of intermediate traffic, which might vary depending on the languages involved. The price is used as an example of how to calculate costs. See **Standard > Standard translation > Text translation** in the [Azure AI Translator pricing table](#) for the most up-to-date pricing information.

Get started

As your first step, try the [speech translation quickstart](#). The speech translation service is available via the [Speech SDK](#) and the [Speech CLI](#).

You find [Speech SDK speech to text and translation samples](#) on GitHub. These samples cover common scenarios, such as reading audio from a file or stream, continuous and single-shot recognition and translation, and working with custom models.

Next steps

- Try the [speech translation quickstart](#)
- Install the [Speech SDK](#)
- Install the [Speech CLI](#)

Quickstart: Recognize and translate speech to text

07/17/2025

[Reference documentation](#) | [Package \(NuGet\)](#) | [Additional samples on GitHub](#)

In this quickstart, you run an application to translate speech from one language to text in another language.

💡 Tip

Try out the [Azure AI Speech Toolkit](#) to easily build and run samples on Visual Studio Code.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#).
- ✓ [Create an AI Services resource for Speech](#) in the Azure portal.
- ✓ Get the Speech resource key and endpoint. After your Speech resource is deployed, select [Go to resource](#) to view and manage keys.

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

You need to authenticate your application to access Azure AI services. This article shows you how to use environment variables to store your credentials. You can then access the environment variables from your code to authenticate your application. For production, use a more secure way to store and access your credentials.

ⓘ Important

We recommend Microsoft Entra ID authentication with [managed identities for Azure resources](#) to avoid storing credentials with your applications that run in the cloud.

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If using API keys, store them securely in Azure Key Vault, rotate the keys regularly, and restrict access to Azure Key Vault using role based access control and network access restrictions. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

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

- To set the `SPEECH_KEY` environment variable, replace *your-key* with one of the keys for your resource.
- To set the `ENDPOINT` environment variable, replace *your-endpoint* with one of the endpoints for your resource.

Windows

Console

```
setx SPEECH_KEY your-key
setx ENDPOINT your-endpoint
```

 **Note**

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

After you add the environment variables, you might need to restart any programs that need to read the environment variables, including the console window. For example, if you're using Visual Studio as your editor, restart Visual Studio before you run the example.

Translate speech from a microphone

Follow these steps to create a new console application and install the Speech SDK.

1. Open a command prompt where you want the new project, and create a console application with the .NET CLI. The `Program.cs` file should be created in the project directory.

.NET CLI

```
dotnet new console
```

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

.NET CLI

```
dotnet add package Microsoft.CognitiveServices.Speech
```

3. Replace the contents of `Program.cs` with the following code.

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Microsoft.CognitiveServices.Speech.Translation;

class Program
{
    // This example requires environment variables named "SPEECH_KEY" and
    "ENDPOINT"
    static string speechKey =
Environment.GetEnvironmentVariable("SPEECH_KEY");
    static string endpoint = Environment.GetEnvironmentVariable("ENDPOINT");

    static void OutputSpeechRecognitionResult(TranslationRecognitionResult
translationRecognitionResult)
    {
        switch (translationRecognitionResult.Reason)
        {
            case ResultReason.TranslatedSpeech:
                Console.WriteLine($"RECOGNIZED: Text=
{translationRecognitionResult.Text}");
                foreach (var element in
translationRecognitionResult.Translations)
                {
                    Console.WriteLine($"TRANSLATED into '{element.Key}':
{element.Value}");
                }
                break;
            case ResultReason.NoMatch:
                Console.WriteLine($"NOMATCH: Speech could not be
recognized.");
                break;
            case ResultReason.Canceled:
                var cancellation =
CancellationDetails.FromResult(translationRecognitionResult);
                break;
        }
    }
}
```

```

        Console.WriteLine($"CANCELED: Reason={cancellation.Reason}");

        if (cancellation.Reason == CancellationReason.Error)
        {
            Console.WriteLine($"CANCELED: ErrorCode={cancellation.ErrorCode}");
            Console.WriteLine($"CANCELED: ErrorDetails={cancellation.ErrorDetails}");
            Console.WriteLine($"CANCELED: Did you set the speech resource key and endpoint values?");
        }
        break;
    }

    async static Task Main(string[] args)
{
    var speechTranslationConfig =
SpeechTranslationConfig.FromEndpoint(speechKey, endpoint);
    speechTranslationConfig.SpeechRecognitionLanguage = "en-US";
    speechTranslationConfig.AddTargetLanguage("it");

    using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
    using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig, audioConfig);

    Console.WriteLine("Speak into your microphone.");
    var translationRecognitionResult = await
translationRecognizer.RecognizeOnceAsync();
    OutputSpeechRecognitionResult(translationRecognitionResult);
}
}

```

4. To change the speech recognition language, replace `en-US` with another [supported language](#). Specify the full locale with a dash (-) separator. For example, `es-ES` for Spanish (Spain). The default language is `en-US` if you don't specify a language. For details about how to identify one of multiple languages that might be spoken, see [language identification](#).

5. To change the translation target language, replace `it` with another [supported language](#). With few exceptions, you only specify the language code that precedes the locale dash (-) separator. For example, use `es` for Spanish (Spain) instead of `es-ES`. The default language is `en` if you don't specify a language.

Run your new console application to start speech recognition from a microphone:

Console

`dotnet run`

Speak into your microphone when prompted. What you speak should be output as translated text in the target language:

Console

Speak into your microphone.

RECOGNIZED: Text=I'm excited to try speech translation.

TRANSLATED into 'it': Sono entusiasta di provare la traduzione vocale.

Remarks

After completing the quickstart, here are some more considerations:

- This example uses the `RecognizeOnceAsync` operation to transcribe utterances of up to 30 seconds, or until silence is detected. For information about continuous recognition for longer audio, including multi-lingual conversations, see [How to translate speech](#).
- To recognize speech from an audio file, use `FromWavFileInput` instead of `FromDefaultMicrophoneInput`:

C#

```
using var audioConfig = AudioConfig.FromWavFileInput("YourAudioFile.wav");
```

- For compressed audio files such as MP4, install GStreamer and use `PullAudioInputStream` or `PushAudioInputStream`. For more information, see [How to use compressed input audio](#).

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Speech resource you created.

Next steps

[Learn more about speech translation](#)

How to recognize and translate speech

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this how-to guide, you learn how to recognize human speech and translate it to another language.

See the speech translation [overview](#) for more information about:

- Translating speech to text
- Translating speech to multiple target languages
- Performing direct speech to speech translation

Sensitive data and environment variables

The example source code in this article depends on environment variables for storing sensitive data, such as the Speech resource's key and region. The `Program` class contains two `static readonly string` values that are assigned from the host machine's environment variables: `SPEECH_SUBSCRIPTION_KEY` and `SPEECH_SERVICE_REGION`. Both of these fields are at the class scope, so they're accessible within method bodies of the class:

C#

```
public class Program
{
    static readonly string SPEECH_SUBSCRIPTION_KEY =
        Environment.GetEnvironmentVariable(nameof(SPEECH_SUBSCRIPTION_KEY));

    static readonly string SPEECH_SERVICE_REGION =
        Environment.GetEnvironmentVariable(nameof(SPEECH_SERVICE_REGION));

    static Task Main() => Task.CompletedTask;
}
```

For more information on environment variables, see [Environment variables and application configuration](#).

ⓘ Important

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If you use an API key, store it securely in Azure Key Vault. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

Create a speech translation configuration

To call the Speech service by using the Speech SDK, you need to create a `SpeechTranslationConfig` instance. This class includes information about your Speech resource, like your key and associated region, endpoint, host, or authorization token.

Tip

Regardless of whether you're performing speech recognition, speech synthesis, translation, or intent recognition, you'll always create a configuration.

You can initialize `SpeechTranslationConfig` in a few ways:

- With a subscription: pass in a key and the associated region.
- With an endpoint: pass in a Speech service endpoint. A key or authorization token is optional.
- With a host: pass in a host address. A key or authorization token is optional.
- With an authorization token: pass in an authorization token and the associated region.

Let's look at how you create a `SpeechTranslationConfig` instance by using a key and region. Get the Speech resource key and region in the [Azure portal](#).

C#

```
public class Program
{
    static readonly string SPEECH__SUBSCRIPTION__KEY =
        Environment.GetEnvironmentVariable(nameof(SPEECH__SUBSCRIPTION__KEY));

    static readonly string SPEECH__SERVICE__REGION =
        Environment.GetEnvironmentVariable(nameof(SPEECH__SERVICE__REGION));

    static Task Main() => TranslateSpeechAsync();

    static async Task TranslateSpeechAsync()
    {
        var speechTranslationConfig =
            SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);
    }
}
```

Change the source language

One common task of speech translation is specifying the input (or source) language. The following example shows how you would change the input language to Italian. In your code, interact with the `SpeechTranslationConfig` instance by assigning it to the `SpeechRecognitionLanguage` property:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    // Source (input) language
    speechTranslationConfig.SpeechRecognitionLanguage = "it-IT";
}
```

The `SpeechRecognitionLanguage` property expects a language-locale format string. Refer to the [list of supported speech translation locales](#).

Add a translation language

Another common task of speech translation is to specify target translation languages. At least one is required, but multiples are supported. The following code snippet sets both French and German as translation language targets:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    speechTranslationConfig.SpeechRecognitionLanguage = "it-IT";

    speechTranslationConfig.AddTargetLanguage("fr");
    speechTranslationConfig.AddTargetLanguage("de");
}
```

With every call to `AddTargetLanguage`, a new target translation language is specified. In other words, when speech is recognized from the source language, each target translation is available as part of the resulting translation operation.

Initialize a translation recognizer

After you created a `SpeechTranslationConfig` instance, the next step is to initialize `TranslationRecognizer`. When you initialize `TranslationRecognizer`, you need to pass it your `speechTranslationConfig` instance. The configuration object provides the credentials that the Speech service requires to validate your request.

If you're recognizing speech by using your device's default microphone, here's what the `TranslationRecognizer` instance should look like:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguages = new List<string> { "it", "fr", "de" };
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
    toLanguages.ForEach(speechTranslationConfig.AddTargetLanguage);

    using var translationRecognizer = new
    TranslationRecognizer(speechTranslationConfig);
}
```

If you want to specify the audio input device, then you need to create an `AudioConfig` class instance and provide the `audioConfig` parameter when initializing `TranslationRecognizer`.



Tip

[Learn how to get the device ID for your audio input device.](#)

First, reference the `AudioConfig` object as follows:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguages = new List<string> { "it", "fr", "de" };
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
```

```
        toLanguages.ForEach(speechTranslationConfig.AddTargetLanguage);

        using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
        using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig, audioConfig);
    }
```

If you want to provide an audio file instead of using a microphone, you still need to provide an `audioConfig` parameter. However, when you create an `AudioConfig` class instance, instead of calling `FromDefaultMicrophoneInput`, you call `FromWavFileInput` and pass the `filename` parameter:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguages = new List<string> { "it", "fr", "de" };
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
    toLanguages.ForEach(speechTranslationConfig.AddTargetLanguage);

    using var audioConfig = AudioConfig.FromWavFileInput("YourAudioFile.wav");
    using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig, audioConfig);
}
```

Translate speech

To translate speech, the Speech SDK relies on a microphone or an audio file input. Speech recognition occurs before speech translation. After all objects are initialized, call the recognize-once function and get the result:

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguages = new List<string> { "it", "fr", "de" };
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
    toLanguages.ForEach(speechTranslationConfig.AddTargetLanguage);
```

```

using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig);

Console.WriteLine($"Say something in '{fromLanguage}' and ");
Console.WriteLine($"we'll translate into '{string.Join("", "", toLanguages)}').\n");

var result = await translationRecognizer.RecognizeOnceAsync();
if (result.Reason == ResultReason.TranslatedSpeech)
{
    Console.WriteLine($"Recognized: \"{result.Text}\":");
    foreach (var element in result.Translations)
    {
        Console.WriteLine($"      TRANSLATED into '{element.Key}': {element.Value}");
    }
}

```

For more information about speech to text, see [the basics of speech recognition](#).

Event based translation

The `TranslationRecognizer` object exposes a `Recognizing` event. The event fires several times and provides a mechanism to retrieve the intermediate translation results.

 Note

Intermediate translation results aren't available when you use [multi-lingual speech translation](#).

The following example prints the intermediate translation results to the console:

C#

```

using (var audioInput = AudioConfig.FromWavFileInput(@"whatstheweatherlike.wav"))
{
    using (var translationRecognizer = new TranslationRecognizer(config,
audioInput))
    {
        // Subscribes to events.
        translationRecognizer.Recognizing += (s, e) =>
        {
            Console.WriteLine($"RECOGNIZING in '{fromLanguage}': Text= {e.Result.Text}");
            foreach (var element in e.Result.Translations)
            {

```

```

        Console.WriteLine($"      TRANSLATING into '{element.Key}':");
{element.Value}");
    }
};

translationRecognizer.Recognized += (s, e) => {
    if (e.Result.Reason == ResultReason.TranslatedSpeech)
    {
        Console.WriteLine($"RECOGNIZED in '{fromLanguage}': Text={e.Result.Text}");
        foreach (var element in e.Result.Translations)
        {
            Console.WriteLine($"      TRANSLATED into '{element.Key}': {element.Value}");
        }
    }
    else if (e.Result.Reason == ResultReason.RecognizedSpeech)
    {
        Console.WriteLine($"RECOGNIZED: Text={e.Result.Text}");
        Console.WriteLine($"      Speech not translated.");
    }
    else if (e.Result.Reason == ResultReason.NoMatch)
    {
        Console.WriteLine($"NOMATCH: Speech could not be recognized.");
    }
};

// Starts continuous recognition. Uses StopContinuousRecognitionAsync() to
stop recognition.
Console.WriteLine("Start translation...");
await
translationRecognizer.StartContinuousRecognitionAsync().ConfigureAwait(false);

// Waits for completion.
// Use Task.WaitAny to keep the task rooted.
Task.WaitAny(new[] { stopTranslation.Task });

// Stops translation.
await
translationRecognizer.StopContinuousRecognitionAsync().ConfigureAwait(false);
}
}

```

Synthesize translations

After a successful speech recognition and translation, the result contains all the translations in a dictionary. The [Translations](#) dictionary key is the target translation language, and the value is the translated text. Recognized speech can be translated and then synthesized in a different language (speech-to-speech).

Event-based synthesis

The `TranslationRecognizer` object exposes a `Synthesizing` event. The event fires several times and provides a mechanism to retrieve the synthesized audio from the translation recognition result. If you're translating to multiple languages, see [Manual synthesis](#).

Specify the synthesis voice by assigning a `VoiceName` instance, and provide an event handler for the `Synthesizing` event to get the audio. The following example saves the translated audio as a .wav file.

ⓘ Important

The event-based synthesis works only with a single translation. *Do not* add multiple target translation languages. Additionally, the `VoiceName` value should be the same language as the target translation language. For example, `"de"` could map to `"de-DE-Hedda"`.

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SUBSCRIPTION__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguage = "de";
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
    speechTranslationConfig.AddTargetLanguage(toLanguage);

    speechTranslationConfig.VoiceName = "de-DE-Hedda";

    using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig);

    translationRecognizer.Synthesizing += (_, e) =>
    {
        var audio = e.Result.GetAudio();
        Console.WriteLine($"Audio synthesized: {audio.Length:#,0} byte(s)
{(audio.Length == 0 ? "(Complete)" : "")}");

        if (audio.Length > 0)
        {
            File.WriteAllBytes("YourAudioFile.wav", audio);
        }
    };

    Console.Write($"Say something in '{fromLanguage}' and ");
    Console.WriteLine($"we'll translate into '{toLanguage}'.\n");
}
```

```
    var result = await translationRecognizer.RecognizeOnceAsync();
    if (result.Reason == ResultReason.TranslatedSpeech)
    {
        Console.WriteLine($"Recognized: \"{result.Text}\\"");
        Console.WriteLine($"Translated into '{toLanguage}':");
        {result.Translations[toLanguage]}");
    }
}
```

Manual synthesis

You can use the [Translations](#) dictionary to synthesize audio from the translation text. Iterate through each translation and synthesize it. When you're creating a [SpeechSynthesizer](#) instance, the [SpeechConfig](#) object needs to have its [SpeechSynthesisVoiceName](#) property set to the desired voice.

The following example translates to five languages. Each translation is then synthesized to an audio file in the corresponding neural language.

C#

```
static async Task TranslateSpeechAsync()
{
    var speechTranslationConfig =
        SpeechTranslationConfig.FromSubscription(SPEECH__SERVICE__KEY,
SPEECH__SERVICE__REGION);

    var fromLanguage = "en-US";
    var toLanguages = new List<string> { "de", "en", "it", "pt", "zh-Hans" };
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;
    toLanguages.ForEach(speechTranslationConfig.AddTargetLanguage);

    using var translationRecognizer = new
TranslationRecognizer(speechTranslationConfig);

    Console.Write($"Say something in '{fromLanguage}' and ");
    Console.WriteLine($"we'll translate into '{string.Join(", ", toLanguages)}'.\n");

    var result = await translationRecognizer.RecognizeOnceAsync();
    if (result.Reason == ResultReason.TranslatedSpeech)
    {
        var languageToVoiceMap = new Dictionary<string, string>
        {
            ["de"] = "de-DE-KatjaNeural",
            ["en"] = "en-US-AriaNeural",
            ["it"] = "it-IT-ElsaNeural",
            ["pt"] = "pt-BR-FranciscaNeural",
            ["zh-Hans"] = "zh-CN-XiaoxiaoNeural"
        };
    }
}
```

```

Console.WriteLine($"Recognized: \'{result.Text}\'');

foreach (var (language, translation) in result.Translations)
{
    Console.WriteLine($"Translated into '{language}': {translation}");

    var speechConfig =
        SpeechConfig.FromSubscription(
            SPEECH_SERVICE_KEY, SPEECH_SERVICE_REGION);
    speechConfig.SpeechSynthesisVoiceName = languageToVoiceMap[language];

    using var audioConfig = AudioConfig.FromWavFileOutput($"{language}-
translation.wav");
    using var speechSynthesizer = new SpeechSynthesizer(speechConfig,
audioConfig);

    await speechSynthesizer.SpeakTextAsync(translation);
}
}
}

```

For more information about speech synthesis, see [the basics of speech synthesis](#).

Multi-lingual translation with language identification

In many scenarios, you might not know which input languages to specify. Using [language identification](#) you can detect up to 10 possible input languages and automatically translate to your target languages.

The following example anticipates that `en-US` or `zh-CN` should be detected because they're defined in `AutoDetectSourceLanguageConfig`. Then, the speech is translated to `de` and `fr` as specified in the calls to `AddTargetLanguage()`.

C#

```

speechTranslationConfig.AddTargetLanguage("de");
speechTranslationConfig.AddTargetLanguage("fr");
var autoDetectSourceLanguageConfig =
    AutoDetectSourceLanguageConfig.FromLanguages(new string[] { "en-US", "zh-CN" });
var translationRecognizer = new TranslationRecognizer(speechTranslationConfig,
autoDetectSourceLanguageConfig, audioConfig);

```

For a complete code sample, see [language identification](#).

Multi-lingual speech translation without source language candidates

Multi-lingual speech translation implements a new level of speech translation technology that unlocks various capabilities, including having no specified input language, and handling language switches within the same session. These features enable a new level of speech translation powers that can be implemented into your products.

Currently when you use Language ID with speech translation, you must create the `SpeechTranslationConfig` object from the v2 endpoint. Replace the string "YourServiceRegion" with your Speech resource region (such as "westus"). Replace "YourSpeechResoureKey" with your Speech resource key.

```
C#
```

```
var v2EndpointInString =
String.Format("wss://'{0}.stt.speech.microsoft.com/speech/universal/v2",
"YourServiceRegion");
var v2EndpointUrl = new Uri(v2EndpointInString);
var speechTranslationConfig = SpeechTranslationConfig.FromEndpoint(v2EndpointUrl,
"YourSpeechResoureKey");
```

Specify the translation target languages. Replace with languages of your choice. You can add more lines.

```
C#
```

```
config.AddTargetLanguage("de");
config.AddTargetLanguage("fr");
```

A key differentiator with multi-lingual speech translation is that you do not need to specify the source language. This is because the service will automatically detect the source language. Create the `AutoDetectSourceLanguageConfig` object with the `fromOpenRange` method to let the service know that you want to use multi-lingual speech translation with no specified source language.

```
C#
```

```
AutoDetectSourceLanguageConfig autoDetectSourceLanguageConfig =
AutoDetectSourceLanguageConfig.fromOpenRange();
var translationRecognizer = new TranslationRecognizer(speechTranslationConfig,
autoDetectSourceLanguageConfig, audioConfig);
```

For a complete code sample with the Speech SDK, see [speech translation samples on GitHub](#).

Using custom translation in speech translation

The custom translation feature in speech translation seamlessly integrates with the Azure Custom Translation service, allowing you to achieve more accurate and tailored translations. As the integration directly harnesses the capabilities of the Azure custom translation service, you need to use a multi-service resource to ensure the correct functioning of the complete set of features. For detailed instructions, please consult the guide on [Create a multi-service resource for Azure AI services](#).

Additionally, for offline training of a custom translator and obtaining a "Category ID," please refer to the step-by-step script provided in the [Quickstart: Build, deploy, and use a custom model - Custom Translator](#).

C#

```
// Creates an instance of a translation recognizer using speech translation
// configuration
// You should use the same subscription key, which you used to generate the custom
// model before.
// V2 endpoint is required for the “Custom Translation” feature. Example:
// "wss://westcentralus.stt.speech.microsoft.com/speech/universal/v2"

try (SpeechTranslationConfig config =
SpeechTranslationConfig.fromEndpoint(URI.create(endpointUrl),
speechSubscriptionKey)) {

    // Sets source and target language(s).
    ...

    // Set the category id
    config.setCustomModelCategoryId("yourCategoryId");

    ...
}
```

Next steps

- [Try the speech to text quickstart](#)
- [Try the speech translation quickstart](#)
- [Improve recognition accuracy with custom speech](#)

What is video translation?

Video translation is a feature in Azure AI Speech that enables you to seamlessly translate and generate videos in multiple languages automatically. This feature is designed to help you localize your video content to cater to diverse audiences around the globe. You can efficiently create immersive, localized videos across various use cases such as vlogs, education, news, enterprise training, advertising, film, TV shows, and more.

The process of replacing the original language of a video with speech recorded in a different language is essential for catering to diverse audiences. This method, typically achieved through human recording and manual post-production, ensures that viewers can enjoy video content in their native language. However, it comes with key pain points:

- **High cost:** Traditional video translation methods often require expensive human voice actors and extensive post-production work, making it a costly endeavor for content creators.
- **Time-consuming:** The manual process of recording and editing translated speech can take a significant amount of time, delaying the release of localized content.
- **Inconsistent quality:** Human voice actors might not always accurately replicate the original speaker's voice, leading to a less immersive experience for viewers.

With video translation in Azure AI Speech, these challenges are effectively addressed. The feature automates the translation process, significantly reducing costs and production time while ensuring high-quality results. Accurately replicating the original speaker's voice creates a seamless and immersive viewing experience for audiences worldwide.

- **Cost-effective:** Reduces the need for expensive human voice actors and manual post-production work.
- **Time-efficient:** Significantly shortens the time required to produce localized videos.
- **High-quality:** Accurately replicates the original speaker's voice, ensuring a seamless and immersive viewing experience.
- **Scalable:** Enables the production of large volumes of localized content quickly and efficiently.

Use case

Video translation provided by Azure AI Speech has a wide range of use cases across various industries and content types. Here are some key applications:

- **News + interviews:** News organizations can translate and dub news segments and interviews to provide accurate and timely information to audiences worldwide.

- **Advertisement + marketing:** Businesses can localize their advertising and marketing videos to resonate with target audiences in different markets, enhancing brand awareness and customer engagement.
- **Education + learning:** Educational institutions and e-learning platforms can dub their instructional videos and lectures into different languages, making learning more accessible and inclusive.
- **Film + TV show:** Film studios and production companies can dub their movies and TV shows for international distribution, reaching a broader audience and maximizing revenue potential.
- **Vlog + short video:** Content owners can easily translate and dub their vlogs and short videos to reach international audiences, expanding their viewership and engagement.
- **Enterprise training:** Corporations can localize their training videos for employees in different regions, ensuring consistent and effective communication across their workforce.

Core features

- **Dialogue audio extraction and spoken content transcription.**

Automatically extracts dialogue audio from the source video and transcribes the spoken content.

- **Translation from language A to B and large language model (LLM) reformulation.**

Translates the transcribed content from the original language (Language A) to the target language (Language B) using advanced language processing techniques. Enhances translation quality and refines gender-aware translated text through LLM reformulation.

- **Automatic translation – voice generation in other language.**

Utilizes AI-powered text-to-speech technology to automatically generate human-like voices in the target language. These voices are precisely synchronized with the video, ensuring a flawless translation experience. This includes utilizing standard voices for high-quality output and offering options for personal voice.

- **Human in the loop for content editing.**

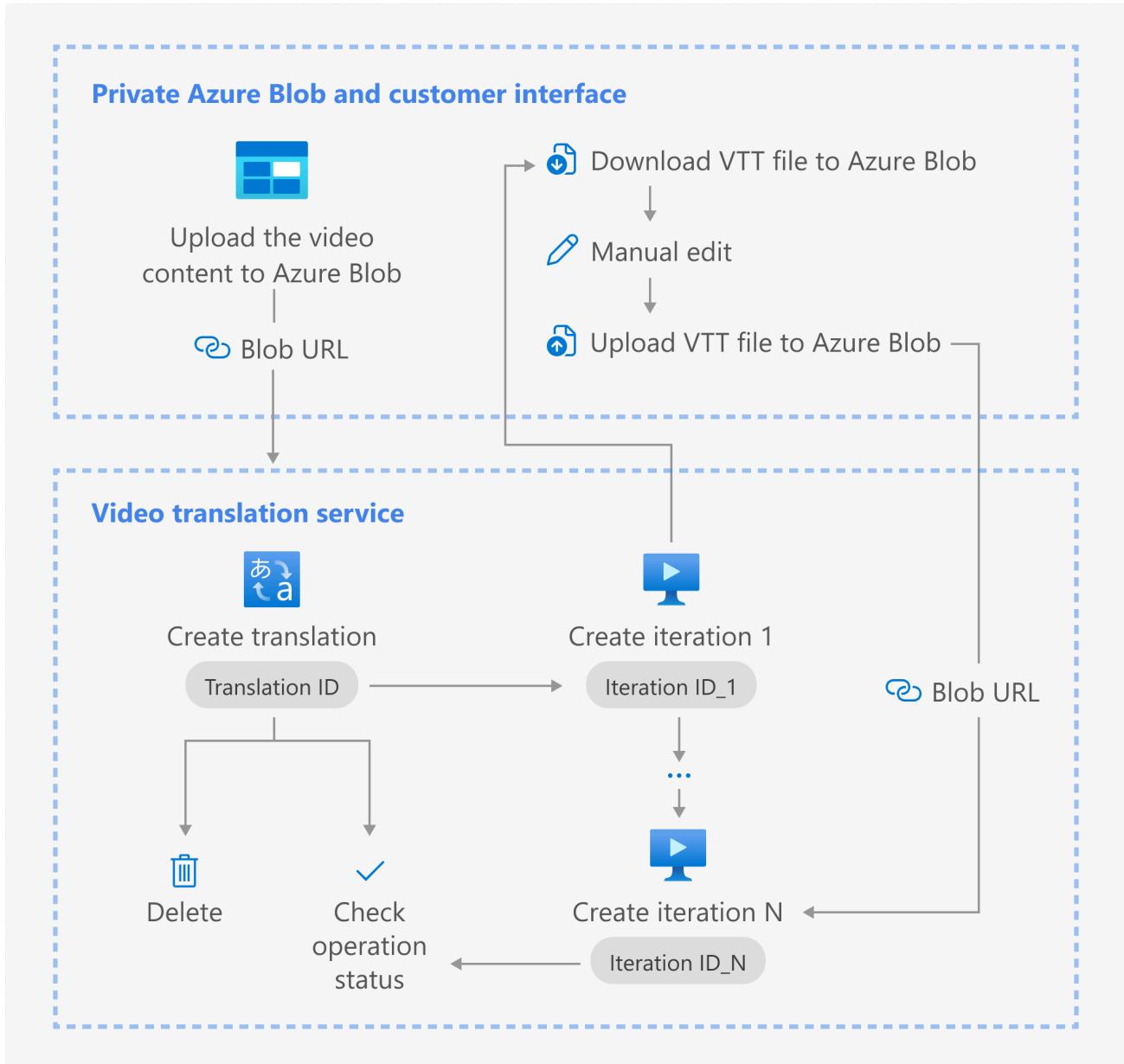
Allows for human intervention to review and edit the translated content, ensuring accuracy and cultural appropriateness before finalizing the dubbed video.

- **Subtitles generation.**

Delivers the fully dubbed video with translated dialogue, synchronized subtitles, and generated voices, ready for download and distribution across various platforms. You can also set the subtitle length on each screen for optimal display.

How it works

This diagram provides a high-level overview of the workflow.



1. You upload the video file that you want translated to Azure Blob Storage.
2. You create a translation by specifying the URL of the video file. Include other parameters, such as the source and target languages, voice type, and whether to burn subtitles into the video.

① Note

Creating a translation doesn't initiate the translation process.

3. You can start translating the video by creating an iteration. An iteration is a specific instance of the translation process. You can create multiple iterations for the same translation, allowing you to experiment with different settings or parameters.
4. After the first iteration, you can use the subtitle file in subsequent iterations. Upload your own subtitle file or make changes to the auto-generated subtitle file and upload the modified subtitle file.
5. Periodically get the status of the translation and iteration. The status will indicate whether the translation is in progress, completed, or failed.
6. Once the translation is complete, you can download the translated video and subtitles. The translated video will have the original speech replaced with the translated speech, and the subtitles will be synchronized with the translated speech.
7. You can also delete the translation and iteration if you no longer need them. Deleting a translation will remove all associated iterations and data.

Supported regions and languages

For the most up-to-date information about regional availability for video translation, see the [Speech service regions table](#).

We support video translation between various languages, enabling you to tailor your content to specific linguistic preferences. For the languages supported for video translation, refer to the [supported source and target languages](#).

Pricing

For pricing details on video translation, see [Speech service pricing](#). Video translation pricing is only visible for service regions where the feature is available. See the [Speech service regions table](#) for current regional availability.

Related content

- To get started with video translation, see [how to use video translation](#).

How to use video translation

Article • 04/16/2025

! Note

This feature is currently in public preview. This preview is provided without a service-level agreement, and we don't recommend it for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

In this article, you learn how to use video translation with Azure AI Speech in the [Azure AI Foundry portal](#).

💡 Tip

Try out video translation in the [Azure AI Foundry portal](#) before using the API. Use the [video translation REST API](#) to integrate video translation into your applications. For more information about the API, see [Video translation REST API](#).

Prerequisites

- An Azure subscription. If you don't have an Azure subscription, create a free account before you begin.
- An Azure AI Services resource for Speech [in a supported region](#). If you don't have a Speech resource, create one in the [Azure portal](#).
- An [Azure Blob Storage](#) account.
- You need a video file in .mp4 format, less than 5 GB, and shorter than 4 hours. For testing purposes, you can use the sample video file provided by Microsoft at <https://speechstudiodprodpublicsa.blob.core.windows.net/ttsvoice/VideoTranslation/PublicDoc/SampleData/es-ES-TryOutOriginal.mp4>.
- Make sure video translation supports your [source and target language](#).

Try out video translation

To try out the video translation demo, follow these steps:

1. Go to the [model catalog in Azure AI Foundry portal](#).
2. Enter and search for "Azure-AI-Speech" in the catalog search box.

Azure AI Foundry / Model catalog

Find the right model to build your custom AI solution

Help

Collections Industry Capabilities Deployment options Inference tasks

Fine-tuning tasks Licenses

Compare models

azure-ai-speech

Models 68

Azure-AI-Speech
Speech recognition, Text to speech

Azure-AI-Language
text-analytics, conversational-ai, ...

Azure-AI-Content-Safety
content-safety, content-filters, re...

Azure-AI-Vision
face-detection, image-analysis, o...

Azure-AI-Translator
Translation, document-translation

Azure-AI-Document-Intellig...
intelligent-document-processing...

3. Select **Azure-AI-Speech** and you're taken to the **Azure-AI-Speech** try out page.

4. Select **Speech capabilities by scenario > Video translation**.

Azure AI Foundry / Model catalog / Azure-AI-Speech

Azure-AI-Speech

Use Service Fine-tune

Try it out Details API Code

Hooray! Speech playground is ready for you.
Try all Speech capabilities with your own data in Speech Playground and explore the possibilities!

All Speech to text Text to speech Speech capabilities by scenario

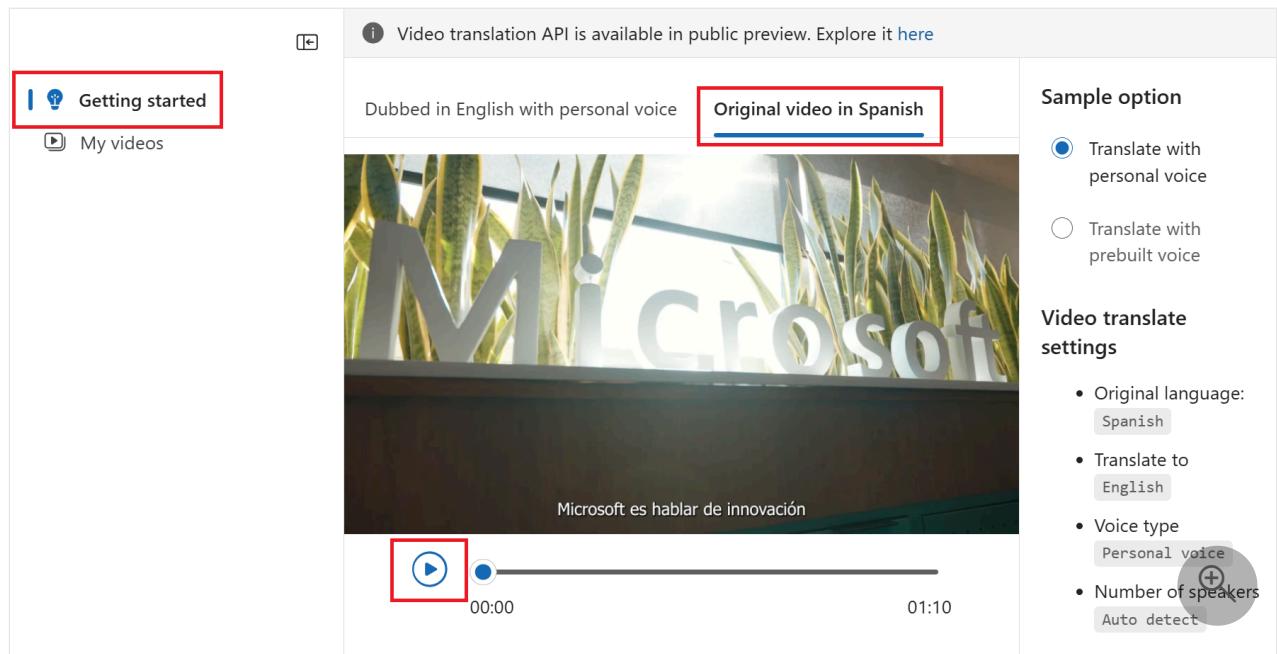
Video translation
Transcribe audio from TV, webcast, live event, or other productions to make your content more accessible.

Go to Speech playground

Video translation API is available in public preview. Explore it [here](#)

5. Under the **Sample** option to the right, select personal or prebuilt voice.

6. Select the **Play** button to hear the translated audio. Select the original video tab to play the original audio.



The voice type options are:

- **Prebuilt voice:** The service automatically selects the most suitable prebuilt voice by matching the speaker's voice in the video with prebuilt voices.
- **Personal voice:** Use the personal voice that matches the voice of the speakers in the video.

ⓘ Note

To use personal voice via the API, you need to apply for [access ↗](#).

Create a video translation project

To create a video translation project, follow these steps:

1. Go to the [model catalog in Azure AI Foundry portal ↗](#).
2. Enter and search for "Azure-AI-Speech" in the catalog search box.

Azure AI Foundry / Model catalog

Find the right model to build your custom AI solution

Help

Collections Industry Capabilities Deployment options Inference tasks

Fine-tuning tasks Licenses

Compare models

azure-ai-speech

Models 68

Azure-AI-Speech
Speech recognition, Text to speech

Azure-AI-Language
text-analytics, conversational-ai, ...

Azure-AI-Content-Safety
content-safety, content-filters, re...

Azure-AI-Vision
face-detection, image-analysis, o...

Azure-AI-Translator
Translation, document-translation

Azure-AI-Document-Intellig...
intelligent-document-processing...

3. Select **Azure-AI-Speech** and you're taken to the **Azure-AI-Speech** try out page.

4. Select **Speech capabilities by scenario > Video translation**.

Azure AI Foundry / Model catalog / Azure-AI-Speech

Azure-AI-Speech

Use Service Fine-tune

Try it out Details API Code

Hooray! Speech playground is ready for you.
Try all Speech capabilities with your own data in Speech Playground and explore the possibilities!

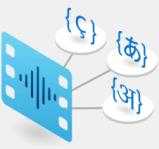
All Speech to text Text to speech Speech capabilities by scenario

Video translation
Transcribe audio from TV, webcast, live event, or other productions to make your content more accessible.

Go to Speech playground

Video translation API is available in public preview. Explore it [here](#)

5. Select **My videos > Upload video**.



Video translation
Seamlessly translate and generate videos in multiple languages automatically.

Getting started

My videos

Upload video

Upload your video to get translated caption and dubbing

6. On the **Upload video** page, select a **Voice type**.

Upload video

Prebuilt neural voice
Automatically match prebuilt neural voices that are similar to the voices in the video.

Personal voice You don't have access
Create the personal voice of the speakers in the video and apply to translated version.
[Watch personal voice sample video](#) and [learn more](#)

Video file *
Drag and drop files (.mp4 file < 500MB, less than 60 minutes)
[Browse files](#)

File name *
Enter a name for your file

[Next: Advanced settings](#) [Create](#) [Cancel](#)

The voice type options are:

- **Prebuilt neural voice:** The service automatically selects the most suitable prebuilt voice by matching the speaker's voice in the video with prebuilt voices.
- **Personal voice:** Use the personal voice that matches the voice of the speakers in the video.

 **Note**

To use personal voice, you need to apply for [access](#).

7. Upload your video file by dragging and dropping the video file or selecting the file manually. The video must be in .mp4 format, less than 5 GB, and shorter than 4 hours.
8. Provide the **Number of speakers**, **Language of the video**, and **Translate to** language.
9. Select the boxes to acknowledge the pricing information and code of conduct.
10. Select **Next: Advanced settings** if you want to adjust the advanced settings.

Upload video X



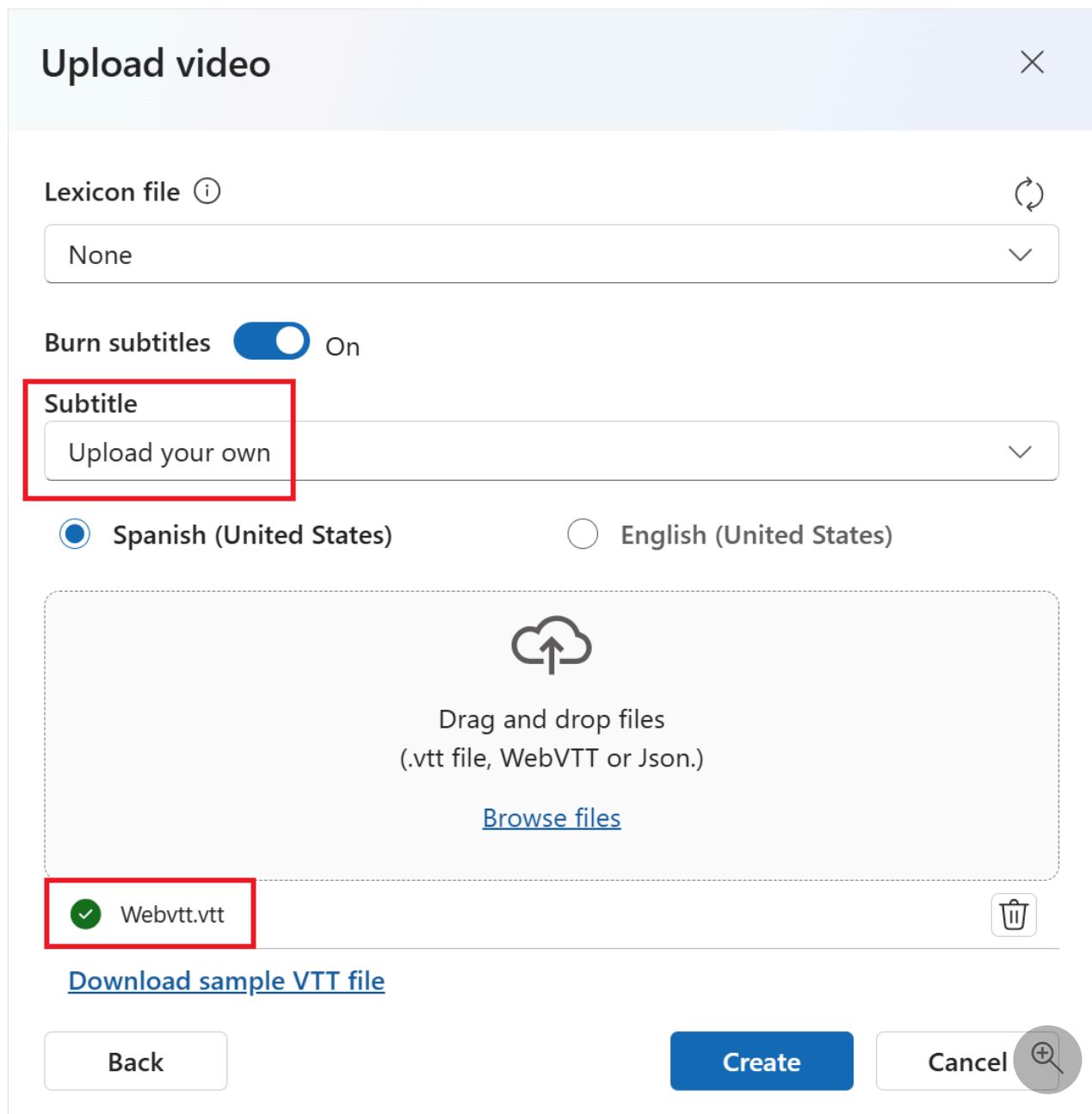
Drag and drop files
.mp4 file < 500MB, less than 60 minutes)

[Browse files](#)

<input checked="" type="checkbox"/> es-ES-TryOutOriginal.mp4	Delete
File name * <input type="text" value="es-ES-TryOutOriginal.mp4"/>	
Number of speakers * Auto Detect <input checked="" type="button"/> <input type="text" value="1"/>	
Language of the Video * Translate to * <div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <input type="text" value="Spanish (United States)"/> </div> <div style="width: 45%;"> <input type="text" value="English (United States)"/> </div> </div>	
<input checked="" type="checkbox"/> * I acknowledge that video translation will incur usage charges to my account, effective from June 2024. See pricing <input checked="" type="checkbox"/> * Accept code of conduct for Azure AI Speech text to speech	
Next: Advanced settings Create Cancel	

11. Optionally, you can adjust the following settings:

- **Lexicon file:** This option allows you to add custom words or phrases that the system should recognize and pronounce correctly. You can create a lexicon file in the [audio content creation tool in the Speech Studio](#) and select it here.
- **Burn subtitles:** This option allows you to add subtitles to the video. The subtitle file can be in WebVTT or JSON format. You can download a sample WebVTT file for your reference by selecting **Download sample VTT file**.



If you want to use your own subtitle files, select **Subtitle > Upload your own**. You can choose to upload either the source subtitle file or the target subtitle file.

- Automatic subtitles: Results in both source and target language subtitles.
- Upload source language subtitles: Results in both source and target language subtitles.
- Upload target language subtitles: Results in only target language subtitles.

12. Select Create.

Once the upload to Azure Blob Storage is complete, you can check the processing status on the project tab.

After the project is created, you can select the project to review detailed settings and make adjustments according to your preferences.

Check and adjust voice settings

Select **My videos** and you should see a video labeled with **Succeeded** status.

The screenshot shows the Microsoft Translator interface. On the left, there are two tabs: 'Getting started' and 'My videos'. The 'My videos' tab is highlighted with a red box. On the right, there is a large video thumbnail of a man with glasses and a white shirt. Above the thumbnail is a blue button labeled 'Upload video'. Below the thumbnail, the video details are listed:

- es-ES-TryOutOriginal
- Total duration: 00:01:10
- Source language: Spanish (United States)
- Translate to: English (United States)

At the bottom of the card, there is a green checkmark icon followed by the word 'Succeeded' and three dots on the right. A magnifying glass icon is located in the bottom right corner of the card.

Select the video to see the **Translated** and **Original** tabs under **Video**. You can compare the original and translated videos by selecting the corresponding tab. The translated video is generated automatically, and you can play it to check the translation quality.

The screenshot shows the Microsoft Translator interface for video translation. At the top, there's a status bar with 'Status: Succeeded', 'Source language: Spanish (United States)', 'Translate to: English (United States)', 'Voice type: Prebuilt neural voice', 'Created: 4/15/2025 08:35 AM', and 'Modified: 4/15/2025 08:36 AM'. Below the status bar are buttons for 'Apply changes', 'Save', 'Voice settings', 'New language', 'Download', and 'Lexicon'. A message box says 'Video translation API is available in public preview. Explore it [here](#)'. On the left, there are tabs for 'Translated' (which is selected) and 'Original'. The main area shows a video player with a thumbnail of a cityscape, a play button, and a progress bar from 00:00:01.927 to 00:01:10.640. To the right, there are two sections of subtitles. The first section is for 'Speaker1' from 00:00:01.010 to 00:00:06.030, with the original text 'Hello this is a sample subtitle.' and the translated text 'Hola, esto es un subtítulo de muestra.'. The second section is for 'Speaker1' from 00:00:07.030 to 00:00:09.030, with the same text pairs. Below the subtitles is an 'Audio wave' visualization from the 'translated file', showing a waveform with markers for 'Speaker1: Hola, esto es un subtítulo de muestra.' at 00:00:01.010 and 'Speaker1: Hola. esto es' at 00:00:07.030. There are also icons for microphone, search, and refresh.

To the right side of the video, you can view both the original script and the translated script. Hovering over each part of the original script triggers the video to automatically jump to the corresponding segment of the original video, while hovering over each part of the translated script triggers the video to jump to the corresponding translated segment.

You can make multiple changes to the video, including adjusting the voice settings, adding or removing segments, and changing the time frame of the scripts. You're only charged after you select **Apply changes** to apply your changes. You can select **Save** to save work in progress without incurring any charges.

If you encounter segments with an "unidentified" voice name, it might be because the system couldn't accurately detect the voice, especially in situations where speaker voices overlap. In such cases, it's advisable to manually change the voice name.

Guest_0	Hey, wow, look, this hat has a triangle shape.	嘿, 哇, 看, 这顶帽子是三角形的。	00:00:19.610 - 00:00:27.840
Unidentified	No, it's a rectangle.	不, 它是个矩形。	00:00:27.880 - 00:00:30.560
Guest_0	Both of you are not correct It is a square consists of two rectangles, right?	你们俩都不对, 这是一个由两个矩形组成的正方形, 对吧?	00:00:30.650 - 00:00:38.230

Translate to another language

You can keep the current translation project and translate the original video into another language.

1. Select **My videos** and then select the tile for your video translation.
2. Select **+ New language**.
3. On the new **Translate to new language** page that appears, choose a new translation language and voice type. Once the video is translated, a new project is automatically created.

Related content

- [Video translation overview](#)

LLM speech for speech transcription and translation (Preview)

ⓘ Note

This feature is currently in public preview. This preview is provided without a service-level agreement, and is not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

LLM speech is powered by a large-language-model-enhanced speech model that delivers improved quality, deep contextual understanding, multilingual support, and prompt-tuning capabilities. It uses GPU acceleration for ultra-fast inference, making it ideal for a wide range of scenarios including generating captions and subtitles from audio files, summarizing meeting notes, assisting call center agents, transcribing voicemails, and more.

The LLM speech API currently supports the following speech tasks:

- `transcribe`
- `translate`

Prerequisites

- An Azure AI Speech resource in one of the regions where the LLM speech API is available.
For the current list of supported regions, see [Speech service regions](#).
- An audio file (less than 2 hours long and less than 300 MB in size) in one of the formats and codecs supported by the batch transcription API: WAV, MP3, OPUS/OGG, FLAC, WMA, AAC, ALAW in WAV container, MULAW in WAV container, AMR, WebM, and SPEEX.
For more information about supported audio formats, see [supported audio formats](#).

Use the LLM speech API

Supported languages

The following languages are currently supported for both `transcribe` and `translate` tasks:

- English, Chinese, German, French, Italian, Japanese, Spanish, Portuguese, and Korean.

Upload audio

You can provide audio data in the following ways:

- Pass inline audio data.

```
--form 'audio=@"YourAudioFile"'
```

- Upload audio file from a public `audioUrl`.

```
--form 'definition": "{\"audioUrl\": \"https://crbn.us/hello.wav\"}'''
```

In the sections below, inline audio upload is used as an example.

Call the LLM speech API

Make a multipart/form-data POST request to the `transcriptions` endpoint with the audio file and the request body properties.

The following example shows how to transcribe an audio file with a specified locale. If you know the locale of the audio file, you can specify it to improve transcription accuracy and minimize the latency.

- Replace `YourSpeechResourceKey` with your Speech resource key.
- Replace `YourServiceRegion` with your Speech resource region.
- Replace `YourAudioFile` with the path to your audio file.

Important

For the recommended keyless authentication with Microsoft Entra ID, replace `--header 'Ocp-Apim-Subscription-Key: YourSpeechResourceKey'` with `--header "Authorization: Bearer YourAccessToken"`. For more information about keyless authentication, see the [role-based access control](#) how-to guide.

Use LLM speech to transcribe an audio

You can transcribe audio in the input language without specifying a locale code. The model automatically detects and selects the appropriate language based on the audio content.

Azure CLI

```
curl --location  
'https://<YourServiceRegion>.api.cognitive.microsoft.com/speechtotext/transcriptions  
:transcribe?api-version=2025-10-15' \  
--header 'Content-Type: multipart/form-data' \  
--header 'Ocp-Apim-Subscription-Key: <YourSpeechResourceKey>' \  
--form 'audio=@"YourAudioFile.wav"' \  
--form 'definition={  
    "enhancedMode": {  
        "enabled": true,  
        "task": "transcribe"  
    }  
}'
```

Use LLM speech to translate an audio file

You can translate audio into a specified target language. To enable translation, you must provide the target language code in the request.

Azure CLI

```
curl --location  
'https://<YourServiceRegion>.api.cognitive.microsoft.com/speechtotext/transcriptions  
:transcribe?api-version=2025-10-15' \  
--header 'Content-Type: multipart/form-data' \  
--header 'Ocp-Apim-Subscription-Key: <YourSpeechResourceKey>' \  
--form 'audio=@"YourAudioFile.wav"' \  
--form 'definition={  
    "enhancedMode": {  
        "enabled": true,  
        "task": "translate",  
        "targetLanguage": "ko"  
    }  
}'
```

Use prompt-tuning to alter performance

You can provide an optional text to guide the output style for `transcribe` or `translate` task.

Azure CLI

```
curl --location  
'https://<YourServiceRegion>.api.cognitive.microsoft.com/speechtotext/transcriptions  
:transcribe?api-version=2025-10-15' \  
--header 'Content-Type: multipart/form-data' \  
--header 'Ocp-Apim-Subscription-Key: <YourSpeechResourceKey>' \  
--form 'audio=@"YourAudioFile.wav"' \  
--form 'definition={
```

```
"enhancedMode": {  
    "enabled": true,  
    "task": "transcribe",  
    "prompt": ["Output must be in lexical format."]  
},  
}'
```

Here are some best practices for prompts:

- Prompts are subject to a maximum length of 4,096 characters.
- Prompts should preferably be written in English.
- Prompts can guide output formatting. By default, responses use a display format optimized for readability. To enforce lexical formatting, include: `Output must be in lexical format.`
- Prompts can amplify the salience of specific phrases or acronyms, improving recognition likelihood. Use: `Pay attention to *phrase1*, *phrase2*, ...`. For best results, limit the number of phrases per prompt.
- Prompts that aren't related to speech tasks (e.g., `Tell me a story.`) are typically disregarded.

More configuration options

You can combine additional configuration options with [fast transcription](#) to enable enhanced features such as `diarization`, `profanityFilterMode`, and `channels`.

Azure CLI

```
curl --location  
'https://<YourServiceRegion>.api.cognitive.microsoft.com/speechtotext/transcriptions  
:transcribe?api-version=2025-10-15' \  
--header 'Content-Type: multipart/form-data' \  
--header 'Ocp-Apim-Subscription-Key: <YourSpeechResourceKey>' \  
--form 'audio=@"YourAudioFile.wav"' \  
--form 'definition={  
    "enhancedMode": {  
        "enabled": true,  
        "task": "transcribe",  
        "prompt": ["Output must be in lexical format."]  
    },  
    "diarization": {  
        "maxSpeakers": 2,  
        "enabled": true  
    },  
    "profanityFilterMode": "Masked"  
}'
```

Some configuration options, such as `locales` and `phraseLists`, are either not required or not applicable with LLM speech, and can be omitted from the request. Learn more from [configuration options of fast transcription](#).

Sample response

In the JSON response, the `combinedPhrases` property contains the full transcribed or translated text, and the `phrases` property contains segment-level and word-level details.

JSON

```
{  
    "durationMilliseconds": 57187,  
    "combinedPhrases": [  
        {  
            "text": "With custom speech,you can evaluate and improve the microsoft  
speech to text accuracy for your applications and products 现成的语音转文本,利用通用语  
言模型作为一个基本模型,使用microsoft自有数据进行训练,并反映常用的口语。此基础模型使用那些代  
表各常见领域的方言和发音进行了预先训练。 Quand vous effectuez une demande de  
reconnaissance vocale, le modèle de base le plus récent pour chaque langue prise en  
charge est utilisé par défaut. Le modèle de base fonctionne très bien dans la  
plupart des scénarios de reconnaissance vocale. A custom model can be used to  
augment the base model to improve recognition of domain specific vocabulary  
specified to the application by providing text data to train the model. It can also  
be used to improve recognition based for the specific audio conditions of the  
application by providing audio data with reference transcriptions."  
        }  
    ],  
    "phrases": [  
        {  
            "offsetMilliseconds": 80,  
            "durationMilliseconds": 6960,  
            "text": "With custom speech,you can evaluate and improve the microsoft  
speech to text accuracy for your applications and products.",  
            "words": [  
                {  
                    "text": "with",  
                    "offsetMilliseconds": 80,  
                    "durationMilliseconds": 160  
                },  
                {  
                    "text": "custom",  
                    "offsetMilliseconds": 240,  
                    "durationMilliseconds": 480  
                },  
                {  
                    "text": "speech",  
                    "offsetMilliseconds": 720,  
                    "durationMilliseconds": 360  
                },  
                ...  
            ]  
        }  
    ]  
}
```

```
// More transcription results...
// Redacted for brevity
    ],
    "locale": "en-us",
    "confidence": 0
},
{
    "offsetMilliseconds": 8000,
    "durationMilliseconds": 8600,
    "text": "现成的语音转文本，利用通用语言模型作为一个基本模型，使用microsoft自有  
数据进行训练，并反映常用的口语。此基础模型使用那些代表各常见领域的方言和发音进行了预先训  
练。",
    "words": [
        {
            "text": "现",
            "offsetMilliseconds": 8000,
            "durationMilliseconds": 40
        },
        {
            "text": "成",
            "offsetMilliseconds": 8040,
            "durationMilliseconds": 40
        },
        ...
    ],
    // More transcription results...
    // Redacted for brevity
    {
        "text": "训",
        "offsetMilliseconds": 16400,
        "durationMilliseconds": 40
    },
    {
        "text": "练",
        "offsetMilliseconds": 16560,
        "durationMilliseconds": 40
    },
    ...
],
"locale": "zh-cn",
"confidence": 0
// More transcription results...
// Redacted for brevity
{
    "text": "with",
    "offsetMilliseconds": 54720,
    "durationMilliseconds": 200
},
{
    "text": "reference",
    "offsetMilliseconds": 54920,
    "durationMilliseconds": 360
},
{
    "text": "transcriptions.",
    "offsetMilliseconds": 55280,
    "durationMilliseconds": 1200
}
```

```
        ],
        "locale": "en-us",
        "confidence": 0
    }
]
}
```

The response format is consistent with other existing speech-to-text outputs, such as fast transcription and batch transcription. Key differences include:

- Word-level `durationMilliseconds` and `offsetMilliseconds` are not supported for `translate` task.
- Diarization is not supported for `translate` task, only the `speaker1` label is returned.
- `confidence` is not available and always `0`.

ⓘ Note

Speech service is an elastic service. If you receive 429 error code (too many requests), please follow the [best practices to mitigate throttling during autoscaling](#).

Related content

- [LLM speech REST API reference](#)
- [Fast transcription](#)

Last updated on 11/05/2025

Transcriptions - Transcribe

Service: Azure AI Services

API Version: 2025-10-15

Synchronous transcription of an audio file.

HTTP

POST {endpoint}/speechtotext/transcriptions:transcribe?api-version=2025-10-15

URI Parameters

[Expand table](#)

Name	In	Required	Type	Description
audio	formData		file (binary)	The content of the audio file to be transcribed. The audio file must be shorter than 2 hours in audio duration and smaller than 250 MB in size.
definition	formData		string	Metadata for a transcription request. This field contains a JSON-serialized object of type <code>TranscribeDefinition</code> .
endpoint	path	True	string	Supported Cognitive Services endpoints (protocol and hostname, for example: https://westus.api.cognitive.microsoft.com).
api-version	query	True	string	The requested api version.

Request Header

Media Types: "multipart/form-data"

[Expand table](#)

Name	Required	Type	Description
Ocp-Apim-Subscription-Key	True	string	Provide your cognitive services account key here.

Responses

[Expand table](#)

Name	Type	Description
200 OK	TranscribeResult	OK
Other Status Codes	Error	An error occurred.

Security

Ocp-Apim-Subscription-Key

Provide your cognitive services account key here.

Type: apiKey

In: header

Examples

Transcribe an audio file

Sample request

HTTP

HTTP

```
POST {endpoint}/speechtotext/transcriptions:transcribe?api-version=2025-10-15
```

Sample response

Status code: 200

JSON

```
{  
    "durationMilliseconds": 2000,  
    "combinedPhrases": [  
        {  
            "text": "Weather"  
        }  
    ],  
    "phrases": [  
        {  
            "offsetMilliseconds": 40,  
            "durationMilliseconds": 320,  
            "text": "Weather",  
            "words": [  
                {  
                    "text": "weather",  
                    "offsetMilliseconds": 40,  
                    "durationMilliseconds": 320  
                }  
            ],  
            "locale": "en-US",  
            "confidence": 0.78983736  
        }  
    ]  
}
```

Definitions

 Expand table

Name	Description
ChannelCombinedPhrases	The full transcript per channel.
DetailedErrorCode	DetailedErrorCode
Error	Error
ErrorCode	ErrorCode
InnerError	InnerError
Phrase	A transcribed phrase.
TranscribeResult	The result of the transcribe operation.

Word

Time-stamped word in the display form.

ChannelCombinedPhrases

Object

The full transcript per channel.

[Expand table](#)

Name	Type	Description
channel	integer (int32)	The 0-based channel index. Only present if channel separation is enabled.
text	string	The transcribed text.

DetailedErrorCode

Enumeration

DetailedErrorCode

[Expand table](#)

Value	Description
InvalidParameterValue	Invalid parameter value.
InvalidRequestBodyFormat	Invalid request body format.
EmptyRequest	Empty Request.
MissingInputRecords	Missing Input Records.
InvalidDocument	Invalid Document.
ModelErrorIncorrect	Model Version Incorrect.
InvalidDocumentBatch	Invalid Document Batch.
UnsupportedLanguageCode	Unsupported language code.
DataImportFailed	Data import failed.
InUseViolation	In use violation.
InvalidLocale	Invalid locale.
InvalidBaseModel	Invalid base model.
InvalidAdaptationMapping	Invalid adaptation mapping.
InvalidDataset	Invalid dataset.
InvalidTest	Invalid test.
FailedDataset	Failed dataset.
InvalidModel	Invalid model.
InvalidTranscription	Invalid transcription.
InvalidPayload	Invalid payload.
InvalidParameter	Invalid parameter.
EndpointWithoutLogging	Endpoint without logging.
InvalidPermissions	Invalid permissions.

InvalidPrerequisite	Invalid prerequisite.
InvalidProductId	Invalid product id.
InvalidSubscription	Invalid subscription.
InvalidProject	Invalid project.
InvalidProjectKind	Invalid project kind.
InvalidRecordingsUri	Invalid recordings uri.
OnlyOneOfUrlsOrContainerOrDataset	Only one of urls or container or dataset.
ExceededNumberOfRecordingsUris	Exceeded number of recordings uris.
InvalidChannels	Invalid channels.
ModelMismatch	Model mismatch.
ProjectGenderMismatch	Project gender mismatch.
ModelDeprecated	Model deprecated.
ModelExists	Model exists.
ModelNotDeployable	Model not deployable.
EndpointNotUpdatable	Endpoint not updatable.
SingleDefaultEndpoint	Single default endpoint.
EndpointCannotBeDefault	Endpoint cannot be default.
InvalidModelError	Invalid model error.
SubscriptionNotFound	Subscription not found.
QuotaViolation	Quota violation.
UnsupportedDelta	Unsupported delta.
UnsupportedFilter	Unsupported filter.
UnsupportedPagination	Unsupported pagination.
UnsupportedDynamicConfiguration	Unsupported dynamic configuration.
UnsupportedOrderBy	Unsupported order by.
NoUtf8WithBom	No utf8 with bom.
ModelDeploymentNotCompleteState	Model deployment not complete state.
SkuLimitsExist	Sku limits exist.
DeployingFailedModel	Deploying failed model.
UnsupportedTimeRange	Unsupported time range.
InvalidLogDate	Invalid log date.
InvalidLogId	Invalid log id.
InvalidLogStartTime	Invalid log start time.
InvalidLogEndTime	Invalid log end time.
InvalidTopForLogs	Invalid top for logs.
InvalidSkipTokenForLogs	Invalid skip token for logs.
DeleteNotAllowed	Delete not allowed.

Forbidden	Forbidden.
DeployNotAllowed	Deploy not allowed.
UnexpectedError	Unexpected error.
InvalidCollection	Invalid collection.
InvalidCallbackUri	Invalid callback uri.
InvalidSasValidityDuration	Invalid sas validity duration.
InaccessibleCustomerStorage	Inaccessible customer storage.
UnsupportedClassBasedAdaptation	Unsupported class based adaptation.
InvalidWebHookEventKind	Invalid web hook event kind.
InvalidTimeToLive	Invalid time to live.
InvalidSourceAzureResourceId	Invalid source Azure resource ID.
ModelCopyAuthorizationExpired	Expired ModelCopyAuthorization.
EndpointLoggingNotSupported	Endpoint logging not supported.
NoLanguageIdentified	Language Identification did not recognize any language.
MultipleLanguagesIdentified	Language Identification recognized multiple languages. No dominant language could be determined.
InvalidAudioFormat	The format of input audio is not supported.
BadChannelConfiguration	There is a mismatch between audio channels in the data, in the configuration, or the requirements of the application.
InvalidChannelSpecification	The selection of channels in the transcription request is not supported (e.g., neither 0 nor 1 have been selected.)
AudioLengthLimitExceeded	The audio file is longer than the maximum allowed duration.
EmptyAudioFile	The audio file is empty.

Error

Object

Error

[+] Expand table

Name	Type	Description
code	Error Code	ErrorCode High level error codes.
details	Error[]	Additional supportive details regarding the error and/or expected policies.
innerError	Inner Error	New Inner Error format which conforms to Cognitive Services API Guidelines which is available at https://microsoft.sharepoint.com/%3Aw%3A/t/CognitiveServicesPMO/EUoytcruJdKpeOKIK_QRC8BptUYQpKBi8JsWyeDMRsWIQ?e=CPq8ow . This contains required properties ErrorCode, message and optional properties target, details(key value pair), inner error(this can be nested).
message	string	High level error message.
target	string	The source of the error. For example it would be "documents" or "document id" in case of invalid document.

ErrorCode

Enumeration

ErrorCode

[Expand table](#)

Value	Description
InvalidRequest	Representing the invalid request error code.
InvalidArgument	Representing the invalid argument error code.
InternalServerError	Representing the internal server error error code.
ServiceUnavailable	Representing the service unavailable error code.
NotFound	Representing the not found error code.
PipelineError	Representing the pipeline error error code.
Conflict	Representing the conflict error code.
InternalCommunicationFailed	Representing the internal communication failed error code.
Forbidden	Representing the forbidden error code.
NotAllowed	Representing the not allowed error code.
Unauthorized	Representing the unauthorized error code.
UnsupportedMediaType	Representing the unsupported media type error code.
TooManyRequests	Representing the too many requests error code.
UnprocessableEntity	Representing the unprocessable entity error code.

InnerError

Object

InnerError

[Expand table](#)

Name	Type	Description
code	Detailed Error Code	Detailed error code enum.
details	object	Additional supportive details regarding the error and/or expected policies.
innerError	Inner Error	New Inner Error format which conforms to Cognitive Services API Guidelines which is available at https://microsoft.sharepoint.com/:3Aw%3A/t/CognitiveServicesPMO/EUoytcrjuJdKpeOKIK_QRC8BPtUYQpKBi8JsWyeDMRsWIQ?e=CPq8ow . This contains required properties ErrorCode, message and optional properties target, details(key value pair), inner error(this can be nested).
message	string	High level error message.
target	string	The source of the error. For example it would be "documents" or "document id" in case of invalid document.

Phrase

Object

A transcribed phrase.

[Expand table](#)

Name	Type	Description
channel	integer (int32)	The 0-based channel index. Only present if channel separation is enabled.
confidence	number (float)	The confidence value for the phrase.
durationMilliseconds	integer (int32)	The duration of the phrase in milliseconds.
locale	string	The locale of the phrase.
offsetMilliseconds	integer (int32)	The start offset of the phrase in milliseconds.
speaker	integer (int32)	A unique integer number that is assigned to each speaker detected in the audio without particular order. Only present if speaker diarization is enabled.
text	string	The transcribed text of the phrase.
words	Word[]	The words that make up the phrase. Only present if word-level timestamps are enabled.

TranscribeResult

Object

The result of the transcribe operation.

[Expand table](#)

Name	Type	Description
combinedPhrases	ChannelCombinedPhrases[]	The full transcript for each channel.
durationMilliseconds	integer (int32)	The duration of the audio in milliseconds.
phrases	Phrase[]	The transcription results segmented into phrases.

Word

Object

Time-stamped word in the display form.

[Expand table](#)

Name	Type	Description
durationMilliseconds	integer (int32)	The duration of the word in milliseconds.
offsetMilliseconds	integer (int32)	The start offset of the word in milliseconds.
text	string	The recognized word, including punctuation.

Voice live API for real-time voice agents

What is the Voice live API?

The Voice live API is a solution enabling low-latency, high-quality speech to speech interactions for voice agents. The API is designed for developers seeking scalable and efficient voice-driven experiences as it eliminates the need to manually orchestrate multiple components. By integrating speech recognition, generative AI, and text to speech functionalities into a single, unified interface, it provides an end-to-end solution for creating seamless experiences.

Understanding speech to speech experiences

Speech to speech technology is revolutionizing how humans interact with systems, offering intuitive voice-based solutions. Traditional implementations involved combining disparate modules such as speech to text, dialog management, text to speech, and more. Such chaining can lead to increased engineering complexity and end-user perceived latency.

With advancements in Large Language Models (LLMs) and multimodal AI, the Voice live API consolidates these functionalities, simplifying workflows for developers. This approach enhances real-time interactions and ensures high-quality, natural communication, making it suitable for industries requiring instant, voice-enabled solutions.

Key Scenarios for Voice live API

Azure AI Voice live API is ideal for scenarios where voice-driven interactions improve user experience. Examples include:

- **Contact centers:** Develop interactive voice bots for customer support, product catalog navigation, and self-service solutions.
- **Automotive assistants:** Enable hands-free, in-car voice assistants for command execution, navigation, and general inquiries.
- **Education:** Create voice-enabled learning companions and virtual tutors for interactive training and education.
- **Public services:** Build voice agents to assist citizens with administrative queries and public service information.
- **Human resources:** Enhance HR processes with voice-enabled tools for employee support, career development, and training.

Features of the Voice live API

The Voice live API includes a comprehensive set of features to support diverse use cases and ensure superior voice interactions:

- **Broad locale coverage:** Supports over 140 locales for speech to text and offers over 600 standard voices across 150+ locales for text to speech, ensuring global accessibility.
- **Customizable input and output:** Use phrase list for lightweight just-in-time customization on audio input or custom speech models for advanced speech recognition fine-tuning. Use custom voice to create unique, brand-aligned voices for audio output. See [How to customize voice live input and output](#) to learn more.
- **Flexible generative AI model options:** Choose from multiple models, including GPT-5, GPT-4.1, GPT-4o, Phi, and more tailored to conversational requirements.
- **Advanced conversational features:**
 - Noise suppression: Reduces environmental noise for clearer communication.
 - Echo cancellation: Prevents the agent from picking up its own responses.
 - Robust interruption detection: Ensures accurate recognition of interruptions during conversations.
 - Advanced end-of-turn detection: Allows natural pauses without prematurely concluding interactions.
- **Avatar integration:** Provides standard or customizable avatars synchronized with audio output, offering a visual identity for voice agents.
- **Function calling:** Enables external actions, use of tools, and grounded responses using the VoiceRAG pattern.

How it works

The Voice live API is fully managed, eliminating the need for customers to handle backend orchestration or component integration. Developers provide audio input and receive audio output, avatar visuals, and action triggers—all with minimal latency. You don't need to deploy or manage any generative AI models, as the API handles the underlying infrastructure.

API design and compatibility

The Voice live API is designed for compatibility with the Azure OpenAI Realtime API. The supported real-time events are mostly in parity with the [Azure OpenAI Realtime API events](#), with some exceptions as described in the [Voice live API how to guide](#).

Features that are unique to the Voice live API are designed to be optional and additive. You can add Azure AI Speech capabilities such as noise suppression, echo cancellation, and advanced end-of-turn detection to your existing applications without needing to change your existing architecture.

The API is supported through WebSocket events, allowing for an easy server-to-server integration. Your backend or middle-tier service connects to the Voice live API via WebSockets. You can use the WebSocket messages directly to interact with the API.

Supported models and regions

To power the intelligence of your voice agent, you have flexibility and choice in the generative AI model between GPT-Realtime, GPT-5, GPT-4.1, Phi, and more options. Different generative AI models provide different types of capabilities, levels of intelligence, speed/latency of inferencing, and cost. Depending on what matters most for your business and use case, you can choose the model that best suits your needs.

All natively supported models are fully managed, meaning you don't have to deploy models, worry about capacity planning, or provisioning throughput. You can use the model you need, and the Voice live API takes care of the rest.

The Voice live API supports the following models. For supported regions, see the [Azure AI Speech service regions](#).

 Expand table

Model	Description
gpt-realtime	GPT real-time + option to use Azure text to speech voices including custom voice for audio.
gpt-realtime-mini	GPT mini real-time + option to use Azure text to speech voices including custom voice for audio.
gpt-4o	GPT-4o + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-4o-mini	GPT-4o mini + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-4.1	GPT-4.1 + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-4.1-mini	GPT-4.1 mini + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-5	GPT-5 + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-5-mini	GPT-5 mini + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.

Model	Description
gpt-5-nano	GPT-5 nano + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
gpt-5-chat	GPT-5 chat + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.
phi4-mm-realtime	Phi4-mm + audio output through Azure text to speech voices including custom voice.
phi4-mini	Phi4-mm + audio input through Azure speech to text + audio output through Azure text to speech voices including custom voice.

Comparing Voice live API with other speech to speech solutions

The Voice live API is an alternative to orchestrating multiple components such as speech recognition, generative AI, and text to speech. This orchestration can be complex and time-consuming, requiring significant engineering effort to integrate and maintain. The Voice live API simplifies this process by providing a single interface for all these components, allowing developers to focus on building their applications rather than managing the underlying infrastructure.

To meet your requirements, you can either build your own solution or use the Voice live API. This table compares the approaches:

[Expand table](#)

Application requirement	Do it yourself	Voice live API
Broad locale coverage with high accuracy (audio input)	✓	✓
Maintain brand and character personality (audio output)	✓	✓
Conversational enhancements	✗	✓
Choice of generative AI models	✓	✓
Visual output with text to speech avatar	✓	✓
Low engineering cost	✗	✓
Low latency perceived by end user	✗	✓

Pricing

Pricing for the Voice live API is in effect from July 1, 2025.

Pricing for the Voice live API is tiered (**Pro**, **Basic**, and **Lite**) based on the generative AI model used.

You don't select a tier. You choose a generative AI model and the corresponding pricing applies.

 Expand table

Pricing category	Models
Voice live pro	<code>gpt-realtime</code> , <code>gpt-4o</code> , <code>gpt-4.1</code> , <code>gpt-5</code> , <code>gpt-5-chat</code>
Voice live basic	<code>gpt-realtime-mini</code> , <code>gpt-4o-mini</code> , <code>gpt-4.1-mini</code> , <code>gpt-5-mini</code>
Voice live lite	<code>gpt-5-nano</code> , <code>phi4-mm-realtime</code> , <code>phi4-mini</code>

If you choose to use custom speech, custom voice or custom avatar for your speech input and/or output, you're charged separately for model training and hosting. Refer to the [Speech Services Pricing](#) for details.

 **Important**

Custom voice access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

 **Important**

Custom text to speech avatar access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

Example pricing scenarios

Here are some example pricing scenarios to help you understand how the Voice live API is charged:

Scenario 1

A customer service agent built with standard Azure AI Speech input, GPT-4.1, custom Azure AI Speech output, and a custom avatar.

You're charged at the voice live pro rate for:

- Text
- Audio with Azure AI Speech - Standard
- Audio with Azure AI Speech - Custom

You're charged separately for the training and model hosting of:

- Custom voice – professional
- Custom avatar

Scenario 2

A learning agent built with `gpt-realtime` native audio input and standard Azure AI Speech output.

You're charged at the voice live pro rate for:

- Text
- Native audio with `gpt-realtime`
- Audio with Azure AI Speech - Standard

Scenario 3

A talent interview agent built with `gpt-realtime-mini` native audio input, and standard Azure AI Speech output and standard avatar.

You're charged at the voice live basic rate for:

- Text
- Native audio with `gpt-realtime-mini`
- Audio with Azure AI Speech - Standard

You're charged separately for:

- Text to speech avatar (standard)

Scenario 4

An in-car assistant built with `phi4-mm-realtime` and Azure custom voice.

You're charged at the voice live lite rate for:

- Text
- Native audio with `phi4-mm-realtime`

You're charged at the voice live pro rate for:

- Audio with Azure AI Speech - Custom

You're charged separately for the training and model hosting of:

- Custom voice – professional

Token usage and cost estimation

Tokens are the units that generative AI models use to process input and generate output.

You can estimate token usage for different model families with the Voice live API based on audio length. The following token calculations apply to each model family:

[] Expand table

Model family	Input audio (tokens per second)	Output audio (tokens per second)
Azure OpenAI models	~10 tokens	~20 tokens
Phi models	~12.5 tokens	~20 tokens

You're also charged for cached audio and text inputs, including the prompt and the context of the conversations.

Related content

- Learn more about [How to use the Voice live API](#)
- Try out the [Voice live API quickstart](#)
- See the [Voice live API reference](#)

Voice live API supported languages

Introduction

The Voice live API supports multiple languages and configuration options. In this document, you learn which languages the Voice live API supports and how to configure them.

Speech input

Depending on which model is being used voice live speech input is processed either by one of the multimodal models (for example, `gpt-realtime`, `gpt-realtime-mini`, and `phi4-mm-realtime`) or by `azure speech to text` models.

Azure speech to text supported languages

Azure speech to text is used for all configuration where a non-multimodal model is being used and for speech input transcriptions with `phi4-mm-realtime`. It supports all languages documented on the [Language and voice support for the Speech service - Speech to text](#) tab.

There are three options for voice live language processing:

- Automatic multilingual configuration using multilingual model (default)
- Single language configuration
- Multilingual configuration using up to 10 defined languages

The current multi-lingual model supports the following languages:

- Chinese (China) [zh-CN]
- English (Australia) [en-AU]
- English (Canada) [en-CA]
- English (India) [en-IN]
- English (United Kingdom) [en-GB]
- English (United States) [en-US]
- French (Canada) [fr-CA]
- French (France) [fr-FR]
- German (Germany) [de-DE]
- Hindi (India) [hi-IN]
- Italian (Italy) [it-IT]
- Japanese (Japan) [ja-JP]
- Korean (Korea) [ko-KR]

- Spanish (Mexico) [es-MX]
- Spanish (Spain) [es-ES]

To use **Automatic multilingual configuration using multilingual model** no extra configuration is required. If you do add the `language` string to the session `session.update` message, make sure to leave it empty.

JSON

```
{  
    "session": {  
        "input_audio_transcription": {  
            "model": "azure-speech",  
            "language": ""  
        }  
    }  
}
```

⚠ Note

The multilingual model generates results for unsupported languages, if no language is defined. In these cases transcription, quality is low. Ensure to configure defined languages, if you're setting up application with languages unsupported by the multilingual model.

To configure a single or multiple languages not supported by the multimodal model, you must add them to the `language` string in the session `session.update` message. A maximum of 10 languages are supported.

JSON

```
{  
    "session": {  
        "input_audio_transcription": {  
            "model": "azure-speech",  
            "language": "en-US,fr-FR,de-DE"  
        }  
    }  
}
```

gpt-realtime and gpt-realtime-mini supported languages

While the underlying model was trained on 98 languages, OpenAI only lists the languages that exceeded <50% word error rate (WER) which is an industry standard benchmark for speech to text model accuracy. The model returns results for languages not listed but the quality will be low.

The following languages are supported by `gpt-realtime` and `gpt-realtime-mini`:

- Afrikaans
- Arabic
- Armenian
- Azerbaijani
- Belarusian
- Bosnian
- Bulgarian
- Catalan
- Chinese
- Croatian
- Czech
- Danish
- Dutch
- English
- Estonian
- Finnish
- French
- Galician
- German
- Greek
- Hebrew
- Hindi
- Hungarian
- Icelandic
- Indonesian
- Italian
- Japanese
- Kannada
- Kazakh
- Korean
- Latvian
- Lithuanian
- Macedonian
- Malay
- Marathi
- Maori
- Nepali
- Norwegian
- Persian

- Polish
- Portuguese
- Romanian
- Russian
- Serbian
- Slovak
- Slovenian
- Spanish
- Swahili
- Swedish
- Tagalog
- Tamil
- Thai
- Turkish
- Ukrainian
- Urdu
- Vietnamese
- Welsh

Multimodal models don't require a language configuration for the general processing. If you configure input audio transcription, you can provide the transcription models with a language hint to improve transcription quality. In this case you need to add the `language` string to the session `session.update` message.

JSON

```
{
  "session": {
    "input_audio_transcription": {
      "model": "gpt-4o-transcribe",
      "language": "English, German, French"
    }
}
```

ⓘ Note

Multimodal gpt models only support the following transcription models: `whisper-1`, `gpt-4o-transcribe`, and `gpt-4o-mini-transcribe`.

phi4-mm-realtime supported languages

The following languages are supported by `phi4-mm-realtime`:

- Chinese
- English
- French
- German
- Italian
- Japanese
- Portuguese
- Spanish

Multimodal models don't require a language configuration for the general processing. If you configure input audio transcription for `phi4-mm-realtime` you need to use the same configuration as for all non-mulitmodal model configuration where `azure-speech` is used for transcription as described.

 **Note**

Multimodal phi models only support the following transcription models: `azure-speech`.

Last updated on 10/31/2025

Quickstart: Create a voice live real-time voice agent

09/29/2025

In this article, you learn how to use voice live with generative AI and Azure AI Speech in the [Azure AI Foundry portal](#).

You create and run an application to use voice live directly with generative AI models for real-time voice agents.

- Using models directly allows specifying custom instructions (prompts) for each session, offering more flexibility for dynamic or experimental use cases.
- Models may be preferable when you want fine-grained control over session parameters or need to frequently adjust the prompt or configuration without updating an agent in the portal.
- The code for model-based sessions is simpler in some respects, as it does not require managing agent IDs or agent-specific setup.
- Direct model use is suitable for scenarios where agent-level abstraction or built-in logic is unnecessary.

To instead use the Voice live API with agents, see the [Voice live API agents quickstart](#).

Prerequisites

- An Azure subscription. [Create one for free](#).
- An [Azure AI Foundry resource](#) created in one of the supported regions. For more information about region availability, see the [voice live overview documentation](#).

💡 Tip

To use voice live, you don't need to deploy an audio model with your Azure AI Foundry resource. Voice live is fully managed, and the model is automatically deployed for you. For more information about models availability, see the [voice live overview documentation](#).

Try out voice live in the Speech playground

To try out the voice live demo, follow these steps:

1. Go to your project in [Azure AI Foundry](#).
2. Select **Playgrounds** from the left pane.
3. In the **Speech** playground tile, select **Try the Speech playground**.
4. Select **Speech capabilities by scenario > Voice live**.

← Speech Playground ▾

[View code](#) [View documentation](#) [Fine-tune](#)

All Speech to text Text to speech Speech capabilities by scenario

Video translation
Seamlessly translate and generate videos in multiple languages automatically.

Voice live Preview
Empower your agents with real-time, customized voices, avatars and a whole suite of conversational enhancements, using a single API.

5. Select a sample scenario, such as **Casual chat**.

← Speech Playground ▾

Customer service
Talk to Andrew, a travel agent who offers assistance for international travel.

Casual chat
Chat with Zara, your personal assistant.

Try with your own
Start from blank
Create your own voice agent by customizing the speech input, generative AI, and speech output settings.

Just say the word
Try speaking out loud, just like you would converse with a real person, and hear the responses you get back.

You are Zara, a human-like AI character developed by Contoso Company in 2025.
You're a good listener and a concise...

Response temperature ⓘ 0.9

Proactive engagement ⓘ

Speech

Language

Voice
Emma2 Dragon HD Latest

Voice temperature ⓘ 0.8

Speaking rate ⓘ

Start

6. Select **Start** to start chatting with the chat agent.
7. Select **End** to end the chat session.

8. Select a new generative AI model from the drop-down list via **Configuration > GenAI > Generative AI model**.

① Note

You can also select an agent that you configured in the **Agents** playground. For more information, see the [voice live with Foundry agents quickstart](#).

The screenshot shows the Speech Playground interface. On the left, there are several options: Language learning, Customer service, Casual chat (which is selected), and Try with your own. Under Try with your own, there is a Start from blank option. In the center, there is a large circular icon with a robot head inside, labeled "Just say the word". To the right, under the heading "GenAI", the "Generative AI model" dropdown is open, showing "GPT-4o Mini Realtime" as the selected option. Other options include GPT-4o Realtime, GPT-4o Mini Realtime, GPT-4o, GPT-4o Mini, and Bring my agent. Below the dropdown are sliders for Response temperature (set to 0.9) and Proactive engagement (set to on). Under the "Speech" section, there is a Language dropdown set to Auto-detect, a Voice dropdown showing "Emma2 Dragon HD Latest", and a "Create custom voice" button.

9. Edit other settings as needed, such as the **Response instructions**, **Voice**, and **Speaking rate**.

10. Select **Start** to start speaking again and select **End** to end the chat session.

Related content

- Try the [Voice live agents quickstart](#)
- Learn more about [How to use the Voice live API](#)
- See the [Voice live API reference](#)

Quickstart: Create a voice live real-time voice agent with Azure AI Foundry Agent Service (Preview)

Note

This feature is currently in public preview. This preview is provided without a service-level agreement, and is not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

In this article, you learn how to use voice live with [Azure AI Foundry Agent Service](#) and [Azure AI Speech](#) in the [Azure AI Foundry portal](#).

You can create and run an application to use voice live with agents for real-time voice agents.

- Using agents allows leveraging a built-in prompt and configuration managed within the agent itself, rather than specifying instructions in the session code.
- Agents encapsulate more complex logic and behaviors, making it easier to manage and update conversational flows without changing the client code.
- The agent approach streamlines integration. The agent ID is used to connect and all necessary settings are handled internally, reducing the need for manual configuration in the code.
- This separation also supports better maintainability and scalability for scenarios where multiple conversational experiences or business logic variations are needed.

To instead use the Voice live API without agents, see the [Voice live API quickstart](#).

Prerequisites

- An Azure subscription. [Create one for free](#).
- An [Azure AI Foundry resource](#) created in one of the supported regions. For more information about region availability, see the [voice live overview documentation](#).
- An Azure AI Foundry agent created in the [Azure AI Foundry portal](#). For more information about creating an agent, see the [Create an agent quickstart](#).

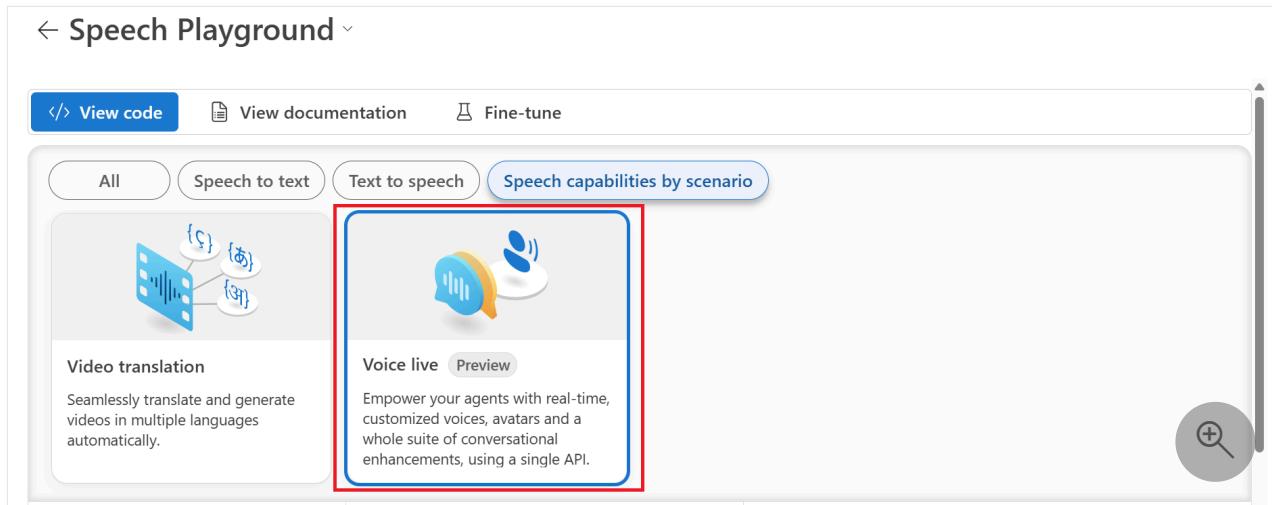
Tip

To use voice live, you don't need to deploy an audio model with your Azure AI Foundry resource. Voice live is fully managed, and the model is automatically deployed for you. For more information about models availability, see the [voice live overview documentation](#).

Try out voice live in the Speech playground

To try out the voice live demo, follow these steps:

1. Go to your project in [Azure AI Foundry](#).
2. Select **Playgrounds** from the left pane.
3. In the **Speech playground** tile, select **Try the Speech playground**.
4. Select **Speech capabilities by scenario > Voice live**.



5. Select an agent that you configured in the **Agents playground**.

← Speech Playground

Language learning
Chat with Ava, a learning companion who assists students in learning English as a second language.

Customer service
Talk to Andrew, a travel agent who offers assistance for international travel.

Casual chat
Chat with Zara, your personal assistant.

Try with your own

Start from blank
Create your own voice agent by customizing the speech input, generative AI, and speech output settings.



Just say the word

Try speaking out loud, just like you would converse with a real person, and hear the responses you get back.

GenAI

Generative AI model ⓘ
Bring my agent

GPT-4o Realtime
GPT-4o Mini Realtime
GPT-4o
GPT-4o Mini
✓ Bring my agent

Language
Auto-detect

Voice
Emma2 Dragon HD Latest
Create custom voice

Voice temperature ⓘ 0.8

Speaking rate 1

Voice activity detection (VAD)
Azure semantic VAD

6. Edit other settings as needed, such as the **Voice**, **Speaking rate**, and **Voice activity detection (VAD)**.

7. Select **Start** to start speaking and select **End** to end the chat session.

Related content

- Try the [Voice live quickstart](#)
- Learn more about [How to use the Voice live API](#)
- See the [Voice live API reference](#)

Last updated on 10/31/2025

Quickstart: Use function calling in a Voice Live session

[Reference documentation](#) | [Package \(PyPi\)](#) ↗ | [Additional samples on GitHub](#) ↗

The Voice Live API supports function calling in voice conversations. This allows you to create voice assistants that can call external functions to get real-time information and include that information in their spoken responses.

Implementation steps

The code sample below does these basic steps to set up function calling.

1. **Write backend functions:** Define Python callables that fulfill business tasks (time lookup, weather, database queries) and serialize outputs to JSON-friendly dictionaries.
2. **Describe tools for Voice Live:** Create **FunctionTool** definitions with names, parameter schemas, and text descriptions, and bundle them into the session configuration so the model understands the available actions.
3. **Initialize the session:** Connect using `azure.ai.voicelive.aio.connect`, provide credentials, pass in the defined **FunctionTool**, choose your target model/voice, and enable audio modalities, transcription, and turn detection.
4. **Start audio processing:** Spin up **AudioProcessor** to capture microphone input, encode it (PCM16, 24 kHz), and stream it to the Voice Live connection; simultaneously prepare playback for assistant audio responses.
5. **Run the event loop:** Await Voice Live events, updating session state, reacting to user speech boundaries, and streaming the assistant's audio/text back to the user interface. When a **ResponseFunctionCallItem** arrives, the application locates the callable, executes it with parsed arguments, packages the result into a **FunctionCallOutputItem**, and sends it back so the assistant can finalize its reply.

Sample code

Below is an example of async function calling with the Voice Live API - Python SDK. To run this code, start by implementing the setup steps in the [Quickstart](#).

For more information see the sample on [GitHub](#) ↗ .

Python

```
# -----
# Copyright (c) Microsoft Corporation. All rights reserved.
# Licensed under the MIT License.
# -----
from __future__ import annotations
import os
import sys
import argparse
import asyncio
import json
import base64
from datetime import datetime
import logging
import queue
import signal
from typing import Union, Optional, Dict, Any, Mapping, Callable, TYPE_CHECKING,
cast

from azure.core.credentials import AzureKeyCredential
from azure.core.credentials_async import AsyncTokenCredential
from azure.identity.aio import AzureCliCredential, DefaultAzureCredential

from azure.ai.vovcelive.aio import connect
from azure.ai.vovcelive.models import (
    AudioEchoCancellation,
    AudioNoiseReduction,
    AzureStandardVoice,
    InputAudioFormat,
    ItemType,
    Modality,
    OutputAudioFormat,
    RequestSession,
    ServerEventType,
    ServerVad,
    FunctionTool,
    FunctionCallOutputItem,
    ToolChoiceLiteral,
    AudioInputTranscriptionOptions,
    Tool,
)
from dotenv import load_dotenv
import pyaudio

if TYPE_CHECKING:
    from azure.ai.vovcelive.aio import VoiceLiveConnection

## Change to the directory where this script is located
os.chdir(os.path.dirname(os.path.abspath(__file__)))

# Environment variable loading
load_dotenv('./.env', override=True)

# Set up logging
## Add folder for logging
```

```

if not os.path.exists('logs'):
    os.makedirs('logs')

## Add timestamp for logfiles
timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")

## Set up logging
logging.basicConfig(
    filename=f'logs/{timestamp}_voicelive.log',
    filemode="w",
    format='%(asctime)s:%(name)s:%(levelname)s:%(message)s',
    level=logging.INFO
)
logger = logging.getLogger(__name__)

class AudioProcessor:
    """
    Handles real-time audio capture and playback for the voice assistant.

    Threading Architecture:
    - Main thread: Event loop and UI
    - Capture thread: PyAudio input stream reading
    - Send thread: Async audio data transmission to VoiceLive
    - Playback thread: PyAudio output stream writing
    """
    loop: asyncio.AbstractEventLoop

    class AudioPlaybackPacket:
        """Represents a packet that can be sent to the audio playback queue."""
        def __init__(self, seq_num: int, data: Optional[bytes]):
            self.seq_num = seq_num
            self.data = data

        def __init__(self, connection):
            self.connection = connection
            self.audio = pyaudio.PyAudio()

            # Audio configuration - PCM16, 24kHz, mono as specified
            self.format = pyaudio.paInt16
            self.channels = 1
            self.rate = 24000
            self.chunk_size = 1200 # 50ms

            # Capture and playback state
            self.input_stream = None

            self.playback_queue: queue.Queue[AudioProcessor.AudioPlaybackPacket] =
queue.Queue()
            self.playback_base = 0
            self.next_seq_num = 0
            self.output_stream: Optional[pyaudio.Stream] = None

        logger.info("AudioProcessor initialized with 24kHz PCM16 mono audio")

```

```
def start_capture(self):
    """Start capturing audio from microphone."""
    def _capture_callback(
        in_data,      # data
        _frame_count, # number of frames
        _time_info,   # dictionary
        _status_flags):
        """Audio capture thread - runs in background."""
        audio_base64 = base64.b64encode(in_data).decode("utf-8")
        asyncio.run_coroutine_threadsafe(
            self.connection.input_audio_buffer.append(audio=audio_base64),
self.loop
        )
    return (None, pyaudio.paContinue)

if self.input_stream:
    return

# Store the current event loop for use in threads
self.loop = asyncio.get_event_loop()

try:
    self.input_stream = self.audio.open(
        format=self.format,
        channels=self.channels,
        rate=self.rate,
        input=True,
        frames_per_buffer=self.chunk_size,
        stream_callback=_capture_callback,
    )
    logger.info("Started audio capture")

except Exception:
    logger.exception("Failed to start audio capture")
    raise

def start_playback(self):
    """Initialize audio playback system."""
    if self.output_stream:
        return

    remaining = bytes()
    def _playback_callback(
        _in_data,
        frame_count, # number of frames
        _time_info,
        _status_flags):

        nonlocal remaining
        frame_count *= pyaudio.get_sample_size(pyaudio.paInt16)

        out = remaining[:frame_count]
        remaining = remaining[frame_count:]
```

```

        while len(out) < frame_count:
            try:
                packet = self.playback_queue.get_nowait()
            except queue.Empty:
                out = out + bytes(frame_count - len(out))
                continue
            except Exception:
                logger.exception("Error in audio playback")
                raise

            if not packet or not packet.data:
                # None packet indicates end of stream
                logger.info("End of playback queue.")
                break

            if packet.seq_num < self.playback_base:
                # skip requested
                # ignore skipped packet and clear remaining
                if len(remaining) > 0:
                    remaining = bytes()
                continue

            num_to_take = frame_count - len(out)
            out = out + packet.data[:num_to_take]
            remaining = packet.data[num_to_take:]

            if len(out) >= frame_count:
                return (out, pyaudio.paContinue)
            else:
                return (out, pyaudio.paComplete)

        try:
            self.output_stream = self.audio.open(
                format=self.format,
                channels=self.channels,
                rate=self.rate,
                output=True,
                frames_per_buffer=self.chunk_size,
                stream_callback=_playback_callback
            )
            logger.info("Audio playback system ready")
        except Exception:
            logger.exception("Failed to initialize audio playback")
            raise

    def _get_and_increase_seq_num(self):
        seq = self.next_seq_num
        self.next_seq_num += 1
        return seq

    def queue_audio(self, audio_data: Optional[bytes]) -> None:
        """Queue audio data for playback."""
        self.playback_queue.put(
            AudioProcessor.AudioPlaybackPacket(
                seq_num=self._get_and_increase_seq_num(),

```

```
        data=audio_data))

def skip_pending_audio(self):
    """Skip current audio in playback queue."""
    self.playback_base = self._get_and_increase_seq_num()

def shutdown(self):
    """Clean up audio resources."""
    if self.input_stream:
        self.input_stream.stop_stream()
        self.input_stream.close()
        self.input_stream = None

    logger.info("Stopped audio capture")

    # Inform thread to complete
    if self.output_stream:
        self.skip_pending_audio()
        self.queue_audio(None)
        self.output_stream.stop_stream()
        self.output_stream.close()
        self.output_stream = None

    logger.info("Stopped audio playback")

    if self.audio:
        self.audio.terminate()

    logger.info("Audio processor cleaned up")



class AsyncFunctionCallingClient:
    """Voice assistant with function calling capabilities using VoiceLive SDK
    patterns."""

    def __init__(
        self,
        endpoint: str,
        credential: Union[AzureKeyCredential, AsyncTokenCredential],
        model: str,
        voice: str,
        instructions: str,
    ):
        self.endpoint = endpoint
        self.credential = credential
        self.model = model
        self.voice = voice
        self.instructions = instructions
        self.connection: Optional["VoiceLiveConnection"] = None
        self.audio_processor: Optional[AudioProcessor] = None
        self.session_ready = False
        self.conversation_started = False
        self._active_response = False
        self._response_api_done = False
```

```
    self._pending_function_call: Optional[Dict[str, Any]] = None

    # Define available functions
    self.available_functions: Dict[str, Callable[[Union[str, Mapping[str,
Any]]], Mapping[str, Any]]] = {
        "get_current_time": self.get_current_time,
        "get_current_weather": self.get_current_weather,
    }

async def start(self):
    """Start the voice assistant session."""
    try:
        logger.info("Connecting to VoiceLive API with model %s", self.model)

        # Connect to VoiceLive WebSocket API
        async with connect(
            endpoint=self.endpoint,
            credential=self.credential,
            model=self.model,
        ) as connection:
            conn = connection
            self.connection = conn

        # Initialize audio processor
        ap = AudioProcessor(conn)
        self.audio_processor = ap

        # Configure session for voice conversation
        await self._setup_session()

        # Start audio systems
        ap.start_playback()

        logger.info("Voice assistant with function calling ready! Start
speaking...")
        print("\n" + "=" * 60)
        print("MIC VOICE ASSISTANT WITH FUNCTION CALLING READY")
        print("Try saying:")
        print("• What's the current time?")
        print("• What's the weather in Seattle?")
        print("Press Ctrl+C to exit")
        print("=" * 60 + "\n")

        # Process events
        await self._process_events()
    finally:
        if self.audio_processor:
            self.audio_processor.shutdown()

async def _setup_session(self):
    """Configure the VoiceLive session for audio conversation with function
tools."""
    logger.info("Setting up voice conversation session with function tools...")

    # Create voice configuration
```

```
voice_config: Union[AzureStandardVoice, str]
    if self.voice.startswith("en-US-") or self.voice.startswith("en-CA-") or "-"
in self.voice:
    # Azure voice
    voice_config = AzureStandardVoice(name=self.voice)
else:
    # OpenAI voice (alloy, echo, fable, onyx, nova, shimmer)
    voice_config = self.voice

# Create turn detection configuration
turn_detection_config = ServerVad(
    threshold=0.5,
    prefix_padding_ms=300,
    silence_duration_ms=500)

# Define function tools
function_tools: list[Tool] = [
    FunctionTool(
        name="get_current_time",
        description="Get the current time",
        parameters={
            "type": "object",
            "properties": {
                "timezone": {
                    "type": "string",
                    "description": "The timezone to get the current time
for, e.g., 'UTC', 'local'",
                }
            },
            "required": []
        },
    ),
    FunctionTool(
        name="get_current_weather",
        description="Get the current weather in a given location",
        parameters={
            "type": "object",
            "properties": {
                "location": {
                    "type": "string",
                    "description": "The city and state, e.g., 'San
Francisco, CA'",
                }
            },
            "unit": {
                "type": "string",
                "enum": ["celsius", "fahrenheit"],
                "description": "The unit of temperature to use (celsius
or fahrenheit)",
            }
        },
        "required": ["location"],
    ),
]
```

```

# Create session configuration with function tools
session_config = RequestSession(
    modalities=[Modality.TEXT, Modality.AUDIO],
    instructions=self.instructions,
    voice=voice_config,
    input_audio_format=InputAudioFormat.PCM16,
    output_audio_format=OutputAudioFormat.PCM16,
    turn_detection=turn_detection_config,
    input_audio_echo_cancellation=AudioEchoCancellation(),

input_audio_noise_reduction=AudioNoiseReduction(type="azure_deep_noise_suppression")
,
    tools=function_tools,
    tool_choice=ToolChoiceLiteral.AUTO,
    input_audio_transcription=AudioInputTranscriptionOptions(model="whisper-
1"),
)

conn = self.connection
assert conn is not None, "Connection must be established before setting up
session"
await conn.session.update(session=session_config)

logger.info("Session configuration with function tools sent")

async def _process_events(self):
    """Process events from the VoiceLive connection."""
    try:
        conn = self.connection
        assert conn is not None, "Connection must be established before
processing events"
        async for event in conn:
            await self._handle_event(event)
    except Exception:
        logger.exception("Error processing events")
        raise

async def _handle_event(self, event):
    """Handle different types of events from VoiceLive."""
    logger.debug("Received event: %s", event.type)
    ap = self.audio_processor
    conn = self.connection
    assert ap is not None, "AudioProcessor must be initialized"
    assert conn is not None, "Connection must be established"

    if event.type == ServerEventType.SESSION_UPDATED:
        logger.info("Session ready: %s", event.session.id)
        self.session_ready = True

        # Proactive greeting
        if not self.conversation_started:
            self.conversation_started = True
            logger.info("Sending proactive greeting request")
            try:
                await conn.response.create()

```

```

        except Exception:
            logger.exception("Failed to send proactive greeting request")

        # Start audio capture once session is ready
        ap.start_capture()

    elif event.type == ServerEventType.INPUT_AUDIO_BUFFER_SPEECH_STARTED:
        logger.info("User started speaking - stopping playback")
        print("🎙️ Listening...")

        ap.skip_pending_audio()

        # Only cancel if response is active and not already done
        if self._active_response and not self._response_api_done:
            try:
                await conn.response.cancel()
                logger.debug("Cancelled in-progress response due to barge-in")
            except Exception as e:
                if "no active response" in str(e).lower():
                    logger.debug("Cancel ignored - response already completed")
                else:
                    logger.warning("Cancel failed: %s", e)

    elif event.type == ServerEventType.INPUT_AUDIO_BUFFER_SPEECH_STOPPED:
        logger.info("🎙️ User stopped speaking")
        print("🤔 Processing...")

    elif event.type == ServerEventType.RESPONSE_CREATED:
        logger.info("🤖 Assistant response created")
        self._active_response = True
        self._response_api_done = False

    elif event.type == ServerEventType.RESPONSE_AUDIO_DELTA:
        logger.debug("Received audio delta")
        ap.queue_audio(event.delta)

    elif event.type == ServerEventType.RESPONSE_AUDIO_DONE:
        logger.info("🤖 Assistant finished speaking")
        print("🎙️ Ready for next input...")

    elif event.type == ServerEventType.RESPONSE_DONE:
        logger.info("✅ Response complete")
        self._active_response = False
        self._response_api_done = True

        # Execute pending function call if arguments are ready
        if self._pending_function_call and "arguments" in
self._pending_function_call:
            await self._execute_function_call(self._pending_function_call)
            self._pending_function_call = None

    elif event.type == ServerEventType.ERROR:
        msg = event.error.message
        if "Cancellation failed: no active response" in msg:

```

```

        logger.debug("Benign cancellation error: %s", msg)
    else:
        logger.error("✖ VoiceLive error: %s", msg)
        print(f"Error: {msg}")

    elif event.type == ServerEventType.CONVERSATION_ITEM_CREATED:
        logger.debug("Conversation item created: %s", event.item.id)

        if event.item.type == ItemType.FUNCTION_CALL:
            function_call_item = event.item
            self._pending_function_call = {
                "name": function_call_item.name,
                "call_id": function_call_item.call_id,
                "previous_item_id": function_call_item.id
            }
            print(f"🔧 Calling function: {function_call_item.name}")
            logger.info(f"Function call detected: {function_call_item.name} with
call_id: {function_call_item.call_id}")

    elif event.type == ServerEventType.RESPONSE_FUNCTION_CALL_ARGUMENTS_DONE:
        if self._pending_function_call and event.call_id ==
self._pending_function_call["call_id"]:
            logger.info(f"Function arguments received: {event.arguments}")
            self._pending_function_call["arguments"] = event.arguments

    async def _execute_function_call(self, function_call_info):
        """Execute a function call and send the result back to the conversation."""
        conn = self.connection
        assert conn is not None, "Connection must be established"

        function_name = function_call_info["name"]
        call_id = function_call_info["call_id"]
        previous_item_id = function_call_info["previous_item_id"]
        arguments = function_call_info["arguments"]

        try:
            if function_name in self.available_functions:
                logger.info(f"Executing function: {function_name}")
                result = self.available_functions[function_name](arguments)

                function_output = FunctionCallOutputItem(call_id=call_id,
output=json.dumps(result))

                # Send result back to conversation
                await
conn.conversation.item.create(previous_item_id=previous_item_id,
item=function_output)
                logger.info(f"Function result sent: {result}")
                print(f"✅ Function {function_name} completed")

                # Request new response to process the function result
                await conn.response.create()
                logger.info("Requested new response with function result")

        else:

```

```
    logger.error(f"Unknown function: {function_name}")

except Exception as e:
    logger.error(f"Error executing function {function_name}: {e}")

def get_current_time(self, arguments: Optional[Union[str, Mapping[str, Any]]] = None) -> Dict[str, Any]:
    """Get the current time."""
    from datetime import datetime, timezone

    if isinstance(arguments, str):
        try:
            args = json.loads(arguments)
        except json.JSONDecodeError:
            args = {}
    else:
        args = arguments if isinstance(arguments, dict) else {}

    timezone_arg = args.get("timezone", "local")
    now = datetime.now()

    if timezone_arg.lower() == "utc":
        now = datetime.now(timezone.utc)
        timezone_name = "UTC"
    else:
        timezone_name = "local"

    formatted_time = now.strftime("%I:%M:%S %p")
    formatted_date = now.strftime("%A, %B %d, %Y")

    return {"time": formatted_time, "date": formatted_date, "timezone": timezone_name}

def get_current_weather(self, arguments: Union[str, Mapping[str, Any]]):

    """Get the current weather for a location."""

    if isinstance(arguments, str):
        try:
            args = json.loads(arguments)
        except json.JSONDecodeError:
            logger.error(f"Failed to parse weather arguments: {arguments}")
            return {"error": "Invalid arguments"}
    else:
        args = arguments if isinstance(arguments, dict) else {}

    location = args.get("location", "Unknown")
    unit = args.get("unit", "celsius")

    # Simulated weather response
    try:
        return {
            "location": location,
            "temperature": 22 if unit == "celsius" else 72,
            "unit": unit,
            "condition": "Partly Cloudy",
            "humidity": 65,
        }
    except Exception as e:
        logger.error(f"Error executing function {function_name}: {e}")
```

```
        "wind_speed": 10,
    }
except Exception as e:
    logger.error(f"Error getting weather: {e}")
    return {"error": str(e)}

def parse_arguments():
    """Parse command line arguments."""
    parser = argparse.ArgumentParser(
        description="Voice Assistant with Function Calling using Azure VoiceLive
SDK",
        formatter_class=argparse.ArgumentDefaultsHelpFormatter,
    )

    parser.add_argument(
        "--api-key",
        help="Azure VoiceLive API key. If not provided, will use
AZURE_VOICELIVE_API_KEY environment variable.",
        type=str,
        default=os.environ.get("AZURE_VOICELIVE_API_KEY"),
    )

    parser.add_argument(
        "--endpoint",
        help="Azure VoiceLive endpoint",
        type=str,
        default=os.environ.get("AZURE_VOICELIVE_ENDPOINT", "https://your-resource-
name.services.ai.azure.com/"),
    )

    parser.add_argument(
        "--model",
        help="VoiceLive model to use",
        type=str,
        default=os.environ.get("AZURE_VOICELIVE_MODEL", "gpt-realtime"),
    )

    parser.add_argument(
        "--voice",
        help="Voice to use for the assistant. E.g. alloy, echo, fable, en-US-
AvaNeural, en-US-GuyNeural",
        type=str,
        default=os.environ.get("AZURE_VOICELIVE_VOICE", "en-US-
Ava:DragonHDLatestNeural"),
    )

    parser.add_argument(
        "--instructions",
        help="System instructions for the AI assistant",
        type=str,
        default=os.environ.get(
            "AZURE_VOICELIVE_INSTRUCTIONS",
            "You are a helpful AI assistant with access to functions. "
            "Use the functions when appropriate to provide accurate, real-time
            "
        )
    )
```

```
information. "
        "If you are asked about the weather, please respond with 'I will get the
weather for you. Please wait a moment.' and then call the get_current_weather
function. "
        "If you are asked about the time, please respond with 'I will get the
time for you. Please wait a moment.' and then call the get_current_time function. "
        "Explain when you're using a function and include the results in your
response naturally. Always start the conversation in English.",

    ),
)

parser.add_argument(
    "--use-token-credential", help="Use Azure token credential instead of API
key", action="store_true", default=False
)

parser.add_argument("--verbose", help="Enable verbose logging",
action="store_true")

return parser.parse_args()

def main():
    """Main function."""
    args = parse_arguments()

    # Set logging level
    if args.verbose:
        logging.getLogger().setLevel(logging.DEBUG)

    # Validate credentials
    if not args.api_key and not args.use_token_credential:
        print("✖ Error: No authentication provided")
        print("Please provide an API key using --api-key or set
AZURE_VOICELIVE_API_KEY environment variable,")
        print("or use --use-token-credential for Azure authentication.")
        sys.exit(1)

    # Create client with appropriate credential
    credential: Union[AzureKeyCredential, AsyncTokenCredential]
    if args.use_token_credential:
        credential = AzureCliCredential()
        logger.info("Using Azure token credential")
    else:
        credential = AzureKeyCredential(args.api_key)
        logger.info("Using API key credential")

    # Create and start voice assistant with function calling
    client = AsyncFunctionCallingClient(
        endpoint=args.endpoint,
        credential=credential,
        model=args.model,
        voice=args.voice,
        instructions=args.instructions,
    )
```

```

# Signal handlers for graceful shutdown
def signal_handler(_sig, _frame):
    logger.info("Received shutdown signal")
    raise KeyboardInterrupt()

signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)

try:
    asyncio.run(client.start())
except KeyboardInterrupt:
    print("\n👋 Voice assistant shut down. Goodbye!")
except Exception as e:
    logger.exception("Fatal error")
    print(f"Fatal Error: {e}")
    sys.exit(1)

if __name__ == "__main__":
    # Check for required dependencies
    dependencies = {
        "pyaudio": "Audio processing",
        "azure.ai.voicelive": "Azure VoiceLive SDK",
        "azure.core": "Azure Core libraries",
    }

    missing_deps = []
    for dep, description in dependencies.items():
        try:
            __import__(dep.replace("-", "_"))
        except ImportError:
            missing_deps.append(f"{dep} ({description})")

    if missing_deps:
        print("✖ Missing required dependencies:")
        for dep in missing_deps:
            print(f" - {dep}")
        print("\nInstall with: pip install azure-ai-voicelive pyaudio python-dotenv")
        sys.exit(1)

    # Check audio system
    try:
        p = pyaudio.PyAudio()
        # Check for input devices
        input_devices = [
            i
            for i in range(p.get_device_count())
            if cast(Union[int, float],
p.get_device_info_by_index(i).get("maxInputChannels", 0) or 0) > 0
        ]
        # Check for output devices
        output_devices = [
            i
            for i in range(p.get_device_count())

```

```
        if cast(Union[int, float],  
p.get_device_info_by_index(i).get("maxOutputChannels", 0) or 0) > 0  
    ]  
    p.terminate()  
  
    if not input_devices:  
        print("✖ No audio input devices found. Please check your microphone.")  
        sys.exit(1)  
    if not output_devices:  
        print("✖ No audio output devices found. Please check your speakers.")  
        sys.exit(1)  
  
except Exception as e:  
    print(f"✖ Audio system check failed: {e}")  
    sys.exit(1)  
  
print("VOICE ASSISTANT WITH FUNCTION CALLING - AZURE VOICELIVE SDK")  
print("=" * 65)  
  
# Run the assistant  
main()
```

Related content

- Try the [Voice live agents quickstart](#)
- Learn more about [How to use the Voice live API](#)
- See the [Voice live API reference](#)

Last updated on 11/07/2025

Bring Your Own Model (BYOM) with Voice Live API

The Voice Live API provides Bring Your Own Model (BYOM) capabilities, allowing you to integrate your custom models into the voice interaction workflow. BYOM is useful for the following scenarios:

- **Fine-tuned models:** Use your custom Azure OpenAI or Azure Foundry models
- **Provisioned throughput:** Use your PTU (Provisioned Throughput Units) deployments for consistent performance
- **Content safety:** Apply customized content safety configurations with your LLM

Important

You can integrate any model that was deployed in the same Azure Foundry resource you're using to call the Voice Live API.

Tip

When you use your own model deployment with Voice Live, we recommend you set its content filtering configuration to [Asynchronous filtering](#) to reduce latency. Content filtering settings can be configured in the [Azure AI Foundry portal](#).

Authentication setup

When using Microsoft Entra ID authentication with Voice Live API, in `byom-azure-openai-chat-completion` mode specifically, you need to configure proper permissions for your Foundry resource. Since tokens expire during long sessions, the system-assigned managed identity of the Foundry resource requires access to model deployments for the `byom-azure-openai-chat-completion` BYOM mode.

Run the following Azure CLI commands to configure the necessary permissions:

Bash

```
export subscription_id=<your-subscription-id>
export resource_group=<your-resource-group>
export foundry_resource=<your-foundry-resource>

# Enable system-assigned managed identity for the foundry resource
az cognitiveservices account identity assign --name ${foundry_resource} --resource-
```

```

group ${resource_group} --subscription ${subscription_id}

# Get the system-assigned managed identity object ID
identity_principal_id=$(az cognitiveservices account show --name ${foundry_resource}
--resource-group ${resource_group} --subscription ${subscription_id} --query
"identity.principalId" -o tsv)

# Assign the Azure AI User role to the system identity of the foundry resource
az role assignment create --assignee-object-id ${identity_principal_id} --role
"Azure AI User" --scope
/subscriptions/${subscription_id}/resourceGroups/${resource_group}/providers/Microsoft.CognitiveServices/accounts/${foundry_resource}

```

Choose BYOM integration mode

The Voice Live API supports two BYOM integration modes:

[Expand table](#)

Mode	Description	Example Models
byom-azure-openai-realtime	Azure OpenAI realtime models for streaming voice interactions	gpt-realtime, gpt-realtime-mini
byom-azure-openai-chat-completion	Azure OpenAI chat completion models for text-based interactions. Also applies to other Foundry models	gpt-4.1, gpt-5-chat, grok-3

Integrate BYOM

REST API call

Update the endpoint URL in your API call to include your BYOM configuration:

curl

```
wss://<your-foundry-resource>.cognitiveservices.azure.com/voice-live/realtime?
api-version=2025-10-01&profile=<your-byom-mode>&model=<your-model-deployment>
```

Get the <your-model-deployment> value from the AI Foundry portal. It corresponds to the name you gave the model at deployment time.

Related content

- Try the [Voice live quickstart](#)
- Learn more about [How to use the Voice live API](#)
- See the [Voice live API reference](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

Last updated on 11/10/2025

How to use the Voice live API

The Voice live API provides a capable WebSocket interface compared to the [Azure OpenAI Realtime API](#).

Unless otherwise noted, the Voice live API uses the [same events](#) as the Azure OpenAI Realtime API. This document provides a reference for the event message properties that are specific to the Voice live API.

Supported models and regions

For a table of supported models and regions, see the [Voice live API overview](#).

Authentication

An [Azure AI Foundry resource](#) or a [Azure AI Speech Services resource](#) is required to use the Voice live API.

! Note

Using Voice Live API is optimized for Azure AI Foundry resources. We recommend using Azure AI Foundry resources for full feature availability and best Azure AI Foundry integration experience.

Azure AI Speech Services resources don't support Azure AI Foundry Agent Service integration and bring-your-own-model (BYOM).

WebSocket endpoint

The WebSocket endpoint for the Voice live API is `wss://<your-ai-foundry-resource-name>.services.ai.azure.com/voice-live/realtime?api-version=2025-10-01` or, for older resources, `wss://<your-ai-foundry-resource-name>.cognitiveservices.azure.com/voice-live/realtime?api-version=2025-10-01`. The endpoint is the same for all models. The only difference is the required `model` query parameter, or, when using the Agent service, the `agent_id` and `project_id` parameters.

For example, an endpoint for a resource with a custom domain would be `wss://<your-ai-foundry-resource-name>.services.ai.azure.com/voice-live/realtime?api-version=2025-10-01&model=gpt-realtime`

Credentials

The Voice live API supports two authentication methods:

- **Microsoft Entra** (recommended): Use token-based authentication for an Azure AI Foundry resource. Apply a retrieved authentication token using a `Bearer` token with the `Authorization` header.
- **API key**: An `api-key` can be provided in one of two ways:
 - Using an `api-key` connection header on the prehandshake connection. This option isn't available in a browser environment.
 - Using an `api-key` query string parameter on the request URI. Query string parameters are encrypted when using `https/wss`.

For the recommended keyless authentication with Microsoft Entra ID, you need to:

- Assign the `Cognitive Services User` role to your user account or a managed identity. You can assign roles in the Azure portal under **Access control (IAM)** > **Add role assignment**.
- Generate a token using the Azure CLI or Azure SDKs. The token must be generated with the `https://ai.azure.com/.default` scope, or the legacy `https://cognitiveservices.azure.com/.default` scope.
- Use the token in the `Authorization` header of the WebSocket connection request, with the format `Bearer <token>`.

Session configuration

Often, the first event sent by the caller on a newly established Voice live API session is the `session.update` event. This event controls a wide set of input and output behavior, with output and response generation properties then later overridable using the `response.create` event.

Here's an example `session.update` message that configures several aspects of the session, including turn detection, input audio processing, and voice output. Most session parameters are optional and can be omitted if not needed.

JSON

```
{  
    "instructions": "You are a helpful AI assistant responding in natural, engaging  
language.",  
    "turn_detection": {  
        "type": "azure_semantic_vad",  
        "threshold": 0.3,  
        "prefix_padding_ms": 200,  
        "silence_duration_ms": 200,  
        "remove_filler_words": false,  
    },  
}
```

```

    "end_of_utterance_detection": {
      "model": "semantic_detection_v1",
      "threshold_level": "default",
      "timeout_ms": 1000
    },
  },
  "input_audio_noise_reduction": {"type": "azure_deep_noise_suppression"},  

  "input_audio_echo_cancellation": {"type": "server_echo_cancellation"},  

  "voice": {
    "name": "en-US-Ava:DragonHDLatestNeural",
    "type": "azure-standard",
    "temperature": 0.8,
  },
}

```

Important

The `"instructions"` property isn't supported when you're using a custom agent.

The server responds with a `session.updated` event to confirm the session configuration.

Session Properties

The following sections describe the properties of the `session` object that can be configured in the `session.update` message.

Tip

For comprehensive descriptions of supported events and properties, see the [Azure OpenAI Realtime API events reference documentation](#). This document provides a reference for the event message properties that are enhancements via the Voice live API.

Input audio properties

You can use input audio properties to configure the input audio stream.

 Expand table

Property	Type	Required or optional	Description
<code>input_audio_sampling_rate</code>	integer	Optional	The sampling rate of the input audio.

Property	Type	Required or optional	Description
			The supported values are <code>16000</code> and <code>24000</code> . The default value is <code>24000</code> .
<code>input_audio_echo_cancellation</code>	object	Optional	<p>Enhances the input audio quality by removing the echo from the model's own voice without requiring any client-side echo cancellation.</p> <p>Set the <code>type</code> property of <code>input_audio_echo_cancellation</code> to enable echo cancellation.</p> <p>The supported value for <code>type</code> is <code>server_echo_cancellation</code>, which is used when the model's voice is played back to the end-user through a speaker, and the microphone picks up the model's own voice.</p>
<code>input_audio_noise_reduction</code>	object	Optional	<p>Enhances the input audio quality by suppressing or removing environmental background noise.</p> <p>Set the <code>type</code> property of <code>input_audio_noise_reduction</code> to enable noise suppression.</p> <p>The supported value for <code>type</code> is <code>azure_deep_noise_suppression</code>, which optimizes for speakers closest to the microphone.</p> <p>You can set this property to <code>near_field</code> or <code>far_field</code> if you're using the Azure OpenAI Realtime API.</p>

Here's an example of input audio properties in a session object:

JSON

```
{
  "input_audio_sampling_rate": 24000,
  "input_audio_noise_reduction": {"type": "azure_deep_noise_suppression"},
  "input_audio_echo_cancellation": {"type": "server_echo_cancellation"},
}
```

Noise suppression and echo cancellation

Noise suppression enhances the input audio quality by suppressing or removing environmental background noise. Noise suppression helps the model understand the end-user with higher accuracy and improves accuracy of signals like interruption detection and end-of-turn detection.

Server echo cancellation enhances the input audio quality by removing the echo from the model's own voice. In this way, client-side echo cancellation isn't required. Server echo cancellation is useful when the model's voice is played back to the end-user through a speaker. This helps avoiding the microphone picking up the model's own voice.

! Note

The service assumes the client plays response audio as soon as it receives them. If playback is delayed for more than two seconds, echo cancellation quality is impacted.

Conversational enhancements

The Voice live API offers conversational enhancements to provide robustness to the natural end-user conversation flow.

Turn Detection Parameters

Turn detection is the process of detecting when the end-user started or stopped speaking. The Voice live API builds on the Azure OpenAI Realtime API `turn_detection` property to configure turn detection. The `azure_semantic_vad` and `azure_multilingual_semantic_vad` types and the advanced `end_of_utterance_detection` are key differentiators between the Voice live API and the Azure OpenAI Realtime API.

[Expand table](#)

Property	Type	Required or optional	Description
<code>type</code>	string	Optional	<p>The type of turn detection system to use. Type <code>server_vad</code> detects start and end of speech based on audio volume.</p> <p>Type <code>semantic_vad</code> uses a semantic classifier to detect when the user has finished speaking, based on the words they have uttered. This type can only be used with the <code>gpt-realtime</code> and <code>gpt-realtime-</code></p>

Property	Type	Required or optional	Description
			<i>mini</i> models.
			Type <code>azure_semantic_vad</code> and <code>azure_semantic_vad_multilingual</code> also detects start and end of speech based on semantic meaning and can be used with <i>all models</i> . Further Azure semantic voice activity detection (VAD) can also improve turn detection by removing filler words to reduce the false alarm rate of barge-in.
			The default value is <code>server_vad</code> .
<code>threshold</code>	number	Optional	A higher threshold requires a higher confidence signal of the user trying to speak.
<code>prefix_padding_ms</code>	integer	Optional	The amount of audio, measured in milliseconds, to include before the start of speech detection signal.
<code>speech_duration_ms</code>	integer	Optional	The duration of user's speech audio required to start detection. If not set or under 80 ms, the detector uses a default value of 80 ms.
<code>silence_duration_ms</code>	integer	Optional	The duration of user's silence, measured in milliseconds, to detect the end of speech.
<code>remove_filler_words</code>	boolean	Optional	Determines whether to remove filler words to reduce the false alarm rate of barge-in. To enable it the property must be set to <code>true</code> . The detected filler words in English are <code>['ah', 'umm', 'mm', 'uh', 'huh', 'oh', 'yeah', 'hmm']</code> . The service ignores these words when there's an ongoing response. Remove filler words feature assumes the client plays response audio as soon as it receives them. The default value is <code>false</code> .
<code>languages</code>	string[]	Optional	Language will be used to improve the <code>remove_filler_words</code> accuracy by reducing the applied languages. The type <code>azure_semantic_vad</code> primarily supports English. Type <code>azure_semantic_vad_multilingual</code> is also available to support a wider variety of languages: English, Spanish, French, Italian, German (DE), Japanese, Portuguese, Chinese, Korean, Hindi. Other languages will be ignored.

Property	Type	Required or optional	Description
<code>create_response</code>	boolean	Optional	Enable or disable whether a response is generated.
<code>eagerness</code>	string	Optional	<p>This is a way to control how eager the model is to interrupt the user, tuning the maximum wait timeout. Only available with type <code>semantic_vad</code>. In transcription mode, even if the model doesn't reply, it affects how the audio is chunked.</p> <p>The following values are allowed:</p> <ul style="list-style-type: none"> - <code>auto</code> (default) is equivalent to <code>medium</code>, - <code>low</code> will let the user take their time to speak, - <code>high</code> will chunk the audio as soon as possible. <p>If you want the model to respond more often in conversation mode, or to return transcription events faster in transcription mode, you can set <code>eagerness</code> to <code>high</code>.</p> <p>On the other hand, if you want to let the user speak uninterrupted in conversation mode, or if you would like larger transcript chunks in transcription mode, you can set <code>eagerness</code> to <code>low</code>.</p>
<code>interrupt_response</code>	boolean	Optional	Enable or disable barge-in interruption (default: false). Only available with type <code>azure_semantic_vad</code> and <code>azure_semantic_vad_multilingual</code> .
<code>auto_truncate</code>	boolean	Optional	Auto-truncate on interruption (default: false).
<code>end_of_utterance_detection</code>	object	Optional	<p>Configuration for end of utterance detection. The Voice live API offers advanced end-of-turn detection to indicate when the end-user stopped speaking while allowing for natural pauses. End of utterance detection can significantly reduce premature end-of-turn signals without adding user-perceivable latency. End of utterance detection can be used with either VAD selection.</p> <p>Properties of <code>end_of_utterance_detection</code> include:</p> <ul style="list-style-type: none"> - <code>model</code>: The model to use for end of utterance detection. The supported values are: <ul style="list-style-type: none"> <code>semantic_detection_v1</code> supporting English. <code>semantic_detection_v1_multilingual</code> supporting English, Spanish, French, Italian, German (DE), Japanese, Portuguese, Chinese, Korean, Hindi. Other languages are bypassed. - <code>threshold_level</code>: Option setting for detection

Property	Type	Required or optional	Description
			threshold level (<code>low</code> , <code>medium</code> , <code>high</code> and <code>default</code>), the default equals <code>medium</code> setting. With a lower setting the probability the sentence is complete will be higher. - <code>timeout_ms</code> : Optional setting for maximum time in milliseconds to wait for more user speech. Defaults to 1000 ms.
			End of utterance detection currently doesn't support gpt-realtime, gpt-4o-mini-realtime, and phi4-mm-realtime.

Here's an example of end of utterance detection in a session object:

JSON

```
{
  "session": {
    "instructions": "You are a helpful AI assistant responding in natural,
engaging language.",
    "turn_detection": {
      "type": "azure_semantic_vad",
      "threshold": 0.3,
      "prefix_padding_ms": 300,
      "speech_duration_ms": 80,
      "silence_duration_ms": 500,
      "remove_filler_words": false,
      "end_of_utterance_detection": {
        "model": "semantic_detection_v1",
        "threshold_level": "default",
        "timeout_ms": 1000
      }
    }
  }
}
```

Audio input through Azure speech to text

Azure speech to text is automatically active when you're using a non-multimodal model like gpt-4o.

In order to explicitly configure it, you can set the `model` to `azure-speech` in `input_audio_transcription`. This can be useful to improve the recognition quality for specific

language situations. See [How to customize voice live input and output](#) learn more about speech input customization configuration.

JSON

```
{  
    "session": {  
        "input_audio_transcription": {  
            "model": "azure-speech",  
            "language": "en"  
        }  
    }  
}
```

Audio output through Azure text to speech

You can use the `voice` parameter to specify a standard or custom voice. The voice is used for audio output.

The `voice` object has the following properties:

[] [Expand table](#)

Property	Type	Required or optional	Description
<code>name</code>	string	Required	Specifies the name of the voice. For example, <code>en-US-AvaNeural</code> .
<code>type</code>	string	Required	Configuration of the type of Azure voice between <code>azure-standard</code> and <code>azure-custom</code> .
<code>temperature</code>	number	Optional	Specifies temperature applicable to Azure HD voices. Higher values provide higher levels of variability in intonation, prosody, etc.

See [How to customize voice live input and output](#) learn more about speech output customization configuration.

Azure standard voices

Here's a partial message example for a standard (`azure-standard`) voice:

JSON

```
{  
    "voice": {
```

```
        "name": "en-US-AvaNeural",
        "type": "azure-standard"
    }
}
```

For the full list of standard voices, see [Language and voice support for the Speech service](#).

Azure high definition voices

Here's an example `session.update` message for a standard high definition voice:

JSON

```
{
    "voice": {
        "name": "en-US-Ava:DragonHDLatestNeural",
        "type": "azure-standard",
        "temperature": 0.8 // optional
    }
}
```

For the full list of standard high definition voices, see [high definition voices documentation](#).

Note

High definition voices are currently supported in the following regions only: southeastasia, centralindia, swedencentral, westeurope, eastus, eastus2, westus2

Speaking rate

Use the `rate` string property to adjust the speaking speed for any standard Azure text to speech voices and custom voices.

The rate value should range from 0.5 to 1.5, with higher values indicating faster speeds.

JSON

```
{
    "voice": {
        "name": "en-US-Ava:DragonHDLatestNeural",
        "type": "azure-standard",
        "temperature": 0.8, // optional
        "rate": "1.2"
    }
}
```

Audio timestamps

When you use Azure voices, and `output_audio_timestamp_types` is configured, the service returns the `response.audio_timestamp.delta` in the response, and `response.audio_timestamp.done` when the all timestamps message are returned.

To configure the audio timestamps, you can set the `output_audio_timestamp_types` in the session.update message.

JSON

```
{  
  "session": {  
    "output_audio_timestamp_types": ["word"]  
  }  
}
```

Service returns the audio timestamps in the response when the audio is generated.

JSON

```
{  
  "event_id": "<event_id>",  
  "type": "response.audio_timestamp.delta",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0,  
  "audio_offset_ms": 490,  
  "audio_duration_ms": 387,  
  "text": "end",  
  "timestamp_type": "word"  
}
```

And a `response.audio_timestamp.done` message is sent when all timestamps are returned.

JSON

```
{  
  "event_id": "<event_id>",  
  "type": "response.audio_timestamp.done",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
}
```

Viseme

A viseme is the visual description of a phoneme in spoken language. It defines the position of the face and mouth while a person is speaking.

You can use Azure standard voice or Azure custom voice with `animation.outputs` set to `{"viseme_id"}`. The service returns the `response.animation_viseme.delta` in the response and `response.animation_viseme.done` when all viseme messages are returned.

💡 Tip

For more information about viseme via Speech Synthesis Markup Language (SSML), see [viseme element documentation](#).

To configure the viseme, you can set the `animation.outputs` in the `session.update` message. The `animation.outputs` parameter is optional. It configures which animation outputs should be returned. Currently, it only supports `viseme_id`.

JSON

```
{
  "type": "session.update",
  "event_id": "your-session-id",
  "session": {
    "voice": {
      "name": "en-US-AvaNeural",
      "type": "azure-standard",
    },
    "modalities": ["text", "audio"],
    "instructions": "You are a helpful AI assistant responding in natural, engaging language.",
    "turn_detection": {
      "type": "server_vad"
    },
    "output_audio_timestamp_types": ["word"], // optional
    "animation": {
      "outputs": ["viseme_id"], // optional
    },
  }
}
```

The `output_audio_timestamp_types` parameter is optional. It configures which audio timestamps should be returned for generated audio. Currently, it only supports `word`.

The service returns the viseme alignment in the response when the audio is generated.

JSON

```
{  
    "event_id": "<event_id>",  
    "type": "response.animation_viseme.delta",  
    "response_id": "<response_id>",  
    "item_id": "<item_id>",  
    "output_index": 0,  
    "content_index": 0,  
    "audio_offset_ms": 455,  
    "viseme_id": 20  
}
```

And a `response.animation_viseme.done` message is sent when all viseme messages are returned.

JSON

```
{  
    "event_id": "<event_id>",  
    "type": "response.animation_viseme.done",  
    "response_id": "<response_id>",  
    "item_id": "<item_id>",  
}
```

Azure text to speech avatar

[Text to speech avatar](#) converts text into a digital video of a photorealistic human (either a standard avatar or a [custom text to speech avatar](#)) speaking with a natural-sounding voice.

You can use the `avatar` parameter to specify a standard or custom avatar. The avatar is synchronized with the audio output.

An `avatar` parameter can be specified to enable avatar output that is synchronized with the audio output:

JSON

```
{  
    "session": {  
        "avatar": {  
            "character": "lisa",  
            "style": "casual-sitting",  
            "customized": false,  
            "ice_servers": [  
                {  
                    "urls": ["REDACTED"],  
                    "username": "",  
                    "credential": ""  
                }  
            ]  
        }  
    }  
}
```

```
],
  "video": {
    "bitrate": 2000000,
    "codec": "h264",
    "crop": {
      "top_left": [560, 0],
      "bottom_right": [1360, 1080],
    },
    "resolution": {
      "width": 1080,
      "height": 1920,
    },
    "background": {
      "color": "#00FF00FF"
      // "image_url": "https://example.com/example.jpg"
    }
  }
}
```

The `ice_servers` field is optional. If you don't specify it, the service returns the server-specific ICE servers in `session.updated` response. And you need to use the server-specific ICE servers to generate the local ICE candidates.

Send the client SDP after ICE candidates are gathered.

JSON

```
{
  "type": "session.avatar.connect",
  "client_sdp": "your-client-sdp"
}
```

And the service responds with the server SDP.

JSON

```
{
  "type": "session.avatar.connecting",
  "server_sdp": "your-server-sdp"
}
```

Then you can connect the avatar with the server SDP.

ⓘ Note

Azure text to speech avatar is currently supported in limited regions. For the current list of supported regions, see the [Speech service regions table](#).

Related content

- Try out the [Voice live API quickstart](#)
- See the [Voice live API reference](#)

Last updated on 11/05/2025

How to customize voice live input and output

Voice live provides multiple options to optimize performance and quality by using custom models. The following customization options are currently available:

- Speech input customization:
 - Phrase-list: A lightweight just-in-time customization based on a list of words or phrases provided as part of the session configuration to help improve recognition quality. To learn more, see [Improve recognition accuracy with phrase list](#).
 - Custom Speech: With custom speech, you can evaluate and improve the accuracy of speech recognition for your applications and products and fine-tune the recognition quality to your business needs. See [What is custom speech?](#) to learn more.
- Speech output customization:
 - Custom lexicon: Custom lexicon allows you to easily customize pronunciation for both standard Azure text to speech voices and custom voices to improve speech synthesis accuracy for your use case. See [custom lexicon for text to speech](#) to learn more.
 - Custom voice: Custom voice lets you create a one-of-a-kind, customized, synthetic voice for your applications. With custom voice, you can build a highly natural-sounding voice for your brand or characters by providing human speech samples as fine-tuning data. See [What is custom voice?](#) to learn more.
 - Custom avatar: Custom text to speech avatar allows you to create a customized, one-of-a-kind synthetic talking avatar for your application. With custom text to speech avatar, you can build a unique and natural-looking avatar for your product or brand by providing video recording data of your selected actors. See [What is custom text to speech avatar?](#) to learn more.

Speech input customization

Phrase list

Use phrase list for lightweight just-in-time customization on audio input. To configure phrase list, you can set the `phrase_list` in the `session.update` message.

JSON

```
{  
    "session": {  
        "input_audio_transcription": {  
            "model": "azure-speech",  
            "phrase_list": ["Neo QLED TV", "TUF Gaming", "AutoQuote Explorer"]  
        }  
    }  
}
```

```
    }
}
```

➊ Note

Phrase list currently doesn't support gpt-realtime, gpt-4o-mini-realtime, and phi4-mm-realtime. To learn more about phrase list, see [phrase list for speech to text](#).

Custom speech configuration

You can use the custom_speech field to specify your custom speech models. This field is defined as a dictionary, where each key represents a locale code and each value corresponds to the [Model ID](#) of the custom speech model. For more information about custom speech, see [What is custom speech?](#).

Voice live supports using a combination of base models and custom models as long as each type is unique per locale with a maximum of 10 languages specified in total.

Example session configuration with custom speech models. In this example when the detected language is English, the base model is used and, when the detected language is Chinese, the custom speech model is used.

JSON

```
{
  "session": {
    "input_audio_transcription": {
      "model": "azure-speech",
      "language": "en",
      "custom_speech": {
        "zh-CN": "847cb03d-7f22-4b11-444-e1be1d77bf17"
      }
    }
  }
}
```

➊ Note

In order to use a custom speech model with Voice live API, the model must be available on the same Azure AI Foundry resource you're using to call the Voice live API. If you trained the model on a different Azure AI Foundry or Azure AI Speech resource, you have to copy the model to the resource you're using to call the Voice live API. You pay

separately for custom speech training and model hosting. For more information on supported regions, see [Speech service supported regions](#).

Speech output customization

Custom lexicon

Use the `custom_lexicon_url` string property to customize pronunciation for both standard Azure text to speech voices and custom voices. To learn more about how to format the custom lexicon (the same as Speech Synthesis Markup Language (SSML)), see [custom lexicon for text to speech](#).

JSON

```
{  
  "voice": {  
    "name": "en-US-Ava:DragonHDLatestNeural",  
    "type": "azure-standard",  
    "temperature": 0.8, // optional  
    "custom_lexicon_url": "<custom lexicon url>"  
  }  
}
```

Azure custom voices

You can use a custom voice for audio output. For information about how to create a custom voice, see [What is custom voice](#).

JSON

```
{  
  "voice": {  
    "name": "en-US-CustomNeural",  
    "type": "azure-custom",  
    "endpoint_id": "your-endpoint-id", // a guid string  
    "temperature": 0.8 // optional, value range 0.0-1.0, only take effect when using  
    HD voices  
  }  
}
```

Important

Custom voice access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

! [Note](#)

In order to use a custom voice model with Voice live API, the model must be available on the same Azure AI Foundry resource you're using to call the Voice live API. If you trained the model on a different Azure AI Foundry or Azure AI Speech resource, you have to copy it to the resource you're using to call the Voice live API. You pay separately for custom voice training and model hosting. For more information on supported regions, see [Speech service supported regions](#).

Azure custom avatar

[Text to speech avatar](#) converts text into a digital video of a photorealistic human (either a standard avatar or a [custom text to speech avatar](#)) speaking with a natural-sounding voice.

The configuration for a custom avatar doesn't differ from the configuration of a standard avatar. Refer to [How to use the Voice live API - Azure text to speech avatar](#) for a detailed example.

! [Important](#)

Custom text to speech avatar access is [limited](#) based on eligibility and usage criteria. Request access on the [intake form](#).

! [Note](#)

In order to use a custom avatar with Voice live API, the avatar must be available on the same Azure AI Foundry resource you're using to call the Voice live API. If you trained the avatar on a different Azure AI Foundry or Azure AI Speech resource, you have to copy the model to the resource you're using to call the Voice live API. You pay separately for custom avatar training and model hosting. For more information on supported regions, see [Speech service supported regions](#).

! [Note](#)

Custom photo avatar (PREVIEW) training isn't yet available as a self-service option and currently requires a manual offline process.

Related content

- Try out the [Voice live API quickstart](#)
- Learn more about [How to use the Voice live API](#)

Last updated on 11/06/2025

Use the Call Center Voice Agent Accelerator

The Call Center Voice Agent Accelerator is a solution template designed to help developers build real-time speech-to-speech voice agents that deliver personalized self-service experiences for call centers. It lets you develop conversational interactions that feel fast and natural, and it integrates seamlessly with telephony systems, making it ideal for modern contact centers looking to enhance customer interactions.

This accelerator combines the Azure Voice live API and Azure Communication Services (ACS) to enable developers to start locally and deploy to an Azure Web App when ready. It doesn't require a PSTN number to get started, which further simplifies the initial setup process.

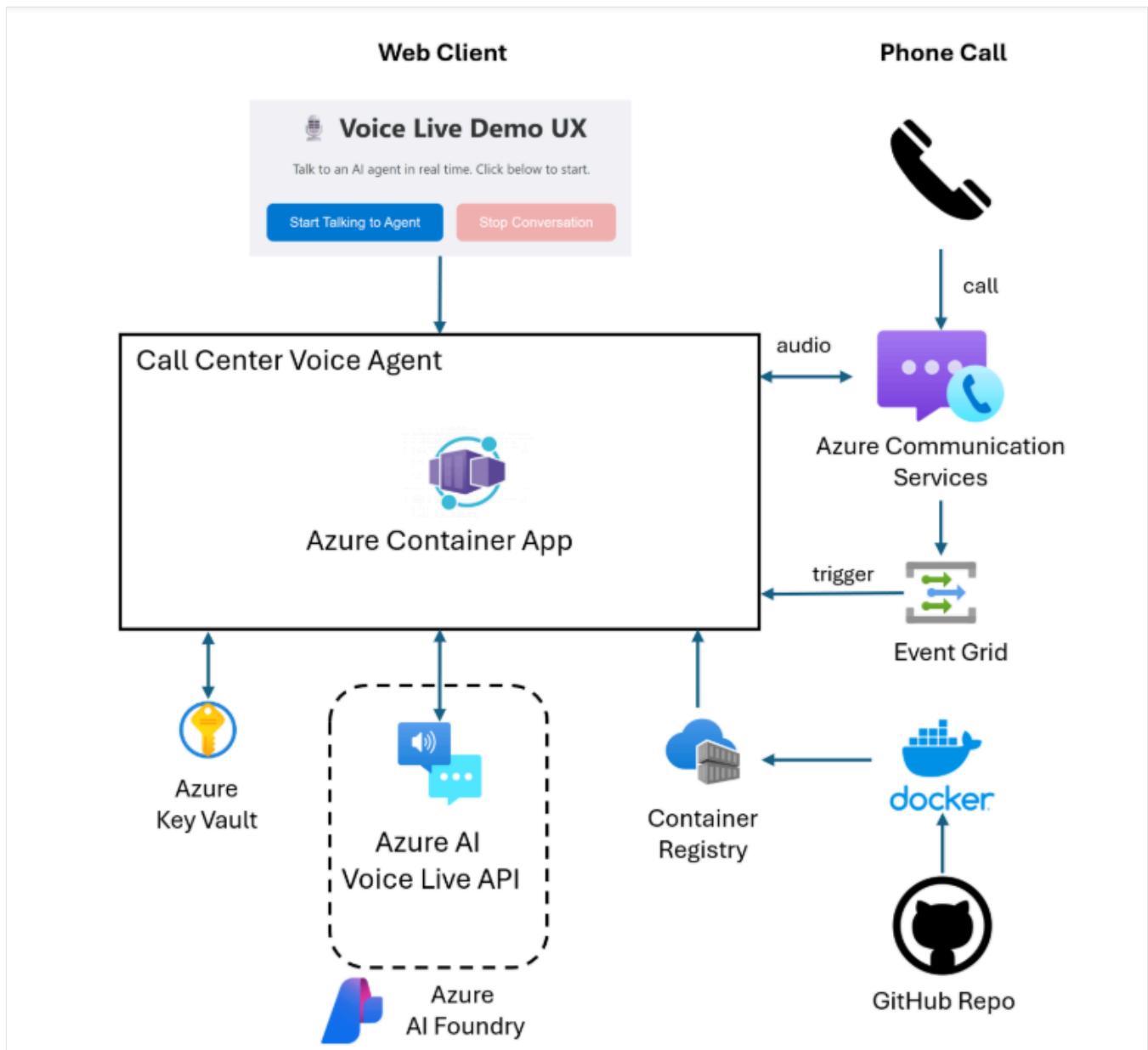
You can find the solution template on GitHub: [Call Center Voice Agent Accelerator with Azure Voice Live API ↗](#).

Solution overview

This solution provides an end-to-end framework for creating scalable, efficient, and low-latency call center voice agent experiences using the Azure Voice Live API and Azure Communication Services APIs.

The Azure Voice Live API provides a single, unified interface that integrates speech recognition, generative AI, and text-to-speech functionalities.

Additionally, the Azure Communication Services Call Automation APIs provide the telephony integration. You can use either an [ACS provided number](#) or direct routing using Session Initiation Protocol (SIP) with your existing PSTN carrier or third-party PBX (see [Use direct routing to connect existing telephony service](#)).



Related content

- Learn more about [Voice live API](#).
- Learn more about [Azure Communication Services](#).

Last updated on 11/04/2025

Voice live FAQ

This article answers commonly asked questions about the Voice live API. If you can't find answers to your questions here, check out [other support options](#).

General

What scenarios does Voice live support?

Voice live API supports a wide range of real-time, natural voice interaction scenarios: contact centers, automotive assistants, accessibility applications, virtual tutors and learning companions, multilingual public service agents, HR support, and training. Used by customers like eClinicalWorks and the Government of Malta.

How does Voice live compare to AOAI Realtime API? When should I choose which?

Voice live API enhances AOAI Realtime API by offering: expanded model selection (including GPT-Realtime, GPT-5, GPT-4.1, PHI), more natural voice options, more supported speech languages, avatar integration, advanced semantic voice activity detection (VAD), seamless Azure AI Foundry Agent Service integration, telephony integration via Azure Communication Services.

What regions does Voice live support?

Voice live is available in 10+ Azure regions. For more information, see [Region support](#).

What is the tokens-per-minute threshold?

The current limit is 100,000 tokens per minute per resource. Customers can request an increase. For more information, see [Speech service quotas and limits](#).

Generative AI Models

What generative AI models are supported?

Supports OpenAI models in Azure AI Foundry, Phi-based LLMs, and SLMs. For more information, see [Voice live overview](#). Voice live also provides an option to bring-your-own model (PREVIEW).

How do I choose the LLM model for my use case?

Consider: accuracy (Azure Speech-based models are more robust for noisy audio), existing LLM solutions (reuse prompts and grounding data), latency (text-based LLMs can have slightly higher latency), inference cost (smaller models can be more cost-effective).

What is response instruction?

Guides model behavior and context. Define agent personality, specify questions, control response formatting. Responses should be concise and normalized for optimal audio synthesis.

What is response temperature?

Controls randomness of output. Lower values = deterministic, higher = creative. Adjust temperature or Top-P, not both.

Speech Input

What languages does Voice live support?

Supports 146 languages/locales for input, 151 for output, 600+ neural voices. See [Voice live language support](#).

How do I get the live transcripts from the call?

Use text output events. Details at [Voice live API reference](#).

What is a phrase list?

Domain-specific terms to improve recognition. Limit to <500 words/phrases. See [How to customize Voice live](#).

Are there other ways to improve speech input recognition accuracy?

Use Azure AI Custom Speech models. Configure multiple custom models per language. See [How to customize Voice live](#).

Speech Output

What voices does Voice live support?

Native audio output with preferred model, Azure AI Speech TTS voices (600+ voices, 150+ locales, 30+ Neural HD voices). Custom voice models via Professional Voice Fine-tuning. For more information, see [Voice live API supported languages](#).

How do I pick a voice?

Use [Voice Gallery](#) in Azure AI Foundry Speech Playground. Consider gender, age, capability, style, personality.

What is voice temperature?

Controls expressiveness. Higher = dynamic/emotive, lower = neutral. Applies to Neural HD voices.

What is speaking rate?

Controls agent's speech speed.

What is a custom lexicon?

Define pronunciation rules for specific words. See [How to customize Voice live](#).

What is Custom Voice?

Create brand-specific synthetic voices using your own audio data. See [How to customize Voice live](#).

What is Avatar support?

Pair speech output with visual avatars for multimodal experiences.

What is Custom Avatar?

Photorealistic digital human using Azure AI TTS. Built from video recordings, tailored to specific actor's appearance and voice.

Conversational Enhancements

What is the difference between Azure Semantic VAD and Basic Server VAD?

Azure Semantic VAD is more noise robust and accurate for detecting utterance boundaries.

What is EOU (End of Utterance) detection?

Uses context to determine if a user finished speaking or just paused.

How does noise suppression work?

Filters background noise based on advanced technology.

How does echo cancellation work?

Removes echo of agent's own voice picked up by microphone.

Function Calling

Does Voice live support function calling?

Yes, including asynchronous function calling.

Is there model context protocol (MCP) support?

Currently MCP isn't supported.

Pricing

Where is the pricing listed?

[Voice live overview](#)

How do I estimate the cost based on my use case?

Estimate by audio minutes; tokens are billing unit. See [pricing](#) and [token usage and cost estimation](#).

Are there separate quota and throttling limits for voice-live?

Yes, quota applies specifically to Voice live API (default: 100k tokens/min).

Additional

Does this service provide an SDK?

Yes, SDKs for Python and C#. See [Voice live - Reference - Voice live SDK](#).

Does this service include content filtering?

Yes, content filtering is included.

Can you modify or disable the content filtering in Voice live API?

No. If you need custom content filtering, you can use the bring-your-own-model (PREVIEW) feature.

Does Voice live API support WebRTC?

WebRTC is currently not supported.

Is SIP supported?

SIP is currently not supported.

Next steps

- Learn more about [How to use the Voice live API](#)
- See the [Voice live API reference](#)
- [What's new](#)

Voice live API Reference

09/27/2025

The Voice live API provides real-time, bidirectional communication for voice-enabled applications using WebSocket connections. This API supports advanced features including speech recognition, text-to-speech synthesis, avatar streaming, animation data, and comprehensive audio processing capabilities.

The API uses JSON-formatted events sent over WebSocket connections to manage conversations, audio streams, avatar interactions, and real-time responses. Events are categorized into client events (sent from client to server) and server events (sent from server to client).

Key Features

- **Real-time Audio Processing:** Support for multiple audio formats including PCM16 at various sample rates and G.711 codecs
- **Advanced Voice Options:** OpenAI voices, Azure custom voices, Azure standard voices, and Azure personal voices
- **Avatar Integration:** WebRTC-based avatar streaming with video, animation, and blendshapes
- **Intelligent Turn Detection:** Multiple VAD options including Azure semantic VAD and server-side detection
- **Audio Enhancement:** Built-in noise reduction and echo cancellation
- **Function Calling:** Tool integration for enhanced conversational capabilities
- **Flexible Session Management:** Configurable modalities, instructions, and response parameters

Client Events

The Voice live API supports the following client events that can be sent from the client to the server:

[+] Expand table

Event	Description
session.update	Update the session configuration including voice, modalities, turn detection, and other settings
session.avatar.connect	Establish avatar connection by providing client SDP for WebRTC negotiation
input_audio_buffer.append	Append audio bytes to the input audio buffer

Event	Description
input_audio_buffer.commit	Commit the input audio buffer for processing
input_audio_buffer.clear	Clear the input audio buffer
conversation.item.create	Add a new item to the conversation context
conversation.item.retrieve	Retrieve a specific item from the conversation
conversation.item.truncate	Truncate an assistant audio message
conversation.item.delete	Remove an item from the conversation
response.create	Instruct the server to create a response via model inference
response.cancel	Cancel an in-progress response

session.update

Update the session's configuration. This event can be sent at any time to modify settings such as voice, modalities, turn detection, tools, and other session parameters. Note that once a session is initialized with a particular model, it can't be changed to another model.

Event Structure

JSON

```
{
  "type": "session.update",
  "session": {
    "modalities": ["text", "audio"],
    "voice": {
      "type": "openai",
      "name": "alloy"
    },
    "instructions": "You are a helpful assistant. Be concise and friendly.",
    "input_audio_format": "pcm16",
    "output_audio_format": "pcm16",
    "input_audio_sampling_rate": 24000,
    "turn_detection": {
      "type": "azure_semantic_vad",
      "threshold": 0.5,
      "prefix_padding_ms": 300,
      "silence_duration_ms": 500
    },
    "temperature": 0.8,
    "max_response_output_tokens": "inf"
  }
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "session.update"
session	RealtimeRequestSession	Session configuration object with fields to update

Example with Azure Custom Voice

JSON

```
{
  "type": "session.update",
  "session": {
    "voice": {
      "type": "azure-custom",
      "name": "my-custom-voice",
      "endpoint_id": "12345678-1234-1234-1234-123456789012",
      "temperature": 0.7,
      "style": "cheerful"
    },
    "input_audio_noise_reduction": {
      "type": "azure_deep_noise_suppression"
    },
    "avatar": {
      "character": "lisa",
      "customized": false,
      "video": {
        "resolution": {
          "width": 1920,
          "height": 1080
        },
        "bitrate": 2000000
      }
    }
  }
}
```

session.avatar.connect

Establish an avatar connection by providing the client's SDP (Session Description Protocol) offer for WebRTC media negotiation. This event is required when using avatar features.

Event Structure

JSON

```
{  
  "type": "session.avatar.connect",  
  "client_sdp": "<client_sdp>"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	Must be "session.avatar.connect"
client_sdp	string	The client's SDP offer for WebRTC connection establishment

input_audio_buffer.append

Append audio bytes to the input audio buffer.

Event Structure

JSON

```
{  
  "type": "input_audio_buffer.append",  
  "audio": "UklGRiQAAABXQVZFm10IBAAAAAABAEARKwAAIhYAQACABAAGF0YQAAAAA="}  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	Must be "input_audio_buffer.append"
audio	string	Base64-encoded audio data

input_audio_buffer.commit

Commit the input audio buffer for processing.

Event Structure

JSON

```
{  
  "type": "input_audio_buffer.commit"  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "input_audio_buffer.commit"

input_audio_buffer.clear

Clear the input audio buffer.

Event Structure

JSON

```
{  
  "type": "input_audio_buffer.clear"  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "input_audio_buffer.clear"

conversation.item.create

Add a new item to the conversation context. This can include messages, function calls, and function call responses. Items can be inserted at specific positions in the conversation history.

Event Structure

JSON

```
{  
  "type": "conversation.item.create",  
  "previous_item_id": "item_ABC123",  
  "item": {  
    "id": "item_DEF456",  
    "type": "message",  
    "role": "user",  
    "content": [  
      {  
        "type": "input_text",  
        "text": "Hello, how are you?"  
      }  
    ]  
  }  
}
```

Properties

[] [Expand table](#)

Field	Type	Description
type	string	Must be "conversation.item.create"
previous_item_id	string	Optional. ID of the item after which to insert this item. If not provided, appends to end
item	RealtimeConversationRequestItem	The item to add to the conversation

Example with Audio Content

JSON

```
{  
  "type": "conversation.item.create",  
  "item": {  
    "type": "message",  
    "role": "user",  
    "content": [  
      {  
        "type": "input_audio",  
        "audio": "Uk1GRiQAAABXQVZFZm10IBAAAAABAEARKwAAIhYAQACABAAZGF0YQAAAAA=",  
        "transcript": "Hello there"  
      }  
    ]  
  }  
}
```

```
}
```

Example with Function Call

JSON

```
{
  "type": "conversation.item.create",
  "item": {
    "type": "function_call",
    "name": "get_weather",
    "call_id": "call_123",
    "arguments": "{\"location\": \"San Francisco\", \"unit\": \"celsius\"}"
  }
}
```

conversation.item.retrieve

Retrieve a specific item from the conversation history. This is useful for inspecting processed audio after noise cancellation and VAD.

Event Structure

JSON

```
{
  "type": "conversation.item.retrieve",
  "item_id": "item_ABC123"
}
```

Properties

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	Must be "conversation.item.retrieve"
item_id	string	The ID of the item to retrieve

conversation.item.truncate

Truncate an assistant message's audio content. This is useful for stopping playback at a specific point and synchronizing the server's understanding with the client's state.

Event Structure

JSON

```
{  
  "type": "conversation.item.truncate",  
  "item_id": "item_ABC123",  
  "content_index": 0,  
  "audio_end_ms": 5000  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	Must be "conversation.item.truncate"
item_id	string	The ID of the assistant message item to truncate
content_index	integer	The index of the content part to truncate
audio_end_ms	integer	The duration up to which to truncate the audio, in milliseconds

conversation.item.delete

Remove an item from the conversation history.

Event Structure

JSON

```
{  
  "type": "conversation.item.delete",  
  "item_id": "item_ABC123"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "conversation.item.delete"
item_id	string	The ID of the item to delete

response.create

Instruct the server to create a response via model inference. This event can specify response-specific configuration that overrides session defaults.

Event Structure

JSON

```
{
  "type": "response.create",
  "response": {
    "modalities": ["text", "audio"],
    "instructions": "Be extra helpful and detailed.",
    "voice": {
      "type": "openai",
      "name": "alloy"
    },
    "output_audio_format": "pcm16",
    "temperature": 0.7,
    "max_response_output_tokens": 1000
  }
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.create"
response	RealtimeResponseOptions	Optional response configuration that overrides session defaults

Example with Tool Choice

JSON

```
{  
  "type": "response.create",  
  "response": {  
    "modalities": ["text"],  
    "tools": [  
      {  
        "type": "function",  
        "name": "get_current_time",  
        "description": "Get the current time",  
        "parameters": {  
          "type": "object",  
          "properties": {}  
        }  
      }  
    ],  
    "tool_choice": "get_current_time",  
    "temperature": 0.3  
  }  
}
```

Example with Animation

JSON

```
{  
  "type": "response.create",  
  "response": {  
    "modalities": ["audio", "animation"],  
    "animation": {  
      "model_name": "default",  
      "outputs": ["blendshapes", "viseme_id"]  
    },  
    "voice": {  
      "type": "azure-custom",  
      "name": "my-expressive-voice",  
      "endpoint_id": "12345678-1234-1234-1234-123456789012",  
      "style": "excited"  
    }  
  }  
}
```

response.cancel

Cancel an in-progress response. This immediately stops response generation and related audio output.

Event Structure

JSON

```
{  
  "type": "response.cancel"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	Must be "response.cancel"

Properties

[+] Expand table

Field	Type	Description
type	string	The event type must be conversation.item.retrieve.
item_id	string	The ID of the item to retrieve.
event_id	string	The ID of the event.

RealtimeClientEventConversationItemTruncate

The client `conversation.item.truncate` event is used to truncate a previous assistant message's audio. The server produces audio faster than realtime, so this event is useful when the user interrupts to truncate audio that was sent to the client but not yet played. The server's understanding of the audio with the client's playback is synchronized.

Truncating audio deletes the server-side text transcript to ensure there isn't text in the context that the user doesn't know about.

If the client event is successful, the server responds with a `conversation.item.truncated` event.

Event structure

JSON

```
{  
  "type": "conversation.item.truncate",
```

```
"item_id": "<item_id>",  
"content_index": 0,  
"audio_end_ms": 0  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>conversation.item.truncate</code> .
item_id	string	The ID of the assistant message item to truncate. Only assistant message items can be truncated.
content_index	integer	The index of the content part to truncate. Set this property to "0".
audio_end_ms	integer	Inclusive duration up to which audio is truncated, in milliseconds. If the <code>audio_end_ms</code> is greater than the actual audio duration, the server responds with an error.

RealtimeClientEventInputAudioBufferAppend

The client `input_audio_buffer.append` event is used to append audio bytes to the input audio buffer. The audio buffer is temporary storage you can write to and later commit.

In Server VAD (Voice Activity Detection) mode, the audio buffer is used to detect speech and the server decides when to commit. When server VAD is disabled, the client can choose how much audio to place in each event up to a maximum of 15 MiB. For example, streaming smaller chunks from the client can allow the VAD to be more responsive.

Unlike most other client events, the server doesn't send a confirmation response to client `input_audio_buffer.append` event.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.append",  
  "audio": "<audio>"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.append</code> .
audio	string	Base64-encoded audio bytes. This value must be in the format specified by the <code>input_audio_format</code> field in the session configuration.

RealtimeClientEventInputAudioBufferClear

The client `input_audio_buffer.clear` event is used to clear the audio bytes in the buffer.

The server responds with an `input_audio_buffer.cleared` event.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.clear"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.clear</code> .

RealtimeClientEventInputAudioBufferCommit

The client `input_audio_buffer.commit` event is used to commit the user input audio buffer, which creates a new user message item in the conversation. Audio is transcribed if `input_audio_transcription` is configured for the session.

When in server VAD mode, the client doesn't need to send this event, the server commits the audio buffer automatically. Without server VAD, the client must commit the audio buffer to create a user message item. This client event produces an error if the input audio buffer is empty.

Committing the input audio buffer doesn't create a response from the model.

The server responds with an `input_audio_buffer.committed` event.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.commit"  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.commit</code> .

RealtimeClientEventResponseCancel

The client `response.cancel` event is used to cancel an in-progress response.

The server will respond with a `response.done` event with a status of `response.status=cancelled`.

Event structure

JSON

```
{  
  "type": "response.cancel"  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.cancel</code> .

RealtimeClientEventResponseCreate

The client `response.create` event is used to instruct the server to create a response via model inference. When the session is configured in server VAD mode, the server creates responses automatically.

A response includes at least one `item`, and can have two, in which case the second is a function call. These items are appended to the conversation history.

The server responds with a `response.created` event, one or more item and content events (such as `conversation.item.created` and `response.content_part.added`), and finally a `response.done` event to indicate the response is complete.

Event structure

JSON

```
{  
  "type": "response.create"  
}
```

Properties

[+] [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.create</code> .
response	RealtimeResponseOptions	The response options.

RealtimeClientEventSessionUpdate

The client `session.update` event is used to update the session's default configuration. The client can send this event at any time to update the session configuration, and any field can be updated at any time, except for voice.

Only fields that are present are updated. To clear a field (such as `instructions`), pass an empty string.

The server responds with a `session.updated` event that contains the full effective configuration.

Event structure

JSON

```
{  
  "type": "session.update"  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>session.update</code> .
session	RealtimeRequestSession	The session configuration.

Server Events

The Voice live API sends the following server events to communicate status, responses, and data to the client:

 [Expand table](#)

Event	Description
error	Indicates an error occurred during processing
session.created	Sent when a new session is successfully established
session.updated	Sent when session configuration is updated
session.avatar.connecting	Indicates avatar WebRTC connection is being established
conversation.item.created	Sent when a new item is added to the conversation
conversation.item.retrieved	Response to <code>conversation.item.retrieve</code> request
conversation.item.truncated	Confirms item truncation
conversation.item.deleted	Confirms item deletion
conversation.item.input_audio_transcription.completed	Input audio transcription is complete
conversation.item.input_audio_transcription.delta	Streaming input audio transcription
conversation.item.input_audio_transcription.failed	Input audio transcription failed

Event	Description
input_audio_buffer.committed	Input audio buffer has been committed for processing
input_audio_buffer.cleared	Input audio buffer has been cleared
input_audio_buffer.speech_started	Speech detected in input audio buffer (VAD)
input_audio_buffer.speech_stopped	Speech ended in input audio buffer (VAD)
response.created	New response generation has started
response.done	Response generation is complete
response.output_item.added	New output item added to response
response.output_item.done	Output item is complete
response.content_part.added	New content part added to output item
response.content_part.done	Content part is complete
response.text.delta	Streaming text content from the model
response.text.done	Text content is complete
response.audio_transcript.delta	Streaming audio transcript
response.audio_transcript.done	Audio transcript is complete
response.audio.delta	Streaming audio content from the model
response.audio.done	Audio content is complete
response.animation_blendshapes.delta	Streaming animation blendshapes data
response.animation_blendshapes.done	Animation blendshapes data is complete
response.audio_timestamp.delta	Streaming audio timestamp information
response.audio_timestamp.done	Audio timestamp information is complete
response.animation_viseme.delta	Streaming animation viseme data
response.animation_viseme.done	Animation viseme data is complete
response.function_call_arguments.delta	Streaming function call arguments
response.function_call_arguments.done	Function call arguments are complete

session.created

Sent when a new session is successfully established. This is the first event received after connecting to the API.

Event Structure

JSON

```
{  
  "type": "session.created",  
  "session": {  
    "id": "sess_ABC123DEF456",  
    "object": "realtime.session",  
    "model": "gpt-4o-realtime-preview",  
    "modalities": ["text", "audio"],  
    "instructions": "You are a helpful assistant.",  
    "voice": {  
      "type": "openai",  
      "name": "alloy"  
    },  
    "input_audio_format": "pcm16",  
    "output_audio_format": "pcm16",  
    "input_audio_sampling_rate": 24000,  
    "turn_detection": {  
      "type": "azure_semantic_vad",  
      "threshold": 0.5,  
      "prefix_padding_ms": 300,  
      "silence_duration_ms": 500  
    },  
    "temperature": 0.8,  
    "max_response_output_tokens": "inf"  
  }  
}
```

Properties

[] Expand table

Field	Type	Description
type	string	Must be "session.created"
session	RealtimeResponseSession	The created session object

session.updated

Sent when session configuration is successfully updated in response to a `session.update` client event.

Event Structure

JSON

```
{  
  "type": "session.updated",  
  "session": {  
    "id": "sess_ABC123DEF456",  
    "voice": {  
      "type": "azure-custom",  
      "name": "my-voice",  
      "endpoint_id": "12345678-1234-1234-1234-123456789012"  
    },  
    "temperature": 0.7,  
    "avatar": {  
      "character": "lisa",  
      "customized": false  
    }  
  }  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	Must be "session.updated"
session	RealtimeResponseSession	The updated session object

session.avatar.connecting

Indicates that an avatar WebRTC connection is being established. This event is sent in response to a `session.avatar.connect` client event.

Event Structure

JSON

```
{  
  "type": "session.avatar.connecting",  
  "server_sdp": "<server_sdp>"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be <code>"session.avatar.connecting"</code>

conversation.item.created

Sent when a new item is added to the conversation, either through a client `conversation.item.create` event or automatically during response generation.

Event Structure

JSON

{
"type": "conversation.item.created",
"previous_item_id": "item_ABC123",
"item": {
"id": "item_DEF456",
"object": "realtime.item",
"type": "message",
"status": "completed",
"role": "user",
"content": [
{
"type": "input_text",
"text": "Hello, how are you?"
}
]
}
}

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be <code>"conversation.item.created"</code>
previous_item_id	string	ID of the item after which this item was inserted
item	RealtimeConversationResponseItem	The created conversation item

Example with Audio Item

JSON

```
{  
  "type": "conversation.item.created",  
  "item": {  
    "id": "item_GHI789",  
    "type": "message",  
    "status": "completed",  
    "role": "user",  
    "content": [  
      {  
        "type": "input_audio",  
        "audio": null,  
        "transcript": "What's the weather like today?"  
      }  
    ]  
  }  
}
```

conversation.item.retrieved

Sent in response to a `conversation.item.retrieve` client event, providing the requested conversation item.

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be <code>"conversation.item.created"</code>
item	RealtimeConversationResponseItem	The created conversation item

conversation.item.truncated

The server `conversation.item.truncated` event is returned when the client truncates an earlier assistant audio message item with a `conversation.item.truncate` event. This event is used to synchronize the server's understanding of the audio with the client's playback.

This event truncates the audio and removes the server-side text transcript to ensure there's no text in the context that the user doesn't know about.

Event structure

JSON

```
{  
  "type": "conversation.item.truncated",  
  "item_id": "<item_id>",  
  "content_index": 0,  
  "audio_end_ms": 0  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>conversation.item.truncated</code> .
item_id	string	The ID of the assistant message item that was truncated.
content_index	integer	The index of the content part that was truncated.
audio_end_ms	integer	The duration up to which the audio was truncated, in milliseconds.

conversation.item.deleted

Sent in response to a `conversation.item.delete` client event, confirming that the specified item has been removed from the conversation.

Event Structure

JSON

```
{  
  "type": "conversation.item.deleted",  
  "item_id": "item_ABC123"  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be <code>"conversation.item.deleted"</code>

Field	Type	Description
item_id	string	ID of the deleted item

response.created

Sent when a new response generation begins. This is the first event in a response sequence.

Event Structure

JSON

```
{  
  "type": "response.created",  
  "response": {  
    "id": "resp_ABC123",  
    "object": "realtime.response",  
    "status": "in_progress",  
    "status_details": null,  
    "output": [],  
    "usage": {  
      "total_tokens": 0,  
      "input_tokens": 0,  
      "output_tokens": 0  
    }  
  }  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.created"
response	RealtimeResponse	The response object that was created

response.done

Sent when response generation is complete. This event contains the final response with all output items and usage statistics.

Event Structure

JSON

```
{
  "type": "response.done",
  "response": {
    "id": "resp_ABC123",
    "object": "realtime.response",
    "status": "completed",
    "status_details": null,
    "output": [
      {
        "id": "item_DEF456",
        "object": "realtime.item",
        "type": "message",
        "status": "completed",
        "role": "assistant",
        "content": [
          {
            "type": "text",
            "text": "Hello! I'm doing well, thank you for asking. How can I help you today?"
          }
        ]
      }
    ],
    "usage": {
      "total_tokens": 87,
      "input_tokens": 52,
      "output_tokens": 35,
      "input_token_details": {
        "cached_tokens": 0,
        "text_tokens": 45,
        "audio_tokens": 7
      },
      "output_token_details": {
        "text_tokens": 15,
        "audio_tokens": 20
      }
    }
  }
}
```

Properties

[] Expand table

Field	Type	Description
type	string	Must be "response.done"
response	RealtimeResponse	The completed response object

response.output_item.added

Sent when a new output item is added to the response during generation.

Event Structure

JSON

```
{  
  "type": "response.output_item.added",  
  "response_id": "resp_ABC123",  
  "output_index": 0,  
  "item": {  
    "id": "item_DEF456",  
    "object": "realtime.item",  
    "type": "message",  
    "status": "in_progress",  
    "role": "assistant",  
    "content": []  
  }  
}
```

Properties

[] [Expand table](#)

Field	Type	Description
type	string	Must be "response.output_item.added"
response_id	string	ID of the response this item belongs to
output_index	integer	Index of the item in the response's output array
item	RealtimeConversationResponseItem	The output item that was added

response.output_item.done

Sent when an output item is complete.

Event Structure

JSON

```
{  
  "type": "response.output_item.done",
```

```

"response_id": "resp_ABC123",
"output_index": 0,
"item": {
  "id": "item_DEF456",
  "object": "realtime.item",
  "type": "message",
  "status": "completed",
  "role": "assistant",
  "content": [
    {
      "type": "text",
      "text": "Hello! I'm doing well, thank you for asking."
    }
  ]
}
}

```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be <code>"response.output_item.done"</code>
response_id	string	ID of the response this item belongs to
output_index	integer	Index of the item in the response's output array
item	RealtimeConversationResponseItem	The completed output item

response.content_part.added

The server `response.content_part.added` event is returned when a new content part is added to an assistant message item during response generation.

Event Structure

JSON

```
{
  "type": "response.content_part.added",
  "response_id": "resp_ABC123",
  "item_id": "item_DEF456",
  "output_index": 0,
  "content_index": 0,
  "part": {
    "type": "text",
    "text": ""
  }
}
```

```
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "response.content_part.added"
response_id	string	ID of the response
item_id	string	ID of the item this content part belongs to
output_index	integer	Index of the item in the response
content_index	integer	Index of this content part in the item
part	RealtimeContentPart	The content part that was added

response.content_part.done

The server `response.content_part.done` event is returned when a content part is done streaming in an assistant message item.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event Structure

JSON

```
{
  "type": "response.content_part.done",
  "response_id": "resp_ABC123",
  "item_id": "item_DEF456",
  "output_index": 0,
  "content_index": 0,
  "part": {
    "type": "text",
    "text": "Hello! I'm doing well, thank you for asking."
  }
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.content_part.done"
response_id	string	ID of the response
item_id	string	ID of the item this content part belongs to
output_index	integer	Index of the item in the response
content_index	integer	Index of this content part in the item
part	RealtimeContentPart	The completed content part

response.text.delta

Streaming text content from the model. Sent incrementally as the model generates text.

Event Structure

JSON

```
{  
  "type": "response.text.delta",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0,  
  "delta": "Hello! I'm"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.text.delta"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part

Field	Type	Description
delta	string	Incremental text content

response.text.done

Sent when text content generation is complete.

Event Structure

JSON
<pre>{ "type": "response.text.done", "response_id": "resp_ABC123", "item_id": "item_DEF456", "output_index": 0, "content_index": 0, "text": "Hello! I'm doing well, thank you for asking. How can I help you today?" }</pre>

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "response.text.done"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
text	string	The complete text content

response.audio.delta

Streaming audio content from the model. Audio is provided as base64-encoded data.

Event Structure

JSON

```
{  
  "type": "response.audio.delta",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0,  
  "delta": "UklGRiQAAABXQVZFZm10IBAAAAAABAAEARKwAAIhYAQACABAAZGF0YQAAAAA="}  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "response.audio.delta"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
delta	string	Base64-encoded audio data chunk

response.audio.done

Sent when audio content generation is complete.

Event Structure

JSON

```
{  
  "type": "response.audio.done",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.audio.done"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part

response.audio_transcript.delta

Streaming transcript of the generated audio content.

Event Structure

JSON

```
{  
  "type": "response.audio_transcript.delta",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0,  
  "delta": "Hello! I'm doing"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	Must be "response.audio_transcript.delta"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
delta	string	Incremental transcript text

response.audio_transcript.done

Sent when audio transcript generation is complete.

Event Structure

JSON

```
{  
  "type": "response.audio_transcript.done",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0,  
  "transcript": "Hello! I'm doing well, thank you for asking. How can I help you  
today?"  
}
```

Properties

[] [Expand table](#)

Field	Type	Description
type	string	Must be "response.audio_transcript.done"
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
transcript	string	The complete transcript text

conversation.item.input_audio_transcription.completed

The server `conversation.item.input_audio_transcription.completed` event is the result of audio transcription for speech written to the audio buffer.

Transcription begins when the input audio buffer is committed by the client or server (in `server_vad` mode). Transcription runs asynchronously with response creation, so this event can come before or after the response events.

Realtime API models accept audio natively, and thus input transcription is a separate process run on a separate speech recognition model such as `whisper-1`. Thus the transcript can diverge somewhat from the model's interpretation, and should be treated as a rough guide.

Event structure

JSON

```
{  
  "type": "conversation.item.input_audio_transcription.completed",  
  "item_id": "<item_id>",  
  "content_index": 0,  
  "transcript": "<transcript>"  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>conversation.item.input_audio_transcription.completed</code> .
item_id	string	The ID of the user message item containing the audio.
content_index	integer	The index of the content part containing the audio.
transcript	string	The transcribed text.

conversation.item.input_audio_transcription.delta

The server `conversation.item.input_audio_transcription.delta` event is returned when input audio transcription is configured, and a transcription request for a user message is in progress. This event provides partial transcription results as they become available.

Event structure

JSON

```
{  
  "type": "conversation.item.input_audio_transcription.delta",  
  "item_id": "<item_id>",  
  "content_index": 0,  
  "delta": "<delta>"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>conversation.item.input_audio_transcription.delta</code> .
item_id	string	The ID of the user message item.
content_index	integer	The index of the content part containing the audio.
delta	string	The incremental transcription text.

conversation.item.input_audio_transcription.failed

The server `conversation.item.input_audio_transcription.failed` event is returned when input audio transcription is configured, and a transcription request for a user message failed. This event is separate from other `error` events so that the client can identify the related item.

Event structure

JSON

```
{  
  "type": "conversation.item.input_audio_transcription.failed",  
  "item_id": "<item_id>",  
  "content_index": 0,  
  "error": {  
    "code": "<code>",  
    "message": "<message>",  
    "param": "<param>"  
  }  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>conversation.item.input_audio_transcription.failed</code> .
item_id	string	The ID of the user message item.
content_index	integer	The index of the content part containing the audio.

Field	Type	Description
error	object	Details of the transcription error. See nested properties in the next table.

Error properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The type of error.
code	string	Error code, if any.
message	string	A human-readable error message.
param	string	Parameter related to the error, if any.

response.animation_blendshapes.delta

The server `response.animation_blendshapes.delta` event is returned when the model generates animation blendshapes data as part of a response. This event provides incremental blendshapes data as it becomes available.

Event structure

JSON
<pre>{ "type": "response.animation_blendshapes.delta", "response_id": "resp_ABC123", "item_id": "item_DEF456", "output_index": 0, "content_index": 0, "frame_index": 0, "frames": [[0.0, 0.1, 0.2, ..., 1.0] ...] }</pre>

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.animation_blendshapes.delta</code> .
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
frame_index	integer	Index of the first frame in this batch of frames
frames	array of array of float	Array of blendshape frames, each frame is an array of blendshape values

response.animation_blendshapes.done

The server `response.animation_blendshapes.done` event is returned when the model has finished generating animation blendshapes data as part of a response.

Event structure

JSON

```
{  
  "type": "response.animation_blendshapes.done",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.animation_blendshapes.done</code> .
response_id	string	ID of the response
item_id	string	ID of the item

Field	Type	Description
output_index	integer	Index of the item in the response

response.audio_timestamp.delta

The server `response.audio_timestamp.delta` event is returned when the model generates audio timestamp data as part of a response. This event provides incremental timestamp data for output audio and text alignment as it becomes available.

Event structure

JSON
<pre>{ "type": "response.audio_timestamp.delta", "response_id": "resp_ABC123", "item_id": "item_DEF456", "output_index": 0, "content_index": 0, "audio_offset_ms": 0, "audio_duration_ms": 500, "text": "Hello", "timestamp_type": "word" }</pre>

Properties

[\[\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.audio_timestamp.delta</code> .
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
audio_offset_ms	integer	Audio offset in milliseconds from the start of the audio
audio_duration_ms	integer	Duration of the audio segment in milliseconds
text	string	The text segment corresponding to this audio timestamp

Field	Type	Description
timestamp_type	string	The type of timestamp, currently only "word" is supported

response.audio_timestamp.done

Sent when audio timestamp generation is complete.

Event Structure

JSON
<pre>{ "type": "response.audio_timestamp.done", "response_id": "resp_ABC123", "item_id": "item_DEF456", "output_index": 0, "content_index": 0 }</pre>

Properties

[\[\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.audio_timestamp.done</code> .
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part

response.animation_viseme.delta

The server `response.animation_viseme.delta` event is returned when the model generates animation viseme data as part of a response. This event provides incremental viseme data as it becomes available.

Event Structure

JSON

```
{  
  "type": "response.animation_viseme.delta",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,  
  "content_index": 0,  
  "audio_offset_ms": 0,  
  "viseme_id": 1  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.animation_viseme.delta</code> .
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part
audio_offset_ms	integer	Audio offset in milliseconds from the start of the audio
viseme_id	integer	The viseme ID corresponding to the mouth shape for animation

response.animation_viseme.done

The server `response.animation_viseme.done` event is returned when the model has finished generating animation viseme data as part of a response.

Event Structure

JSON

```
{  
  "type": "response.animation_viseme.done",  
  "response_id": "resp_ABC123",  
  "item_id": "item_DEF456",  
  "output_index": 0,
```

```
    "content_index": 0
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.animation_viseme.done</code> .
response_id	string	ID of the response
item_id	string	ID of the item
output_index	integer	Index of the item in the response
content_index	integer	Index of the content part

The server `response.animation_viseme.delta` event is returned when the model generates animation viseme data as part of a response. This event provides incremental viseme data as it becomes available.

error

The server `error` event is returned when an error occurs, which could be a client problem or a server problem. Most errors are recoverable and the session stays open.

Event structure

JSON

```
{
  "type": "error",
  "error": {
    "code": "<code>",
    "message": "<message>",
    "param": "<param>",
    "event_id": "<event_id>"
  }
}
```

Properties

[\[\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>error</code> .
error	object	Details of the error.
See nested properties in the next table.		

Error properties

[\[\] Expand table](#)

Field	Type	Description
type	string	The type of error. For example, "invalid_request_error" and "server_error" are error types.
code	string	Error code, if any.
message	string	A human-readable error message.
param	string	Parameter related to the error, if any.
event_id	string	The ID of the client event that caused the error, if applicable.

input_audio_buffer.cleared

The server `input_audio_buffer.cleared` event is returned when the client clears the input audio buffer with a `input_audio_buffer.clear` event.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.cleared"  
}
```

Properties

[\[\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.cleared</code> .

input_audio_buffer.committed

The server `input_audio_buffer.committed` event is returned when an input audio buffer is committed, either by the client or automatically in server VAD mode. The `item_id` property is the ID of the user message item created. Thus a `conversation.item.created` event is also sent to the client.

Event structure

JSON

```
{
  "type": "input_audio_buffer.committed",
  "previous_item_id": "<previous_item_id>",
  "item_id": "<item_id>"
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.committed</code> .
previous_item_id	string	The ID of the preceding item after which the new item is inserted.
item_id	string	The ID of the user message item created.

input_audio_buffer.speech_started

The server `input_audio_buffer.speech_started` event is returned in `server_vad` mode when speech is detected in the audio buffer. This event can happen any time audio is added to the buffer (unless speech is already detected).

 Note

The client might want to use this event to interrupt audio playback or provide visual feedback to the user.

The client should expect to receive a `input_audio_buffer.speech_stopped` event when speech stops. The `item_id` property is the ID of the user message item created when speech stops. The `item_id` is also included in the `input_audio_buffer.speech_stopped` event unless the client manually commits the audio buffer during VAD activation.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.speech_started",  
  "audio_start_ms": 0,  
  "item_id": "<item_id>"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.speech_started</code> .
audio_start_ms	integer	Milliseconds from the start of all audio written to the buffer during the session when speech was first detected. This property corresponds to the beginning of audio sent to the model, and thus includes the <code>prefix_padding_ms</code> configured in the session.
item_id	string	The ID of the user message item created when speech stops.

input_audio_buffer.speech_stopped

The server `input_audio_buffer.speech_stopped` event is returned in `server_vad` mode when the server detects the end of speech in the audio buffer.

The server also sends a `conversation.item.created` event with the user message item created from the audio buffer.

Event structure

JSON

```
{  
  "type": "input_audio_buffer.speech_stopped",  
  "audio_end_ms": 0,  
  "item_id": "<item_id>"  
}
```

Properties

[\[\]](#) Expand table

Field	Type	Description
type	string	The event type must be <code>input_audio_buffer.speech_stopped</code> .
audio_end_ms	integer	Milliseconds since the session started when speech stopped. This property corresponds to the end of audio sent to the model, and thus includes the <code>min_silence_duration_ms</code> configured in the session.
item_id	string	The ID of the user message item created.

rate_limits.updated

The server `rate_limits.updated` event is emitted at the beginning of a response to indicate the updated rate limits.

When a response is created, some tokens are reserved for the output tokens. The rate limits shown here reflect that reservation, which is then adjusted accordingly once the response is completed.

Event structure

JSON

```
{  
  "type": "rate_limits.updated",  
  "rate_limits": [  
    {  
      "name": "<name>",  
      "limit": 0,  
      "remaining": 0,  
      "reset_seconds": 0  
    }  
  ]  
}
```

Properties

[+] [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>rate_limits.updated</code> .
rate_limits	array of RealtimeRateLimitsItem	The list of rate limit information.

response.audio.delta

The server `response.audio.delta` event is returned when the model-generated audio is updated.

Event structure

JSON

```
{  
  "type": "response.audio.delta",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0,  
  "delta": "<delta>"  
}
```

Properties

[+] [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.audio.delta</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.
delta	string	Base64-encoded audio data delta.

response.audio.done

The server `response.audio.done` event is returned when the model-generated audio is done.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event structure

JSON

```
{  
  "type": "response.audio.done",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0  
}
```

Properties

 [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.audio.done</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.

response.audio_transcript.delta

The server `response.audio_transcript.delta` event is returned when the model-generated transcription of audio output is updated.

Event structure

JSON

```
{  
  "type": "response.audio_transcript.delta",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,
```

```
"content_index": 0,  
"delta": "<delta>"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	The event type must be <code>response.audio_transcript.delta</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.
delta	string	The transcript delta.

response.audio_transcript.done

The server `response.audio_transcript.done` event is returned when the model-generated transcription of audio output is done streaming.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event structure

JSON

```
{  
  "type": "response.audio_transcript.done",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0,  
  "transcript": "<transcript>"  
}
```

Properties

[+] Expand table

Field	Type	Description
type	string	The event type must be <code>response.audio_transcript.done</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.
transcript	string	The final transcript of the audio.

response.function_call_arguments.delta

The server `response.function_call_arguments.delta` event is returned when the model-generated function call arguments are updated.

Event structure

JSON

```
{
  "type": "response.function_call_arguments.delta",
  "response_id": "<response_id>",
  "item_id": "<item_id>",
  "output_index": 0,
  "call_id": "<call_id>",
  "delta": "<delta>"
}
```

Properties

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.function_call_arguments.delta</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the function call item.
output_index	integer	The index of the output item in the response.
call_id	string	The ID of the function call.

Field	Type	Description
delta	string	The arguments delta as a JSON string.

response.function_call_arguments.done

The server `response.function_call_arguments.done` event is returned when the model-generated function call arguments are done streaming.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event structure

JSON
<pre>{ "type": "response.function_call_arguments.done", "response_id": "<response_id>", "item_id": "<item_id>", "output_index": 0, "call_id": "<call_id>", "arguments": "<arguments>" }</pre>

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.function_call_arguments.done</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the function call item.
output_index	integer	The index of the output item in the response.
call_id	string	The ID of the function call.
arguments	string	The final arguments as a JSON string.

response.output_item.added

The server `response.output_item.added` event is returned when a new item is created during response generation.

Event structure

JSON

```
{  
  "type": "response.output_item.added",  
  "response_id": "<response_id>",  
  "output_index": 0  
}
```

Properties

[\[+\] Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.output_item.added</code> .
response_id	string	The ID of the response to which the item belongs.
output_index	integer	The index of the output item in the response.
item	RealtimeConversationResponseItem	The item that was added.

response.output_item.done

The server `response.output_item.done` event is returned when an item is done streaming.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event structure

JSON

```
{  
  "type": "response.output_item.done",  
  "response_id": "<response_id>",  
  "output_index": 0  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.output_item.done</code> .
response_id	string	The ID of the response to which the item belongs.
output_index	integer	The index of the output item in the response.
item	RealtimeConversationResponseItem	The item that is done streaming.

response.text.delta

The server `response.text.delta` event is returned when the model-generated text is updated. The text corresponds to the `text` content part of an assistant message item.

Event structure

JSON

```
{  
  "type": "response.text.delta",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0,  
  "delta": "<delta>"  
}
```

Properties

[Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.text.delta</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.
delta	string	The text delta.

response.text.done

The server `response.text.done` event is returned when the model-generated text is done streaming. The text corresponds to the `text` content part of an assistant message item.

This event is also returned when a response is interrupted, incomplete, or canceled.

Event structure

JSON

```
{  
  "type": "response.text.done",  
  "response_id": "<response_id>",  
  "item_id": "<item_id>",  
  "output_index": 0,  
  "content_index": 0,  
  "text": "<text>"  
}
```

Properties

[] [Expand table](#)

Field	Type	Description
type	string	The event type must be <code>response.text.done</code> .
response_id	string	The ID of the response.
item_id	string	The ID of the item.
output_index	integer	The index of the output item in the response.
content_index	integer	The index of the content part in the item's content array.
text	string	The final text content.

Components

Audio Formats

RealtimeAudioFormat

Base audio format used for input audio.

Allowed Values:

- `pcm16` - 16-bit PCM audio format
- `g711_ulaw` - G.711 μ-law audio format
- `g711_alaw` - G.711 A-law audio format

RealtimeOutputAudioFormat

Audio format used for output audio with specific sampling rates.

Allowed Values:

- `pcm16` - 16-bit PCM audio format at default sampling rate (24kHz)
- `pcm16_8000hz` - 16-bit PCM audio format at 8kHz sampling rate
- `pcm16_16000hz` - 16-bit PCM audio format at 16kHz sampling rate
- `g711_ulaw` - G.711 μ-law (mu-law) audio format at 8kHz sampling rate
- `g711_alaw` - G.711 A-law audio format at 8kHz sampling rate

RealtimeAudioInputTranscriptionSettings

Configuration for input audio transcription.

[\[+\] Expand table](#)

Field	Type	Description
model	string	The transcription model. Supported: <code>whisper-1</code> , <code>gpt-4o-transcribe</code> , <code>gpt-4o-minitranscribe</code> , <code>azure-speech</code>
language	string	Optional language code in BCP-47 (e.g., <code>en-us</code>), or ISO-639-1 (e.g., <code>en</code>), or multi languages with auto detection, (e.g., <code>en,zh</code>).
custom_speech	object	Optional configuration for custom speech models
phrase_list	string[]	Optional list of phrase hints to bias recognition

RealtimeInputAudioNoiseReductionSettings

Configuration for input audio noise reduction.

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "azure_deep_noise_suppression"

RealtimeInputAudioEchoCancellationSettings

Echo cancellation configuration for server-side audio processing.

[\[\] Expand table](#)

Field	Type	Description
type	string	Must be "server_echo_cancellation"

Voice Configuration

RealtimeVoice

Union of all supported voice configurations.

This can be:

- An [RealtimeOpenAVoice](#) object
- An [RealtimeAzureVoice](#) object

RealtimeOpenAVoice

OpenAI voice configuration with explicit type field.

[\[\] Expand table](#)

Field	Type	Description
type	string	Must be "openai"
name	string	OpenAI voice name: <code>alloy</code> , <code>ash</code> , <code>ballad</code> , <code>coral</code> , <code>echo</code> , <code>sage</code> , <code>shimmer</code> , <code>verse</code>

RealtimeAzureVoice

Base for Azure voice configurations. This is a discriminated union with different types:

RealtimeAzureCustomVoice

Azure custom voice configuration (preferred for custom voices).

[+] Expand table

Field	Type	Description
type	string	Must be "azure-custom"
name	string	Voice name (cannot be empty)
endpoint_id	string	Endpoint ID (cannot be empty)
temperature	number	Optional. Temperature between 0.0 and 1.0
custom_lexicon_url	string	Optional. URL to custom lexicon
prefer_locales	string[]	Optional. Preferred locales
locale	string	Optional. Locale specification
style	string	Optional. Voice style
pitch	string	Optional. Pitch adjustment
rate	string	Optional. Speech rate adjustment
volume	string	Optional. Volume adjustment

Example:

JSON

```
{  
  "type": "azure-custom",  
  "name": "my-custom-voice",  
  "endpoint_id": "12345678-1234-1234-1234-123456789012",  
  "temperature": 0.7,  
  "style": "cheerful",  
  "locale": "en-US"  
}
```

RealtimeAzureStandardVoice

Azure standard voice configuration.

[+] Expand table

Field	Type	Description
type	string	Must be "azure-standard"

Field	Type	Description
name	string	Voice name (cannot be empty)
temperature	number	Optional. Temperature between 0.0 and 1.0
custom_lexicon_url	string	Optional. URL to custom lexicon
prefer_locales	string[]	Optional. Preferred locales
locale	string	Optional. Locale specification
style	string	Optional. Voice style
pitch	string	Optional. Pitch adjustment
rate	string	Optional. Speech rate adjustment
volume	string	Optional. Volume adjustment

RealtimeAzurePersonalVoice

Azure personal voice configuration.

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "azure-personal"
name	string	Voice name (cannot be empty)
temperature	number	Optional. Temperature between 0.0 and 1.0
model	string	Underlying neural model: DragonLatestNeural, PhoenixLatestNeural, PhoenixV2Neural

Turn Detection

RealtimeTurnDetection

Configuration for turn detection. This is a discriminated union supporting multiple VAD types.

RealtimeServerVad

Base VAD-based turn detection.

[Expand table](#)

Field	Type	Description
type	string	Must be "server_vad"
threshold	number	Optional. Activation threshold (0.0-1.0)
prefix_padding_ms	integer	Optional. Audio padding before speech starts
silence_duration_ms	integer	Optional. Silence duration to detect speech end
end_of_utterance_detection	RealtimeEOUDetection	Optional. End-of-utterance detection config
auto_truncate	boolean	Optional. Auto-truncate on interruption (default: false)

RealtimeAzureSemanticVad

Azure semantic VAD (default variant).

[Expand table](#)

Field	Type	Description
type	string	Must be "azure_semantic_vad"
threshold	number	Optional. Activation threshold
prefix_padding_ms	integer	Optional. Audio padding before speech
silence_duration_ms	integer	Optional. Silence duration for speech end
end_of_utterance_detection	RealtimeEOUDetection	Optional. EOU detection config
neg_threshold	number	Optional. Negative threshold
speech_duration_ms	integer	Optional. Minimum speech duration
window_size	integer	Optional. Analysis window size
distinct_ci_phones	integer	Optional. Distinct CI phones requirement
require_vowel	boolean	Optional. Require vowel in speech
remove_filler_words	boolean	Optional. Remove filler words (default: false)
languages	string[]	Optional. Supported languages
auto_truncate	boolean	Optional. Auto-truncate on interruption (default: false)

RealtimeEOUDetection

End-of-utterance semantic detection configuration.

RealtimeAzureSemanticDetection

Azure semantic end-of-utterance detection (default).

[\[+\] Expand table](#)

Field	Type	Description
model	string	Must be "semantic_detection_v1"
threshold	number	Optional. Detection threshold
timeout	number	Optional. Detection timeout

Avatar Configuration

RealtimeAvatarConfig

Configuration for avatar streaming and behavior.

[\[+\] Expand table](#)

Field	Type	Description
ice_servers	RealtimeIceServer[]	Optional. ICE servers for WebRTC
character	string	Character name or ID for the avatar
style	string	Optional. Avatar style (emotional tone, speaking style)
customized	boolean	Whether the avatar is customized
video	RealtimeVideoParams	Optional. Video configuration

RealtimeIceServer

ICE server configuration for WebRTC connection negotiation.

[\[+\] Expand table](#)

Field	Type	Description
urls	string[]	ICE server URLs (TURN or STUN endpoints)
username	string	Optional. Username for authentication
credential	string	Optional. Credential for authentication

RealtimeVideoParams

Video streaming parameters for avatar.

[\[+\] Expand table](#)

Field	Type	Description
bitrate	integer	Optional. Bitrate in bits per second (default: 2000000)
codec	string	Optional. Video codec, currently only <code>h264</code> (default: <code>h264</code>)
crop	RealtimeVideoCrop	Optional. Cropping settings
resolution	RealtimeVideoResolution	Optional. Resolution settings

RealtimeVideoCrop

Video crop rectangle definition.

[\[+\] Expand table](#)

Field	Type	Description
top_left	integer[]	Top-left corner [x, y], non-negative integers
bottom_right	integer[]	Bottom-right corner [x, y], non-negative integers

RealtimeVideoResolution

Video resolution specification.

[\[+\] Expand table](#)

Field	Type	Description
width	integer	Width in pixels (must be > 0)

Field	Type	Description
height	integer	Height in pixels (must be > 0)

Animation Configuration

RealtimeAnimation

Configuration for animation outputs including blendshapes and visemes.

[\[+\] Expand table](#)

Field	Type	Description
model_name	string	Optional. Animation model name (default: <code>"default"</code>)
outputs	RealtimeAnimationOutputType[]	Optional. Output types (default: <code>["blendshapes"]</code>)

RealtimeAnimationOutputType

Types of animation data to output.

Allowed Values:

- `blendshapes` - Facial blendshapes data
- `viseme_id` - Viseme identifier data

Session Configuration

RealtimeRequestSession

Session configuration object used in `session.update` events.

[\[+\] Expand table](#)

Field	Type	Description
model	string	Optional. Model name to use
modalities	RealtimeModality[]	Optional. The supported modalities for the session. For example, "modalities": <code>["text", "audio"]</code> is the default

Field	Type	Description
		setting that enables both text and audio modalities. To enable only text, set "modalities": ["text"]. To enable avatar output, set "modalities": ["text", "audio", "avatar"]. You can't enable only audio.
animation	RealtimeAnimation	Optional. Animation configuration
voice	RealtimeVoice	Optional. Voice configuration
instructions	string	Optional. System instructions for the model. The instructions could guide the output audio if OpenAI voices are used but may not apply to Azure voices.
input_audio_sampling_rate	integer	Optional. Input audio sampling rate in Hz (default: 24000 for <code>pcm16</code> , 8000 for <code>g711_ulaw</code> and <code>g711_alaw</code>)
input_audio_format	RealtimeAudioFormat	Optional. Input audio format (default: <code>pcm16</code>)
output_audio_format	RealtimeOutputAudioFormat	Optional. Output audio format (default: <code>pcm16</code>)
input_audio_noise_reduction	RealtimeInputAudioNoiseReductionSettings	Configuration for input audio noise reduction. This can be set to null to turn off. Noise reduction filters audio added to the input audio buffer before it is sent to VAD and the model. Filtering the audio can improve VAD and turn detection accuracy (reducing false positives) and model performance by improving perception of the input audio. This property is nullable.
input_audio_echo_cancellation	RealtimeInputAudioEchoCancellationSettings	Configuration for input audio echo cancellation. This can be set to null to turn off. This service side echo cancellation can help improve the quality

Field	Type	Description
		of the input audio by reducing the impact of echo and reverberation.
		This property is nullable.
input_audio_transcription	RealtimeAudioInputTranscriptionSettings	The configuration for input audio transcription. The configuration is null (off) by default. Input audio transcription isn't native to the model, since the model consumes audio directly. Transcription runs asynchronously through the <code>/audio/transcriptions</code> endpoint and should be treated as guidance of input audio content rather than precisely what the model heard. For additional guidance to the transcription service, the client can optionally set the language and prompt for transcription.
		This property is nullable.
turn_detection	RealtimeTurnDetection	The turn detection settings for the session. This can be set to null to turn off.
		This property is nullable.
tools	array of RealtimeTool	The tools available to the model for the session.
tool_choice	RealtimeToolChoice	The tool choice for the session. Allowed values: <code>auto</code> , <code>none</code> , and <code>required</code> . Otherwise, you can specify the name of the function to use.
temperature	number	The sampling temperature for the model. The allowed temperature values are limited to [0.6, 1.2]. Defaults to 0.8.

Field	Type	Description
max_response_output_tokens	integer or "inf"	The maximum number of output tokens per assistant response, inclusive of tool calls. Specify an integer between 1 and 4096 to limit the output tokens. Otherwise, set the value to "inf" to allow the maximum number of tokens.
		For example, to limit the output tokens to 1000, set <code>"max_response_output_tokens": 1000</code> . To allow the maximum number of tokens, set <code>"max_response_output_tokens": "inf"</code> .
		Defaults to <code>"inf"</code> .
avatar	RealtimeAvatarConfig	Optional. Avatar configuration
output_audio_timestamp_types	RealtimeAudioTimestampType[]	Optional. Timestamp types for output audio

RealtimeModality

Supported session modalities.

Allowed Values:

- `text` - Text input/output
- `audio` - Audio input/output
- `animation` - Animation output
- `avatar` - Avatar video output

RealtimeAudioTimestampType

Output timestamp types supported in audio response content.

Allowed Values:

- `word` - Timestamps per word in the output audio

Tool Configuration

RealtimeTool

Tool definition for function calling.

[\[\] Expand table](#)

Field	Type	Description
type	string	Must be "function"
name	string	Function name
description	string	Function description and usage guidelines
parameters	object	Function parameters as JSON schema object

RealtimeToolChoice

Tool selection strategy.

This can be:

- "auto" - Let the model choose
- "none" - Don't use tools
- "required" - Must use a tool
- { "type": "function", "name": "function_name" } - Use specific function

RealtimeConversationResponseItem

This is a union type that can be one of the following:

RealtimeConversationUserMessageItem

User message item.

[\[\] Expand table](#)

Field	Type	Description
id	string	The unique ID of the item.
type	string	Must be "message"

Field	Type	Description
object	string	Must be "conversation.item"
role	string	Must be "user"
content	RealtimeInputTextContentPart[]	The content of the message.
status	RealtimeItemStatus	The status of the item.

RealtimeConversationAssistantMessageItem

Assistant message item.

[\[+\] Expand table](#)

Field	Type	Description
id	string	The unique ID of the item.
type	string	Must be "message"
object	string	Must be "conversation.item"
role	string	Must be "assistant"
content	RealtimeOutputTextContentPart[] or RealtimeOutputAudioContentPart[]	The content of the message.
status	RealtimeItemStatus	The status of the item.

RealtimeConversationSystemMessageItem

System message item.

[\[+\] Expand table](#)

Field	Type	Description
id	string	The unique ID of the item.
type	string	Must be "message"
object	string	Must be "conversation.item"
role	string	Must be "system"
content	RealtimeInputTextContentPart[]	The content of the message.

Field	Type	Description
status	RealtimeItemStatus	The status of the item.

RealtimeConversationFunctionCallItem

Function call request item.

[Expand table](#)

Field	Type	Description
id	string	The unique ID of the item.
type	string	Must be "function_call"
object	string	Must be "conversation.item"
name	string	The name of the function to call.
arguments	string	The arguments for the function call as a JSON string.
call_id	string	The unique ID of the function call.
status	RealtimeItemStatus	The status of the item.

RealtimeConversationFunctionCallOutputItem

Function call response item.

[Expand table](#)

Field	Type	Description
id	string	The unique ID of the item.
type	string	Must be "function_call_output"
object	string	Must be "conversation.item"
name	string	The name of the function that was called.
output	string	The output of the function call.
call_id	string	The unique ID of the function call.
status	RealtimeItemStatus	The status of the item.

RealtimeItemStatus

Status of conversation items.

Allowed Values:

- `in_progress` - Currently being processed
- `completed` - Successfully completed
- `incomplete` - Incomplete (interrupted or failed)

RealtimeContentPart

Content part within a message.

RealtimeInputTextContentPart

Text content part.

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be <code>"input_text"</code>
text	string	The text content

RealtimeOutputTextContentPart

Text content part.

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be <code>"text"</code>
text	string	The text content

RealtimeInputAudioContentPart

Audio content part.

[\[+\] Expand table](#)

Field	Type	Description
type	string	Must be "input_audio"
audio	string	Optional. Base64-encoded audio data
transcript	string	Optional. Audio transcript

RealtimeOutputAudioContentPart

Audio content part.

[Expand table](#)

Field	Type	Description
type	string	Must be "audio"
audio	string	Base64-encoded audio data
transcript	string	Optional. Audio transcript

Response Objects

RealtimeResponse

Response object representing a model inference response.

[Expand table](#)

Field	Type	Description
id	string	Optional. Response ID
object	string	Optional. Always "realtime.response"
status	RealtimeResponseStatus	Optional. Response status
status_details	RealtimeResponseStatusDetails	Optional. Status details
output	RealtimeConversationResponseItem[]	Optional. Output items
usage	RealtimeUsage	Optional. Token usage statistics
conversation_id	string	Optional. Associated conversation ID
voice	RealtimeVoice	Optional. Voice used for response

Field	Type	Description
modalities	string[]	Optional. Modalities used
output_audio_format	RealtimeOutputAudioFormat	Optional. Audio format used
temperature	number	Optional. Temperature used
max_response_output_tokens	integer or "inf"	Optional. Max tokens used

RealtimeResponseStatus

Response status values.

Allowed Values:

- `in_progress` - Response is being generated
- `completed` - Response completed successfully
- `cancelled` - Response was cancelled
- `incomplete` - Response incomplete (interrupted)
- `failed` - Response failed with error

RealtimeUsage

Token usage statistics.

 Expand table

Field	Type	Description
total_tokens	integer	Total tokens used
input_tokens	integer	Input tokens used
output_tokens	integer	Output tokens generated
input_token_details	TokenDetails	Breakdown of input tokens
output_token_details	TokenDetails	Breakdown of output tokens

TokenDetails

Detailed token usage breakdown.

 Expand table

Field	Type	Description
cached_tokens	integer	Optional. Cached tokens used
text_tokens	integer	Optional. Text tokens used
audio_tokens	integer	Optional. Audio tokens used

Error Handling

RealtimeErrorDetails

Error information object.

[] [Expand table](#)

Field	Type	Description
type	string	Error type (e.g., <code>"invalid_request_error"</code> , <code>"server_error"</code>)
code	string	Optional. Specific error code
message	string	Human-readable error description
param	string	Optional. Parameter related to the error
event_id	string	Optional. ID of the client event that caused the error

RealtimeConversationRequestItem

You use the `RealtimeConversationRequestItem` object to create a new item in the conversation via the [conversation.item.create](#) event.

This is a union type that can be one of the following:

RealtimeSystemMessageItem

A system message item.

[] [Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>message</code>

Field	Type	Description
role	string	The role of the message. Allowed values: <code>system</code>
content	array of RealtimeInputTextContentPart	The content of the message.
id	string	The unique ID of the item. The client can specify the ID to help manage server-side context. If the client doesn't provide an ID, the server generates one.

RealtimeUserMessageItem

A user message item.

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>message</code>
role	string	The role of the message. Allowed values: <code>user</code>
content	array of RealtimeInputTextContentPart or RealtimeInputAudioContentPart	The content of the message.
id	string	The unique ID of the item. The client can specify the ID to help manage server-side context. If the client doesn't provide an ID, the server generates one.

RealtimeAssistantMessageItem

An assistant message item.

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>message</code>

Field	Type	Description
role	string	The role of the message. Allowed values: <code>assistant</code>
content	array of RealtimeOutputTextContentPart	The content of the message.

RealtimeFunctionCallItem

A function call item.

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>function_call</code>
name	string	The name of the function to call.
arguments	string	The arguments of the function call as a JSON string.
call_id	string	The ID of the function call item.
id	string	The unique ID of the item. The client can specify the ID to help manage server-side context. If the client doesn't provide an ID, the server generates one.

RealtimeFunctionCallOutputItem

A function call output item.

[\[\]](#) [Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>function_call_output</code>
call_id	string	The ID of the function call item.
output	string	The output of the function call, this is a free-form string with the function result, also could be empty.
id	string	The unique ID of the item. If the client doesn't provide an ID, the server generates one.

RealtimeFunctionTool

The definition of a function tool as used by the realtime endpoint.

[\[+\] Expand table](#)

Field	Type	Description
type	string	The type of the tool. Allowed values: <code>function</code>
name	string	The name of the function.
description	string	The description of the function, including usage guidelines. For example, "Use this function to get the current time."
parameters	object	The parameters of the function in the form of a JSON object.

RealtimeItemStatus

Allowed Values:

- `in_progress`
- `completed`
- `incomplete`

RealtimeResponseAudioContentPart

[\[+\] Expand table](#)

Field	Type	Description
type	string	The type of the content part. Allowed values: <code>audio</code>
transcript	string	The transcript of the audio. This property is nullable.

RealtimeResponseFunctionCallItem

[\[+\] Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>function_call</code>
name	string	The name of the function call item.
call_id	string	The ID of the function call item.
arguments	string	The arguments of the function call item.
status	RealtimeItemStatus	The status of the item.

RealtimeResponseFunctionCallOutputItem

[\[+\] Expand table](#)

Field	Type	Description
type	string	The type of the item. Allowed values: <code>function_call_output</code>
call_id	string	The ID of the function call item.
output	string	The output of the function call item.

RealtimeResponseOptions

[\[+\] Expand table](#)

Field	Type	Description
modalities	array	The modalities that the session supports. Allowed values: <code>text</code> , <code>audio</code> For example, <code>"modalities": ["text", "audio"]</code> is the default setting that enables both text and audio modalities. To enable only text, set <code>"modalities": ["text"]</code> . You can't enable only audio.
instructions	string	The instructions (the system message) to guide the model's responses.
voice	RealtimeVoice	The voice used for the model response for the session.

Field	Type	Description
		Once the voice is used in the session for the model's audio response, it can't be changed.
tools	array of RealtimeTool	The tools available to the model for the session.
tool_choice	RealtimeToolChoice	The tool choice for the session.
temperature	number	The sampling temperature for the model. The allowed temperature values are limited to [0.6, 1.2]. Defaults to 0.8.
max_response_output_tokens	integer or "inf"	<p>The maximum number of output tokens per assistant response, inclusive of tool calls.</p> <p>Specify an integer between 1 and 4096 to limit the output tokens. Otherwise, set the value to "inf" to allow the maximum number of tokens.</p> <p>For example, to limit the output tokens to 1000, set <code>"max_response_output_tokens": 1000</code>. To allow the maximum number of tokens, set <code>"max_response_output_tokens": "inf"</code>.</p> <p>Defaults to <code>"inf"</code>.</p>
conversation	string	<p>Controls which conversation the response is added to. The supported values are <code>auto</code> and <code>none</code>.</p> <p>The <code>auto</code> value (or not setting this property) ensures that the contents of the response are added to the session's default conversation.</p> <p>Set this property to <code>none</code> to create an out-of-band response where items won't be added to the default conversation.</p> <p>Defaults to <code>"auto"</code></p>
metadata	map	<p>Set of up to 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.</p> <p>For example: <code>metadata: { topic: "classification" }</code></p>

RealtimeResponseSession

The `RealtimeResponseSession` object represents a session in the Realtime API. It's used in some of the server events, such as:

- `session.created`
- `session.updated`

[+] [Expand table](#)

Field	Type	Description
object	string	The session object. Allowed values: <code>realtime.session</code>
id	string	The unique ID of the session.
model	string	The model used for the session.
modalities	array	The modalities that the session supports. Allowed values: <code>text</code> , <code>audio</code> For example, <code>"modalities": ["text", "audio"]</code> is the default setting that enables both text and audio modalities. To enable only text, set <code>"modalities": ["text"]</code> . You can't enable only audio.
instructions	string	The instructions (the system message) to guide the model's text and audio responses. Here are some example instructions to help guide content and format of text and audio responses: <code>"instructions": "be succinct"</code> <code>"instructions": "act friendly"</code> <code>"instructions": "here are examples of good responses"</code> Here are some example instructions to help guide audio behavior:

Field	Type	Description
		<pre>"instructions": "talk quickly" "instructions": "inject emotion into your voice" "instructions": "laugh frequently"</pre> <p>While the model might not always follow these instructions, they provide guidance on the desired behavior.</p>
voice	RealtimeVoice	The voice used for the model response for the session. Once the voice is used in the session for the model's audio response, it can't be changed.
input_audio_sampling_rate	integer	The sampling rate for the input audio.
input_audio_format	RealtimeAudioFormat	The format for the input audio.
output_audio_format	RealtimeAudioFormat	The format for the output audio.
input_audio_transcription	RealtimeAudioInputTranscriptionSettings	The settings for audio input transcription. This property is nullable.
turn_detection	RealtimeTurnDetection	The turn detection settings for the session. This property is nullable.
tools	array of RealtimeTool	The tools available to the model for the session.
tool_choice	RealtimeToolChoice	The tool choice for the session.
temperature	number	The sampling temperature for the model. The allowed temperature values are limited to [0.6, 1.2]. Defaults to 0.8.
max_response_output_tokens	integer or "inf"	The maximum number of output tokens per assistant response, inclusive of tool calls. Specify an integer between 1 and 4096 to limit the output

Field	Type	Description
		<p>tokens. Otherwise, set the value to "inf" to allow the maximum number of tokens.</p> <p>For example, to limit the output tokens to 1000, set <code>"max_response_output_tokens": 1000</code>. To allow the maximum number of tokens, set <code>"max_response_output_tokens": "inf"</code>.</p>

RealtimeResponseStatusDetails

[] Expand table

Field	Type	Description
type	RealtimeResponseStatus	The status of the response.

RealtimeRateLimitsItem

[] Expand table

Field	Type	Description
name	string	The rate limit property name that this item includes information about.
limit	integer	The maximum configured limit for this rate limit property.
remaining	integer	The remaining quota available against the configured limit for this rate limit property.
reset_seconds	number	The remaining time, in seconds, until this rate limit property is reset.

Related Resources

- Try the [Voice live quickstart](#)
- Try the [Voice live agents quickstart](#)
- Learn more about [How to use the Voice live API](#)

Azure VoiceLive client library for .NET - version 1.0.0

10/13/2025

Azure VoiceLive is a managed service that enables low-latency, high-quality speech-to-speech interactions for voice agents. The API consolidates speech recognition, generative AI, and text-to-speech functionalities into a single, unified interface, providing an end-to-end solution for creating seamless voice-driven experiences.

Use the client library to:

- Create real-time voice assistants and conversational agents
- Build speech-to-speech applications with minimal latency
- Integrate advanced conversational features like noise suppression and echo cancellation
- Leverage multiple AI models (GPT-4o, GPT-4o-mini, Phi) for different use cases
- Implement function calling and tool integration for dynamic responses
- Create avatar-enabled voice interactions with visual components

[Source code](#) | [Package \(NuGet\)](#) | [API reference documentation](#) | [Product documentation](#)
| [Samples](#)

Getting started

This section includes everything a developer needs to install the package and create their first VoiceLive client connection.

Install the package

Install the client library for .NET with [NuGet](#):

.NET CLI

```
dotnet add package Azure.AI.VoiceLive
```

Prerequisites

You must have an [Azure subscription](#) and an [Azure AI Foundry resource](#) to use this service.

The client library targets .NET Standard 2.0 and .NET 8.0, providing compatibility with a wide range of .NET implementations. To use the async streaming features demonstrated in the

examples, you'll need .NET 6.0 or later.

Authenticate the client

The Azure.AI.VoiceLive client supports two authentication methods:

1. **Microsoft Entra ID (recommended)**: Use token-based authentication
2. **API Key**: Use your resource's API key

Authentication with Microsoft Entra ID

C#

```
Uri endpoint = new Uri("https://your-resource.cognitiveservices.azure.com");
DefaultAzureCredential credential = new DefaultAzureCredential();
VoiceLiveClient client = new VoiceLiveClient(endpoint, credential);
```

Authentication with API Key

C#

```
Uri endpoint = new Uri("https://your-resource.cognitiveservices.azure.com");
AzureKeyCredential credential = new AzureKeyCredential("your-api-key");
VoiceLiveClient client = new VoiceLiveClient(endpoint, credential);
```

For the recommended keyless authentication with Microsoft Entra ID, you need to:

1. Assign the `Cognitive Services User` role to your user account or managed identity in the Azure portal under Access control (IAM) > Add role assignment
2. Use a `TokenCredential` implementation - the SDK automatically handles token acquisition and refresh with the appropriate scope

Service API versions

The client library targets the latest service API version by default. You can optionally specify the API version when creating a client instance.

Select a service API version

You have the flexibility to explicitly select a supported service API version when instantiating a client by configuring its associated options:

C#

```
Uri endpoint = new Uri("https://your-resource.cognitiveservices.azure.com");
DefaultAzureCredential credential = new DefaultAzureCredential();
VoiceLiveClientOptions options = new
VoiceLiveClientOptions(VoiceLiveClientOptions.ServiceVersion.V2025_10_01);
VoiceLiveClient client = new VoiceLiveClient(endpoint, credential, options);
```

Key concepts

The Azure.AI.VoiceLive client library provides several key classes for real-time voice interactions:

VoiceLiveClient

The primary entry point for the Azure.AI.VoiceLive service. Use this client to establish sessions and configure authentication.

VoiceLiveSession

Represents an active WebSocket connection to the VoiceLive service. This class handles bidirectional communication, allowing you to send audio input and receive audio output, text transcriptions, and other events in real-time.

Session Configuration

The service uses session configuration to control various aspects of the voice interaction:

- **Turn Detection:** Configure how the service detects when users start and stop speaking
- **Audio Processing:** Enable noise suppression and echo cancellation
- **Voice Selection:** Choose from standard Azure voices, high-definition voices, or custom voices
- **Model Selection:** Select the AI model (GPT-4o, GPT-4o-mini, Phi variants) that best fits your needs

Models and Capabilities

The VoiceLive API supports multiple AI models with different capabilities:

 Expand table

Model	Description	Use Case
gpt-4o-realtime-preview	GPT-4o with real-time audio processing	High-quality conversational AI
gpt-4o-mini-realtime-preview	Lightweight GPT-4o variant	Fast, efficient interactions
phi4-mm-realtime	Phi model with multimodal support	Cost-effective voice applications

Conversational Enhancements

The VoiceLive API provides Azure-specific enhancements:

- **Azure Semantic VAD:** Advanced voice activity detection that removes filler words
- **Noise Suppression:** Reduces environmental background noise
- **Echo Cancellation:** Removes echo from the model's own voice
- **End-of-Turn Detection:** Allows natural pauses without premature interruption

Thread safety

We guarantee that all client instance methods are thread-safe and independent of each other ([guideline ↴](#)). This ensures that the recommendation of reusing client instances is always safe, even across threads.

Additional concepts

[Client options ↴](#) | [Accessing the response ↴](#) | [Long-running operations ↴](#) | [Handling failures ↴](#) | [Diagnostics ↴](#) | [Mocking ↴](#) | [Client lifetime ↴](#)

Examples

You can familiarize yourself with different APIs using [Samples ↴](#).

Basic voice assistant

C#

```
// Create the VoiceLive client
Uri endpoint = new Uri("https://your-resource.cognitiveservices.azure.com");
DefaultAzureCredential credential = new DefaultAzureCredential();
VoiceLiveClient client = new VoiceLiveClient(endpoint, credential);
```

```

var model = "gpt-4o-mini-realtime-preview"; // Specify the model to use
// Start a new session
VoiceLiveSession session = await
client.StartSessionAsync(model).ConfigureAwait(false);

// Configure session for voice conversation
VoiceLiveSessionOptions sessionOptions = new()
{
    Model = model,
    Instructions = "You are a helpful AI assistant. Respond naturally and
conversationally.",
    Voice = new AzureStandardVoice("en-US-AvaNeural"),
    TurnDetection = new AzureSemanticVadTurnDetection()
    {
        Threshold = 0.5f,
        PrefixPadding = TimeSpan.FromMilliseconds(300),
        SilenceDuration = TimeSpan.FromMilliseconds(500)
    },
    InputAudioFormat = InputAudioFormat.Pcm16,
    OutputAudioFormat = OutputAudioFormat.Pcm16
};

// Ensure modalities include audio
sessionOptions.Modalities.Clear();
sessionOptions.Modalities.Add(InteractionModality.Text);
sessionOptions.Modalities.Add(InteractionModality.Audio);

await session.ConfigureSessionAsync(sessionOptions).ConfigureAwait(false);

// Process events from the session
await foreach (SessionUpdate serverEvent in
session.GetUpdatesAsync().ConfigureAwait(false))
{
    if (serverEvent is SessionUpdateResponseAudioDelta audioDelta)
    {
        // Play audio response
        byte[] audioData = audioDelta.Delta.ToArray();
        // ... audio playback logic
    }
    else if (serverEvent is SessionUpdateResponseTextDelta textDelta)
    {
        // Display text response
        Console.WriteLine(textDelta.Delta);
    }
}

```

Configuring custom voice and advanced features

C#

```

VoiceLiveSessionOptions sessionOptions = new()
{
    Model = model,
    Instructions = "You are a customer service representative. Be helpful and
professional.",
    Voice = new AzureCustomVoice("your-custom-voice-name", "your-custom-voice-
endpoint-id")
    {
        Temperature = 0.8f
    },
    TurnDetection = new AzureSemanticVadTurnDetection()
    {
        RemoveFillerWords = true
    },
    InputAudioFormat = InputAudioFormat.Pcm16,
    OutputAudioFormat = OutputAudioFormat.Pcm16
};

// Ensure modalities include audio
sessionOptions.Modalities.Clear();
sessionOptions.Modalities.Add(InteractionModality.Text);
sessionOptions.Modalities.Add(InteractionModality.Audio);

await session.ConfigureSessionAsync(sessionOptions).ConfigureAwait(false);

```

Function calling example

C#

```

// Define a function for the assistant to call
var getCurrentWeatherFunction = new
VoiceLiveFunctionDefinition("get_current_weather")
{
    Description = "Get the current weather for a given location",
    Parameters = BinaryData.FromString("""
    {
        "type": "object",
        "properties": {
            "location": {
                "type": "string",
                "description": "The city and state or country"
            }
        },
        "required": ["location"]
    }
""")
};

VoiceLiveSessionOptions sessionOptions = new()
{
    Model = model,
    Instructions = "You are a weather assistant. Use the get_current_weather

```

```

        function to help users with weather information.",
        Voice = new AzureStandardVoice("en-US-AvaNeural"),
        InputAudioFormat = InputAudioFormat.Pcm16,
        OutputAudioFormat = OutputAudioFormat.Pcm16
    };

    // Add the function tool
    sessionOptions.Tools.Add(getCurrentWeatherFunction);

    // Ensure modalities include audio
    sessionOptions.Modalities.Clear();
    sessionOptions.Modalities.Add(InteractionModality.Text);
    sessionOptions.Modalities.Add(InteractionModality.Audio);

    await session.ConfigureSessionAsync(sessionOptions).ConfigureAwait(false);

```

Function Response Handling

C#

```

// Process events from the session
await foreach (SessionUpdate serverEvent in
session.GetUpdatesAsync().ConfigureAwait(false))
{
    if (serverEvent is SessionUpdateResponseFunctionCallArgumentsDone
functionCall)
    {
        if (functionCall.Name == "get_current_weather")
        {
            // Extract parameters from the function call
            var parametersString = functionCall.Arguments;
            var parameters =
System.Text.Json.JsonSerializer.Deserialize<Dictionary<string, string>>
(parametersString);

            string location = parameters != null ? parameters["location"] :
string.Empty;

            // Call your external weather service here and get the result
            string weatherInfo = $"The current weather in {location} is sunny with
a temperature of 75°F.';

            // Send the function response back to the session
            await session.AddItemAsync(new
FunctionCallOutputItem(functionCall.CallId, weatherInfo)).ConfigureAwait(false);

            // Start the next response.
            await session.StartResponseAsync().ConfigureAwait(false);
        }
    }
}

```

Adding a user text message

C#

```
// Add a user message to the session
await session.AddItemAsync(new UserMessageItem("Hello, can you help me with my
account?")).ConfigureAwait(false);
// Start the response from the assistant
await session.StartResponseAsync().ConfigureAwait(false);
```

Troubleshooting

Common errors and exceptions

Authentication Errors: If you receive authentication errors, verify that:

- Your Azure AI Foundry resource is correctly configured
- Your API key or credential has the necessary permissions
- The endpoint URL is correct and accessible

WebSocket Connection Issues: VoiceLive uses WebSocket connections. Ensure that:

- Your network allows WebSocket connections
- Firewall rules permit connections to *.cognitiveservices.azure.com
- The service is available in your selected region

Audio Processing Errors: For audio-related issues:

- Verify audio input format is supported (16kHz or 24kHz PCM)
- Check that audio devices are accessible and functioning
- Ensure proper audio codec configuration

Logging and diagnostics

Enable logging to help diagnose issues:

C#

```
using Azure.Core.Diagnostics;

// Enable logging for Azure SDK
using AzureEventSourceListener listener =
AzureEventSourceListener.CreateConsoleLogger();
```

Rate limiting and throttling

The VoiceLive service implements rate limiting based on:

- Concurrent connections per resource
- Token consumption rates
- Model-specific limits

Implement appropriate retry logic and connection management to handle throttling gracefully.

Next steps

- Explore the comprehensive [samples](#) including basic voice assistants and customer service bots
- Learn about [voice customization](#) to create unique brand voices
- Understand [avatar integration](#) for visual voice experiences
- Review the [VoiceLive API documentation](#) for advanced configuration options

Contributing

This project welcomes contributions and suggestions. Most contributions require you to agree to a Contributor License Agreement (CLA) declaring that you have the right to, and actually do, grant us the rights to use your contribution. For details, visit <https://cla.opensource.microsoft.com>.

When you submit a pull request, a CLA bot will automatically determine whether you need to provide a CLA and decorate the PR appropriately (e.g., status check, comment). Simply follow the instructions provided by the bot. You will only need to do this once across all repos using our CLA.

This project has adopted the [Microsoft Open Source Code of Conduct](#). For more information see the [Code of Conduct FAQ](#) or contact opencode@microsoft.com with any additional questions or comments.

Azure AI VoiceLive client library for Python

This package provides a **real-time, speech-to-speech** client for Azure AI VoiceLive. It opens a WebSocket session to stream microphone audio to the service and receive typed server events (including audio) for responsive, interruptible conversations.

Status: General Availability (GA). This is a stable release suitable for production use.

Important: As of version 1.0.0, this SDK is **async-only**. The synchronous API has been removed to focus exclusively on async patterns. All examples and samples use `async / await` syntax.

Getting started

Prerequisites

- Python 3.9+
- An Azure subscription
- A VoiceLive resource and endpoint
- A working **microphone** and **speakers/headphones** if you run the voice samples

Install

Install the stable GA version:

Bash

```
# Base install (core client only)
python -m pip install azure-ai-voicelive

# For asynchronous streaming (uses aiohttp)
python -m pip install "azure-ai-voicelive[aiohttp]"

# For voice samples (includes audio processing)
python -m pip install azure-ai-voicelive[aiohttp] pyaudio python-dotenv
```

The SDK provides async-only WebSocket connections using `aiohttp` for optimal performance and reliability.

Authenticate

You can authenticate with an API key or an Azure Active Directory (AAD) token.

API Key Authentication (Quick Start)

Set environment variables in a `.env` file or directly in your environment:

Bash

```
# In your .env file or environment variables
AZURE_VOICELIVE_API_KEY="your-api-key"
AZURE_VOICELIVE_ENDPOINT="your-endpoint"
```

Then, use the key in your code:

Python

```
import asyncio
from azure.core.credentials import AzureKeyCredential
from azure.ai.vocielive import connect

async def main():
    async with connect(
        endpoint="your-endpoint",
        credential=AzureKeyCredential("your-api-key"),
        model="gpt-4o-realtime-preview"
    ) as connection:
        # Your async code here
        pass

asyncio.run(main())
```

AAD Token Authentication

For production applications, AAD authentication is recommended:

Python

```
import asyncio
from azure.identity.aio import DefaultAzureCredential
from azure.ai.vocielive import connect

async def main():
    credential = DefaultAzureCredential()

    async with connect(
        endpoint="your-endpoint",
        credential=credential,
        model="gpt-4o-realtime-preview"
    ) as connection:
```

```
# Your async code here
pass

asyncio.run(main())
```

Key concepts

- **VoiceLiveConnection** – Manages an active async WebSocket connection to the service
- **Session Management** – Configure conversation parameters:
 - **SessionResource** – Update session parameters (voice, formats, VAD) with async methods
 - **RequestSession** – Strongly-typed session configuration
 - **ServerVad** – Configure voice activity detection
 - **AzureStandardVoice** – Configure voice settings
- **Audio Handling:**
 - **InputAudioBufferResource** – Manage audio input to the service with async methods
 - **OutputAudioBufferResource** – Control audio output from the service with async methods
- **Conversation Management:**
 - **ResponseResource** – Create or cancel model responses with async methods
 - **ConversationResource** – Manage conversation items with async methods
- **Error Handling:**
 - **ConnectionError** – Base exception for WebSocket connection errors
 - **ConnectionClosed** – Raised when WebSocket connection is closed
- **Strongly-Typed Events** – Process service events with type safety:
 - `SESSION_UPDATED`, `RESPONSE_AUDIO_DELTA`, `RESPONSE_DONE`
 - `INPUT_AUDIO_BUFFER_SPEECH_STARTED`, `INPUT_AUDIO_BUFFER_SPEECH_STOPPED`
 - `ERROR`, and more

Examples

Basic Voice Assistant (Featured Sample)

The Basic Voice Assistant sample demonstrates full-featured voice interaction with:

- Real-time speech streaming
- Server-side voice activity detection
- Interruption handling

- High-quality audio processing

Bash

```
# Run the basic voice assistant sample
# Requires [aiohttp] for async
python samples/basic_voice_assistant_async.py

# With custom parameters
python samples/basic_voice_assistant_async.py --model gpt-4o-realtime-preview --
voice alloy --instructions "You're a helpful assistant"
```

Minimal example

Python

```
import asyncio
from azure.core.credentials import AzureKeyCredential
from azure.ai.vovcelive.aio import connect
from azure.ai.vovcelive.models import (
    RequestSession, Modality, InputAudioFormat, OutputAudioFormat, ServerVad,
    ServerEventType
)

API_KEY = "your-api-key"
ENDPOINT = "wss://your-endpoint.com/openai realtime"
MODEL = "gpt-4o-realtime-preview"

async def main():
    async with connect(
        endpoint=ENDPOINT,
        credential=AzureKeyCredential(API_KEY),
        model=MODEL,
    ) as conn:
        session = RequestSession(
            modalities=[Modality.TEXT, Modality.AUDIO],
            instructions="You are a helpful assistant.",
            input_audio_format=InputAudioFormat.PCM16,
            output_audio_format=OutputAudioFormat.PCM16,
            turn_detection=ServerVad(
                threshold=0.5,
                prefix_padding_ms=300,
                silence_duration_ms=500
            ),
        )
        await conn.session.update(session=session)

    # Process events
    async for evt in conn:
        print(f"Event: {evt.type}")
        if evt.type == ServerEventType.RESPONSE_DONE:
            break
```

```
asyncio.run(main())
```

Available Voice Options

Azure Neural Voices

Python

```
# Use Azure Neural voices
voice_config = AzureStandardVoice(
    name="en-US-AvaNeural", # Or another voice name
    type="azure-standard"
)
```

Popular voices include:

- `en-US-AvaNeural` - Female, natural and professional
- `en-US-JennyNeural` - Female, conversational
- `en-US-GuyNeural` - Male, professional

OpenAI Voices

Python

```
# Use OpenAI voices (as string)
voice_config = "alloy" # Or another OpenAI voice
```

Available OpenAI voices:

- `alloy` - Versatile, neutral
- `echo` - Precise, clear
- `fable` - Animated, expressive
- `onyx` - Deep, authoritative
- `nova` - Warm, conversational
- `shimmer` - Optimistic, friendly

Handling Events

Python

```

async for event in connection:
    if event.type == ServerEventType.SESSION_UPDATED:
        print(f"Session ready: {event.session.id}")
        # Start audio capture

    elif event.type == ServerEventType.INPUT_AUDIO_BUFFER_SPEECH_STARTED:
        print("User started speaking")
        # Stop playback and cancel any current response

    elif event.type == ServerEventType.RESPONSE_AUDIO_DELTA:
        # Play the audio chunk
        audio_bytes = event.delta

    elif event.type == ServerEventType.ERROR:
        print(f"Error: {event.error.message}")

```

Troubleshooting

Connection Issues

- **WebSocket connection errors (1006/timeout):**

Verify `AZURE_VOICELIVE_ENDPOINT`, network rules, and that your credential has access.

- **Missing WebSocket dependencies:**

If you see import errors, make sure you have installed the package: `pip install azure-ai-voicelive[aiohttp]`

- **Auth failures:**

For API key, double-check `AZURE_VOICELIVE_API_KEY`. For AAD, ensure the identity is authorized.

Audio Device Issues

- **No microphone/speaker detected:**

Check device connections and permissions. On headless CI environments, audio samples can't run.

- **Audio library installation problems:**

On Linux/macOS you may need PortAudio:

Bash

```

# Debian/Ubuntu
sudo apt-get install -y portaudio19-dev libasound2-dev

```

```
# macOS (Homebrew)  
brew install portaudio
```

Enable Verbose Logging

Python

```
import logging  
logging.basicConfig(level=logging.DEBUG)
```

Next steps

1. Run the featured sample:

- Try `samples/basic_voice_assistant_async.py` for a complete voice assistant implementation

2. Customize your implementation:

- Experiment with different voices and parameters
- Add custom instructions for specialized assistants
- Integrate with your own audio capture/playback systems

3. Advanced scenarios:

- Add function calling support
- Implement tool usage
- Create multi-turn conversations with history

4. Explore other samples:

- Check the `samples/` directory for specialized examples
- See `samples/README.md` for a full list of samples

Contributing

This project follows the Azure SDK guidelines. If you'd like to contribute:

1. Fork the repo and create a feature branch
2. Run linters and tests locally
3. Submit a pull request with a clear description of the change

Release notes

Changelogs are available in the package directory.

License

This project is released under the [MIT License](#).

Last updated on 11/04/2025

Azure VoiceLive client library for Java - version 1.0.0-beta.2

The Azure VoiceLive client library for Java enables real-time, bidirectional voice conversations with AI assistants. Built on WebSocket technology, it provides low-latency audio streaming with support for voice activity detection, interruption handling, and flexible authentication.

Use the Azure VoiceLive client library for Java to:

- Create real-time voice conversations with AI assistants
- Stream audio input from microphone with automatic voice activity detection
- Receive and play audio responses with interruption support
- Handle conversational flow with turn detection and session management
- Authenticate using API keys or Azure AD (token credentials)

[Source code][source_code] | [API reference documentation ↗](#) | [Product documentation ↗](#) | [Samples][samples_folder]

Getting started

Prerequisites

- [Java Development Kit \(JDK\)](#) with version 8 or above
- [Azure Subscription ↗](#)
- Azure VoiceLive resource with endpoint and API key

Adding the package to your product

XML

```
<dependency>
    <groupId>com.azure</groupId>
    <artifactId>azure-ai-voicelive</artifactId>
    <version>1.0.0-beta.2</version>
</dependency>
```

Authentication

To interact with the Azure VoiceLive service, you'll need to create an instance of the [VoiceLiveAsyncClient][voicelive_client_async] using [VoiceLiveClientBuilder][voicelive_client_builder]. The client supports two authentication methods:

Authenticate with API Key

Get your Azure VoiceLive API key from the Azure Portal:

Java

```
VoiceLiveAsyncClient client = new VoiceLiveClientBuilder()  
    .endpoint("https://your-resource.openai.azure.com/")  
    .credential(new AzureKeyCredential("your-api-key"))  
    .buildAsyncClient();
```

Authenticate with Azure AD (Token Credential)

Azure SDK for Java supports Azure Identity, making it easy to use Microsoft identity platform for authentication.

First, add the Azure Identity package:

XML

```
<dependency>  
    <groupId>com.azure</groupId>  
    <artifactId>azure-identity</artifactId>  
    <version>1.18.1</version>  
</dependency>
```

Then create a client with DefaultAzureCredential:

Java

```
TokenCredential credential = new DefaultAzureCredentialBuilder().build();  
VoiceLiveAsyncClient client = new VoiceLiveClientBuilder()  
    .endpoint("https://your-resource.openai.azure.com/")  
    .credential(credential)  
    .buildAsyncClient();
```

For development and testing, you can use Azure CLI credentials:

Java

```
TokenCredential credential = new AzureCliCredentialBuilder().build();  
VoiceLiveAsyncClient client = new VoiceLiveClientBuilder()  
    .endpoint("https://your-resource.openai.azure.com/")  
    .credential(credential)  
    .buildAsyncClient();
```

Key concepts

VoiceLiveAsyncClient

The main entry point for interacting with the Azure VoiceLive service. Use the `VoiceLiveClientBuilder` to construct a client instance. The client provides methods to start sessions and manage real-time voice conversations.

VoiceLiveSessionAsyncClient

Represents an active WebSocket connection for bidirectional streaming communication. This async client supports:

- Sending audio input streams via `sendInputAudio()`
- Sending command events via `sendEvent()`
- Receiving server events as a reactive stream (Flux) via `receiveEvents()`
- Graceful shutdown with `close()` or `closeAsync()`

VoiceLiveSessionOptions

Configuration options for customizing session behavior:

- **Model selection:** Specify the AI model (e.g., "gpt-4o-realtime-preview")
- **Voice settings:** Choose from OpenAI voices (Alloy, Ash, Ballad, Coral, Echo, Sage, Shimmer, Verse) or Azure voices
- **Modalities:** Configure text and/or audio interaction modes
- **Turn detection:** Server-side voice activity detection with configurable thresholds
- **Audio formats:** PCM16 input/output with configurable sample rates
- **Audio enhancements:** Noise reduction and echo cancellation
- **Transcription:** Optional input audio transcription using Whisper models

Audio Requirements

The VoiceLive service uses specific audio formats:

- **Sample Rate:** 24kHz (24000 Hz)
- **Bit Depth:** 16-bit PCM
- **Channels:** Mono (1 channel)
- **Format:** Signed PCM, little-endian

Examples

The following sections provide code snippets for common scenarios:

- [Simple voice assistant](#)
- [Configure session options](#)
- [Send audio input](#)
- [Handle event types](#)
- [Voice configuration](#)
- [Complete voice assistant with microphone](#)

Focused Sample Files

For easier learning, explore these focused samples in order:

1. `BasicVoiceConversationSample.java` - Start here to learn the basics

- Minimal setup and session management
- Client creation and configuration
- Basic event handling

2. `AuthenticationMethodsSample.java` - Learn authentication options

- API Key authentication (default)
- Token Credential authentication with `DefaultAzureCredential`

3. `MicrophoneInputSample.java` - Add audio input capability

- Real-time microphone audio capture
- Audio format configuration (24kHz, 16-bit PCM, mono)
- Streaming audio to the service
- Speech detection events

4. `AudioPlaybackSample.java` - Add audio output capability

- Receiving audio responses from service
- Audio playback to speakers
- Response completion tracking

5. `VoiceAssistantSample.java` - Complete production-ready implementation

- Full bidirectional audio streaming
- Voice Activity Detection (VAD) with interruption handling
- Audio transcription with Whisper

- Noise reduction and echo cancellation
- Multi-threaded audio processing

Note: To run audio samples (AudioPlaybackSample, MicrophoneInputSample, VoiceAssistantSample):

Bash

```
mvn exec:java -Dexec.mainClass=com.azure.ai.voicelive.AudioPlaybackSample -Dexec.classpathScope=test
```

These samples use `javax.sound.sampled` for audio I/O.

Simple voice assistant

Create a basic voice assistant session:

Java

```
// Start session with default options
client.startSession("gpt-4o-realtime-preview")
    .flatMap(session -> {
        System.out.println("Session started");

        // Subscribe to receive events
        session.receiveEvents()
            .subscribe(
                event -> System.out.println("Event: " + event.getType()),
                error -> System.err.println("Error: " + error.getMessage())
            );

        return Mono.just(session);
    })
    .block();
```

Configure session options

Customize the session with specific options:

Java

```
// Configure server-side voice activity detection
ServerVadTurnDetection turnDetection = new ServerVadTurnDetection()
    .setThreshold(0.5)                      // Sensitivity threshold (0.0-1.0)
    .setPrefixPaddingMs(300)                  // Audio before speech detection
    .setSilenceDurationMs(500)                // Silence to end turn
    .setInterruptResponse(true)               // Allow user interruptions
    .setAutoTruncate(true)                   // Auto-truncate on interruption
```

```

.setCreateResponse(true); // Auto-create response after turn

// Configure input audio transcription
AudioInputTranscriptionOptions transcription = new AudioInputTranscriptionOptions(
    AudioInputTranscriptionOptionsModel.WHISPER_1);

// Create session options
VoiceLiveSessionOptions options = new VoiceLiveSessionOptions()
    .setInstructions("You are a helpful AI voice assistant. Respond naturally and
conversationally.")
    .setVoice(BinaryData.fromObject(new OpenAIVoice(OpenAIVoiceName.ALLOY)))
    .setModalities(Arrays.asList(InteractionModality.TEXT,
InteractionModality.AUDIO))
    .setInputAudioFormat(InputAudioFormat.PCM16)
    .setOutputAudioFormat(OutputAudioFormat.PCM16)
    .setInputAudioSamplingRate(24000)
    .setInputAudioNoiseReduction(new
AudioNoiseReduction(AudioNoiseReductionType.NEAR_FIELD))
    .setInputAudioEchoCancellation(new AudioEchoCancellation())
    .setInputAudioTranscription(transcription)
    .setTurnDetection(turnDetection);

// Start session with options
client.startSession("gpt-4o-realtime-preview")
    .flatMap(session -> {
        // Send session configuration
        ClientEventSessionUpdate updateEvent = new
ClientEventSessionUpdate(options);
        return session.sendEvent(updateEvent).then(Mono.just(session));
    })
    .subscribe(
        session -> System.out.println("Session configured"),
        error -> System.err.println("Error: " + error.getMessage())
    );

```

Send audio input

Stream audio data to the service:

Java

```

// Send audio chunk
byte[] audioData = readAudioChunk(); // Your audio data in PCM16 format
session.sendInputAudio(BinaryData.fromBytes(audioData))
    .subscribe();

// Send audio from file
try {
    Path audioFile = Paths.get("audio.wav");
    byte[] fileData = Files.readAllBytes(audioFile);
    session.sendInputAudio(BinaryData.fromBytes(fileData))
        .subscribe();
}

```

```

} catch (IOException e) {
    System.err.println("Error reading audio file: " + e.getMessage());
}

```

Handle event types

Process different event types for complete conversation flow:

Java

```

session.receiveEvents()
    .subscribe(event -> {
        ServerEventType eventType = event.getType();

        if (ServerEventType.SESSION_CREATED.equals(eventType)) {
            System.out.println("✓ Session created - ready to start");
        } else if (ServerEventType.SESSION_UPDATED.equals(eventType)) {
            System.out.println("✓ Session configured - starting conversation");
            if (event instanceof SessionUpdateSessionUpdated) {
                SessionUpdateSessionUpdated updated = (SessionUpdateSessionUpdated)
event;
                // Access session configuration details
                String json = BinaryData.fromObject(updated).toString();
                System.out.println("Config: " + json);
            }
        } else if
(ServerEventType.INPUT_AUDIO_BUFFER_SPEECH_STARTED.equals(eventType)) {
            System.out.println("🎤 User started speaking");
        } else if
(ServerEventType.INPUT_AUDIO_BUFFER_SPEECH_STOPPED.equals(eventType)) {
            System.out.println("😢 User stopped speaking - processing...");
        } else if (ServerEventType.RESPONSE_AUDIO_DELTA.equals(eventType)) {
            // Play audio response
            if (event instanceof SessionUpdateResponseAudioDelta) {
                SessionUpdateResponseAudioDelta audioEvent =
(SessionUpdateResponseAudioDelta) event;
                playAudioChunk(audioEvent.getDelta());
            }
        } else if (ServerEventType.RESPONSE_AUDIO_DONE.equals(eventType)) {
            System.out.println("🔊 Assistant finished speaking");
        } else if (ServerEventType.RESPONSE_DONE.equals(eventType)) {
            System.out.println("✅ Response complete - ready for next input");
        } else if (ServerEventType.ERROR.equals(eventType)) {
            if (event instanceof SessionUpdateError) {
                SessionUpdateError errorEvent = (SessionUpdateError) event;
                System.err.println("✖ Error: "
+ errorEvent.getError().getMessage());
            }
        }
    });
}

```

Voice configuration

The SDK supports multiple voice providers:

OpenAI Voices

Java

```
// Use OpenAIVoiceName enum for available voices (ALLOY, ASH, BALLAD, CORAL, ECHO, SAGE, SHIMMER, VERSE)
VoiceLiveSessionOptions options = new VoiceLiveSessionOptions()
    .setVoice(BinaryData.fromObject(new OpenAIVoice(OpenAIVoiceName.ALLOY)));
```

Azure Voices

Azure voices include `AzureStandardVoice`, `AzureCustomVoice`, and `AzurePersonalVoice` (all extend `AzureVoice`):

Java

```
// Azure Standard Voice - use any Azure TTS voice name
// See: https://learn.microsoft.com/azure/ai-services/speech-service/language-support?tabs=tts
VoiceLiveSessionOptions options = new VoiceLiveSessionOptions()
    .setVoice(BinaryData.fromObject(new AzureStandardVoice("en-US-JennyNeural")));

// Azure Custom Voice - requires custom voice name and endpoint ID
VoiceLiveSessionOptions options2 = new VoiceLiveSessionOptions()
    .setVoice(BinaryData.fromObject(new AzureCustomVoice("myCustomVoice",
"myEndpointId")));

// Azure Personal Voice - requires speaker profile ID and model
// Models: DRAGON_LATEST_NEURAL, PHOENIX_LATEST_NEURAL, PHOENIX_V2NEURAL
VoiceLiveSessionOptions options3 = new VoiceLiveSessionOptions()
    .setVoice(BinaryData.fromObject(
        new AzurePersonalVoice("speakerProfileId",
        PersonalVoiceModels.PHOENIX_LATEST_NEURAL)));
```

Complete voice assistant with microphone

A full example demonstrating real-time microphone input and audio playback:

Java

```
String endpoint = System.getenv("AZURE_VOICELIVE_ENDPOINT");
String apiKey = System.getenv("AZURE_VOICELIVE_API_KEY");
```

```

// Create the VoiceLive client
VoiceLiveAsyncClient client = new VoiceLiveClientBuilder()
    .endpoint(endpoint)
    .credential(new AzureKeyCredential(apiKey))
    .buildAsyncClient();

// Configure session options for voice conversation
ServerVadTurnDetection turnDetection = new ServerVadTurnDetection()
    .setThreshold(0.5)
    .setPrefixPaddingMs(300)
    .setSilenceDurationMs(500)
    .setInterruptResponse(true)
    .setAutoTruncate(true)
    .setCreateResponse(true);

AudioInputTranscriptionOptions transcriptionOptions = new
AudioInputTranscriptionOptions(
    AudioInputTranscriptionOptionsModel.WHISPER_1);

VoiceLiveSessionOptions sessionOptions = new VoiceLiveSessionOptions()
    .setInstructions("You are a helpful AI voice assistant.")
    .setVoice(BinaryData.fromObject(new OpenAIVoice(OpenAIVoiceName.ALLOY)))
    .setModalities(Arrays.asList(InteractionModality.TEXT,
InteractionModality.AUDIO))
    .setInputAudioFormat(InputAudioFormat.PCM16)
    .setOutputAudioFormat(OutputAudioFormat.PCM16)
    .setInputAudioSamplingRate(24000)
    .setInputAudioNoiseReduction(new
AudioNoiseReduction(AudioNoiseReductionType.NEAR_FIELD))
    .setInputAudioEchoCancellation(new AudioEchoCancellation())
    .setInputAudioTranscription(transcriptionOptions)
    .setTurnDetection(turnDetection);

// Start session and handle events
client.startSession("gpt-4o-realtime-preview")
    .flatMap(session -> {
        // Subscribe to receive server events
        session.receiveEvents()
            .subscribe(
                event -> handleEvent(event, session),
                error -> System.err.println("Error: " + error.getMessage())
            );
        // Send session configuration
        ClientEventSessionUpdate updateEvent = new
ClientEventSessionUpdate(sessionOptions);
        return session.sendEvent(updateEvent).then(Mono.just(session));
    })
    .block();

```

For complete, runnable implementations, see the [Focused Sample Files](#) section above.

Troubleshooting

Enable client logging

You can set the `AZURE_LOG_LEVEL` environment variable to view logging statements made in the client library. For example, setting `AZURE_LOG_LEVEL=2` would show all informational, warning, and error log messages. The log levels can be found here: [log levels][log_levels].

Common issues

Audio system not available

Ensure your system has a working microphone and speakers/headphones. The VoiceLive service requires:

- **Microphone:** For capturing audio input (24kHz, 16-bit PCM, mono)
- **Speakers:** For playing audio responses (24kHz, 16-bit PCM, mono)

WebSocket connection failures

If you encounter connection issues:

- Verify your endpoint URL is correct
- Check that your API key or token credential is valid
- Ensure your network allows WebSocket connections
- Confirm your Azure VoiceLive resource is properly provisioned

Authentication errors

For API key authentication:

- Verify the `AZURE_VOICELIVE_API_KEY` environment variable is set correctly
- Ensure the API key matches your Azure VoiceLive resource

For token credential authentication:

- Run `az login` before using Azure CLI credentials
- Verify the credential has appropriate permissions for the VoiceLive resource
- Check that the Azure Identity library is properly configured

Default HTTP Client

All client libraries by default use the Netty HTTP client. Adding the above dependency will automatically configure the client library to use the Netty HTTP client. For more information on HTTP client configuration, see the [HTTP clients wiki][http_clients_wiki].

Default SSL library

All client libraries, by default, use the Tomcat-native Boring SSL library to enable native-level performance for SSL operations. The Boring SSL library is an uber jar containing native libraries for Linux / macOS / Windows, and provides better performance compared to the default SSL implementation within the JDK. For more information, including how to reduce the dependency size, refer to the [performance tuning][performance_tuning] section of the wiki.

Next steps

Sample files

All sample files are located in the `src/samples/java/com/azure/ai/voicelive/` directory. See the [Focused Sample Files](#) section for detailed descriptions and running instructions.

Additional documentation

- [Azure VoiceLive product documentation ↗](#)
- [\[Azure SDK for Java\]\[azure_sdk_java\]](#)

Contributing

For details on contributing to this repository, see the [\[contributing guide\]\[contributing\]](#).

1. Fork it
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Add some feature'`)
4. Push to the branch (`git push origin my-new-feature`)
5. Create new Pull Request

Azure VoiceLive client library for JavaScript - version 1.0.0-beta.1

Azure VoiceLive is a managed service that enables low-latency, high-quality speech-to-speech interactions for voice agents. The service consolidates speech recognition, generative AI, and text-to-speech functionalities into a single, unified interface, providing an end-to-end solution for creating seamless voice-driven experiences.

Use the client library to:

- Create real-time voice assistants and conversational agents
- Build speech-to-speech applications with minimal latency
- Integrate advanced conversational features like noise suppression and echo cancellation
- Leverage multiple AI models (GPT-4o, GPT-4o-mini, Phi) for different use cases
- Implement function calling and tool integration for dynamic responses
- Create avatar-enabled voice interactions with visual components

Note: This package supports both browser and Node.js environments. WebSocket connections are used for real-time communication.

Getting started

Currently supported environments

- [LTS versions of Node.js](#)
- Latest versions of Safari, Chrome, Edge and Firefox

Prerequisites

- An [Azure subscription](#)
- An [Azure AI Foundry resource](#) with Voice Live API access

Install the package

Install the Azure VoiceLive client library using npm:

Bash

```
npm install @azure/ai-voicelive
```

Install the identity library

VoiceLive clients authenticate using the Azure Identity Library. Install it as well:

```
Bash
```

```
npm install @azure/identity
```

Configure TypeScript

TypeScript users need to have Node type definitions installed:

```
Bash
```

```
npm install @types/node
```

You also need to enable `compilerOptions.allowSyntheticDefaultImports` in your `tsconfig.json`.

Note that if you have enabled `compilerOptions.esModuleInterop`,
`allowSyntheticDefaultImports` is enabled by default.

JavaScript Bundle

To use this client library in the browser, first you need to use a bundler. For details on how to do this, please refer to our [bundling documentation](#).

Key concepts

VoiceLiveClient

The primary interface for establishing connections to the Azure VoiceLive service. Use this client to authenticate and create sessions for real-time voice interactions.

VoiceLiveSession

Represents an active WebSocket connection for real-time voice communication. This class handles bidirectional communication, allowing you to send audio input and receive audio output, text transcriptions, and other events in real-time.

Session Configuration

The service uses session configuration to control various aspects of voice interaction:

- **Turn Detection:** Configure how the service detects when users start and stop speaking
- **Audio Processing:** Enable noise suppression and echo cancellation
- **Voice Selection:** Choose from standard Azure voices, high-definition voices, or custom voices
- **Model Selection:** Select the AI model (GPT-4o, GPT-4o-mini, Phi variants) that best fits your needs

Models and Capabilities

The VoiceLive API supports multiple AI models with different capabilities:

 Expand table

Model	Description	Use Case
gpt-4o-realtime-preview	GPT-4o with real-time audio processing	High-quality conversational AI
gpt-4o-mini-realtime-preview	Lightweight GPT-4o variant	Fast, efficient interactions
phi4-mm-realtime	Phi model with multimodal support	Cost-effective voice applications

Conversational Enhancements

The VoiceLive API provides Azure-specific enhancements:

- **Azure Semantic VAD:** Advanced voice activity detection that removes filler words
- **Noise Suppression:** Reduces environmental background noise
- **Echo Cancellation:** Removes echo from the model's own voice
- **End-of-Turn Detection:** Allows natural pauses without premature interruption

Authenticating with Azure Active Directory

The VoiceLive service relies on Azure Active Directory to authenticate requests to its APIs. The [@azure/identity](#) package provides a variety of credential types that your application can use to do this. The [README for @azure/identity](#) provides more details and samples to get you started.

To interact with the Azure VoiceLive service, you need to create an instance of the `VoiceLiveClient` class, a **service endpoint** and a credential object. The examples shown in this document use a credential object named `DefaultAzureCredential`, which is appropriate for most scenarios, including local development and production environments. We recommend using a [managed identity](#) for authentication in production environments.

You can find more information on different ways of authenticating and their corresponding credential types in the [Azure Identity documentation](#).

Here's a quick example. First, import `DefaultAzureCredential` and `VoiceLiveClient`:

```
ts
```

```
import { DefaultAzureCredential } from "@azure/identity";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const credential = new DefaultAzureCredential();

// Build the URL to reach your AI Foundry resource
const endpoint = "https://your-resource.cognitiveservices.azure.com";

// Create the VoiceLive client
const client = new VoiceLiveClient(endpoint, credential);
```

Authentication with API Key

For development scenarios, you can also authenticate using an API key:

```
ts
```

```
import { AzureKeyCredential } from "@azure/core-auth";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const endpoint = "https://your-resource.cognitiveservices.azure.com";
const credential = new AzureKeyCredential("your-api-key");

const client = new VoiceLiveClient(endpoint, credential);
```

Examples

The following sections provide code snippets that cover some of the common tasks using Azure VoiceLive. The scenarios covered here consist of:

- [Creating a basic voice assistant](#)
- [Configuring session options](#)
- [Handling real-time events](#)

- Implementing function calling

Creating a basic voice assistant

This example shows how to create a simple voice assistant that can handle speech-to-speech interactions:

```
ts

import { DefaultAzureCredential } from "@azure/identity";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const credential = new DefaultAzureCredential();
const endpoint = "https://your-resource.cognitiveservices.azure.com";

// Create the client
const client = new VoiceLiveClient(endpoint, credential);

// Create and connect a session
const session = await client.startSession("gpt-4o-mini-realtime-preview");

// Configure session for voice conversation
await session.updateSession({
  modalities: ["text", "audio"],
  instructions: "You are a helpful AI assistant. Respond naturally and conversationally.",
  voice: {
    type: "azure-standard",
    name: "en-US-AvaNeural",
  },
  turnDetection: {
    type: "server_vad",
    threshold: 0.5,
    prefixPaddingMs: 300,
    silenceDurationMs: 500,
  },
  inputAudioFormat: "pcm16",
  outputAudioFormat: "pcm16",
});
```

Configuring session options

You can customize various aspects of the voice interaction:

```
ts

import { DefaultAzureCredential } from "@azure/identity";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const credential = new DefaultAzureCredential();
```

```

const endpoint = "https://your-resource.cognitiveservices.azure.com";
const credential = new VoiceLiveClient(endpoint, credential);
const session = await client.startSession("gpt-4o-realtime-preview");

// Advanced session configuration
await session.updateSession({
  modalities: ["audio", "text"],
  instructions: "You are a customer service representative. Be helpful and
professional.",
  voice: {
    type: "azure-custom",
    name: "your-custom-voice-name",
    endpointId: "your-custom-voice-endpoint",
  },
  turnDetection: {
    type: "server_vad",
    threshold: 0.6,
    prefixPaddingMs: 200,
    silenceDurationMs: 300,
  },
  inputAudioFormat: "pcm16",
  outputAudioFormat: "pcm16",
});

```

Handling real-time events

The VoiceLive client provides event-driven communication for real-time interactions:

ts

```

import { DefaultAzureCredential } from "@azure/identity";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const credential = new DefaultAzureCredential();
const endpoint = "https://your-resource.cognitiveservices.azure.com";
const client = new VoiceLiveClient(endpoint, credential);
const session = await client.startSession("gpt-4o-mini-realtime-preview");

// Set up event handlers using subscription pattern
const subscription = session.subscribe({
  onResponseAudioDelta: async (event, context) => {
    // Handle incoming audio chunks
    const audioData = event.delta;
    // Play audio using Web Audio API or other audio system
    playAudioChunk(audioData);
  },
  onResponseTextDelta: async (event, context) => {
    // Handle incoming text deltas
    console.log("Assistant:", event.delta);
  },
});

```

```

onInputAudioTranscriptionCompleted: async (event, context) => {
  // Handle user speech transcription
  console.log("User said:", event.transcript);
},
});

// Send audio data from microphone
function sendAudioChunk(audioBuffer: ArrayBuffer) {
  session.sendAudio(audioBuffer);
}

```

Implementing function calling

Enable your voice assistant to call external functions and tools:

```

ts

import { DefaultAzureCredential } from "@azure/identity";
import { VoiceLiveClient } from "@azure/ai-voicelive";

const credential = new DefaultAzureCredential();
const endpoint = "https://your-resource.cognitiveservices.azure.com";
const client = new VoiceLiveClient(endpoint, credential);
const session = await client.startSession("gpt-4o-mini-realtime-preview");

// Define available functions
const tools = [
{
  type: "function",
  name: "get_weather",
  description: "Get current weather for a location",
  parameters: {
    type: "object",
    properties: {
      location: {
        type: "string",
        description: "The city and state or country",
      },
    },
    required: ["location"],
  },
},
];
;

// Configure session with tools
await session.updateSession({
  modalities: ["audio", "text"],
  instructions:
    "You can help users with weather information. Use the get_weather function when needed.",
  tools: tools,
  toolChoice: "auto",
});

```

```

});
```

```

// Handle function calls
const subscription = session.subscribe({
  onResponseFunctionCallArgumentsDone: async (event, context) => {
    if (event.name === "get_weather") {
      const args = JSON.parse(event.arguments);
      const weatherData = await getWeatherData(args.location);

      // Send function result back
      await session.addConversationItem({
        type: "function_call_output",
        callId: event.callId,
        output: JSON.stringify(weatherData),
      });
    }
  },
});
```

Troubleshooting

Common errors and exceptions

Authentication Errors: If you receive authentication errors, verify that:

- Your Azure AI Foundry resource is correctly configured
- Your API key or credential has the necessary permissions
- The endpoint URL is correct and accessible

WebSocket Connection Issues: VoiceLive uses WebSocket connections. Ensure that:

- Your network allows WebSocket connections
- Firewall rules permit connections to *.cognitiveservices.azure.com
- Browser policies allow WebSocket and microphone access (for browser usage)

Audio Issues: For audio-related problems:

- Verify microphone permissions in the browser
- Check that audio formats (PCM16, PCM24) are supported
- Ensure proper audio context setup for playback

Logging

Enabling logging may help uncover useful information about failures. In order to see a log of WebSocket messages and responses, set the `AZURE_LOG_LEVEL` environment variable to `info`. Alternatively, logging can be enabled at runtime by calling `setLogLevel` in the `@azure/logger`:

```
ts
```

```
import { setLogLevel } from "@azure/logger";  
  
setLogLevel("info");
```

For more detailed instructions on how to enable logs, you can look at the [@azure/logger package docs](#).

Next steps

You can find more code samples through the following links:

- [VoiceLive Samples \(JavaScript/TypeScript\)](#)
- [VoiceLive Test Cases](#)

Contributing

If you'd like to contribute to this library, please read the [contributing guide](#) to learn more about how to build and test the code.

Last updated on 11/18/2025

What is keyword recognition?

08/07/2025

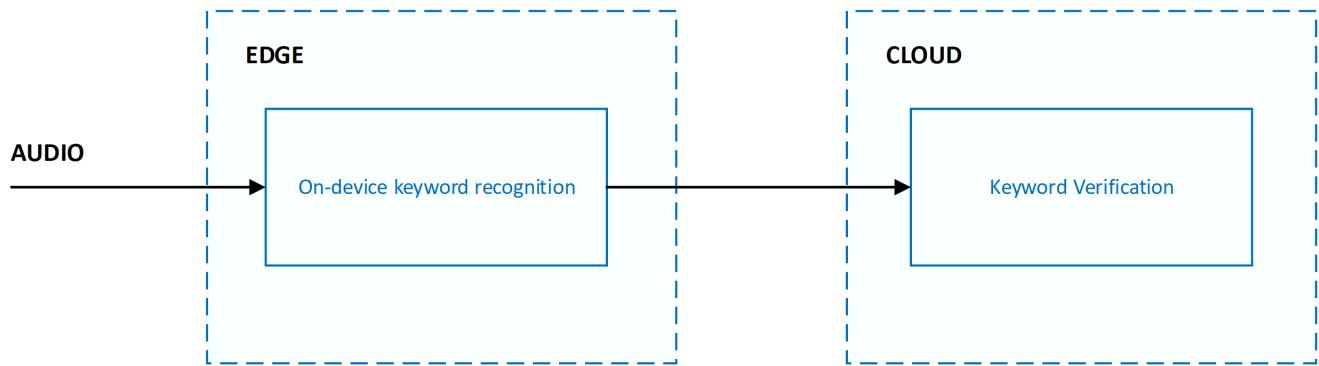
Keyword recognition detects a word or short phrase within a stream of audio. This technique is also referred to as keyword spotting.

The most common use case of keyword recognition is voice activation of virtual assistants. For example, "Hey Cortana" is the keyword for the Cortana assistant. Upon recognition of the keyword, a scenario-specific action is carried out. For virtual assistant scenarios, a common resulting action is speech recognition of audio that follows the keyword.

Generally, virtual assistants are always listening. Keyword recognition acts as a privacy boundary for the user. A keyword requirement acts as a gate that prevents unrelated user audio from crossing the local device to the cloud.

To balance accuracy, latency, and computational complexity, keyword recognition is implemented as a multistage system. For all stages beyond the first, audio is only processed if the stage prior to it recognizes the keyword of interest.

The current system is designed with multiple stages that span the edge and cloud:



Accuracy of keyword recognition is measured via the following metrics:

- **Correct accept rate:** Measures the system's ability to recognize the keyword spoken by a user. The correct accept rate is also known as the true positive rate.
- **False accept rate:** Measures the system's ability to filter out audio that isn't the keyword spoken by a user. The false accept rate is also known as the false positive rate.

The goal is to maximize the correct accept rate while minimizing the false accept rate. The current system is designed to detect a keyword or phrase preceded by a short amount of silence. Detecting a keyword in the middle of a sentence or utterance isn't supported.

Custom keyword for on-device models

With the [Custom Keyword portal on Speech Studio](#), you can generate keyword recognition models that execute at the edge by specifying any word or short phrase. You can further personalize your keyword model by choosing the right pronunciations.

Pricing

There's no cost to use custom keyword to generate models, including both Basic and Advanced models. There's also no cost to run models on-device with the Speech SDK when used with other Speech service features such as speech to text.

Types of models

You can use custom keyword to generate two types of on-device models for any keyword.

 Expand table

Model type	Description
Basic	Best suited for demo or rapid prototyping purposes. Models are generated with a common base model and can take up to 15 minutes to be ready. Models might not have optimal accuracy characteristics.
Advanced	Best suited for product integration purposes. Models are generated with adaptation of a common base model by using simulated training data to improve accuracy characteristics. It can take up to 48 hours for models to be ready.

Note

You can view a list of regions that support the **Advanced** model type in the [keyword recognition region support](#) documentation.

Neither model type requires you to upload training data. Custom keyword fully handles data generation and model training.

Pronunciations

When you create a new model, custom keyword automatically generates possible pronunciations of the provided keyword. You can listen to each pronunciation and choose all variations that closely represent the way you expect users to say the keyword. All other pronunciations shouldn't be selected.

It's important to be deliberate about the pronunciations you select to ensure the best accuracy characteristics. For example, if you choose more pronunciations than you need, you might get higher false accept rates. If you choose too few pronunciations, where not all expected variations are covered, you might get lower correct accept rates.

Test models

After custom keyword generates on-device models, the models can be tested directly on the portal. You can use the portal to speak directly into your browser and get keyword recognition results.

Keyword verification

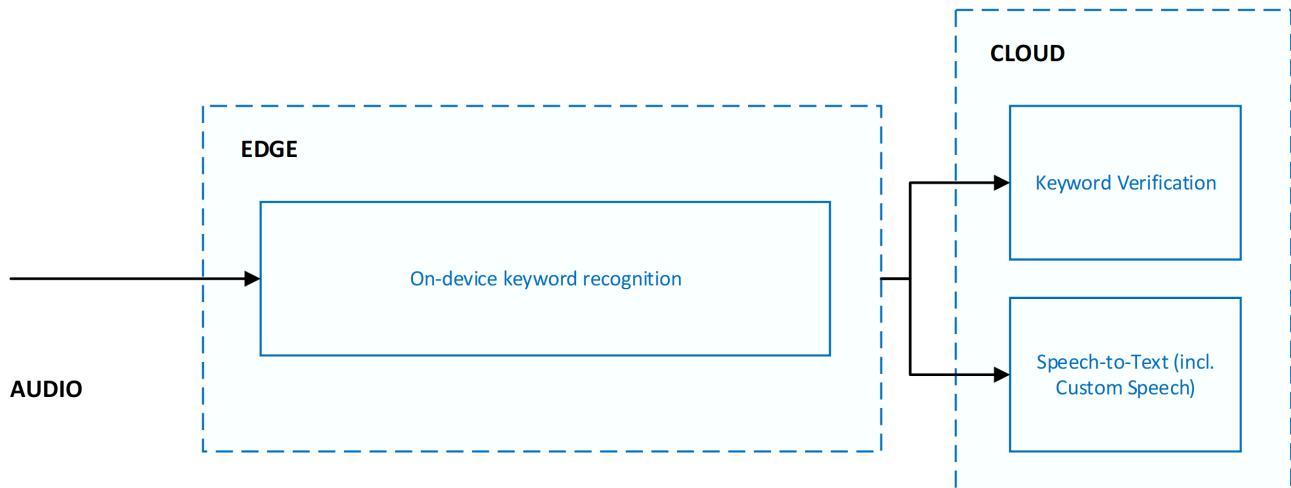
Keyword verification is a cloud service that reduces the effect of false accepts from on-device models with robust models running on Azure. Tuning or training isn't required for keyword verification to work with your keyword. Incremental model updates are continually deployed to the service to improve accuracy and latency and are transparent to client applications.

Pricing

Keyword verification is always used in combination with speech to text. There's no cost to use keyword verification beyond the cost of speech to text.

Keyword verification and speech to text

When keyword verification is used, it's always in combination with speech to text. Both services run in parallel, which means audio is sent to both services for simultaneous processing.



Running keyword verification and speech to text in parallel yields the following benefits:

- **No other latency on speech to text results:** Parallel execution means that keyword verification adds no latency. The client receives speech to text results as quickly. If keyword verification determines the keyword wasn't present in the audio, speech to text processing is terminated. This action protects against unnecessary speech to text processing. Network and cloud model processing increases the user-perceived latency of voice activation. For more information, see [Recommendations and guidelines](#).
- **Forced keyword prefix in speech to text results:** Speech to text processing ensures that the results sent to the client are prefixed with the keyword. This behavior allows for increased accuracy in the speech to text results for speech that follows the keyword.
- **Increased speech to text timeout:** Because of the expected presence of the keyword at the beginning of audio, speech to text allows for a longer pause of up to five seconds after the keyword before it determines the end of speech and terminates speech to text processing. This behavior ensures that the user experience is correctly handled for staged commands (`<keyword> <pause> <command>`) and chained commands (`<keyword> <command>`).

Keyword verification responses and latency considerations

For each request to the service, keyword verification returns one of two responses: accepted or rejected. The processing latency varies depending on the length of the keyword and the length of the audio segment expected to contain the keyword. Processing latency doesn't include network cost between the client and Speech services.

[] [Expand table](#)

Keyword verification response	Description
Accepted	Indicates the service believed that the keyword was present in the audio stream provided as part of the request.
Rejected	Indicates the service believed that the keyword wasn't present in the audio stream provided as part of the request.

Rejected cases often yield higher latencies as the service processes more audio than accepted cases. By default, keyword verification processes a maximum of two seconds of audio to search for the keyword. If the keyword isn't found in two seconds, the service times out and signals a rejected response to the client.

Use keyword verification with on-device models from custom keyword

The Speech SDK enables seamless use of on-device models generated by using custom keyword with keyword verification and speech to text. It transparently handles:

- Audio gating to keyword verification and speech recognition based on the outcome of an on-device model.
- Communicating the keyword to keyword verification.
- Communicating any more metadata to the cloud for orchestrating the end-to-end scenario.

You don't need to explicitly specify any configuration parameters. All necessary information is automatically extracted from the on-device model generated by custom keyword.

Speech SDK integration and scenarios

The Speech SDK enables easy use of personalized on-device keyword recognition models generated with custom keyword and keyword verification. To ensure that your product needs can be met, the SDK supports the following two scenarios:

The offline keyword recognition scenario is best suited for products without network connectivity that use a customized on-device keyword model from custom keyword.

- [C# on Windows UWP sample ↗](#)
- [Java on Android sample ↗](#)

Related content

- [Read the quickstart to generate on-device keyword recognition models using custom keyword](#)
- [Learn more about voice agents](#)

Quickstart: Create a custom keyword

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this quickstart, you learn the basics of working with custom keywords. A keyword is a word or short phrase, which allows your product to be voice activated. You create keyword models in Speech Studio. Then export a model file that you use with the Speech SDK in your applications.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#) ↗.
- ✓ [Create an AI Foundry resource for Speech](#) ↗ 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.

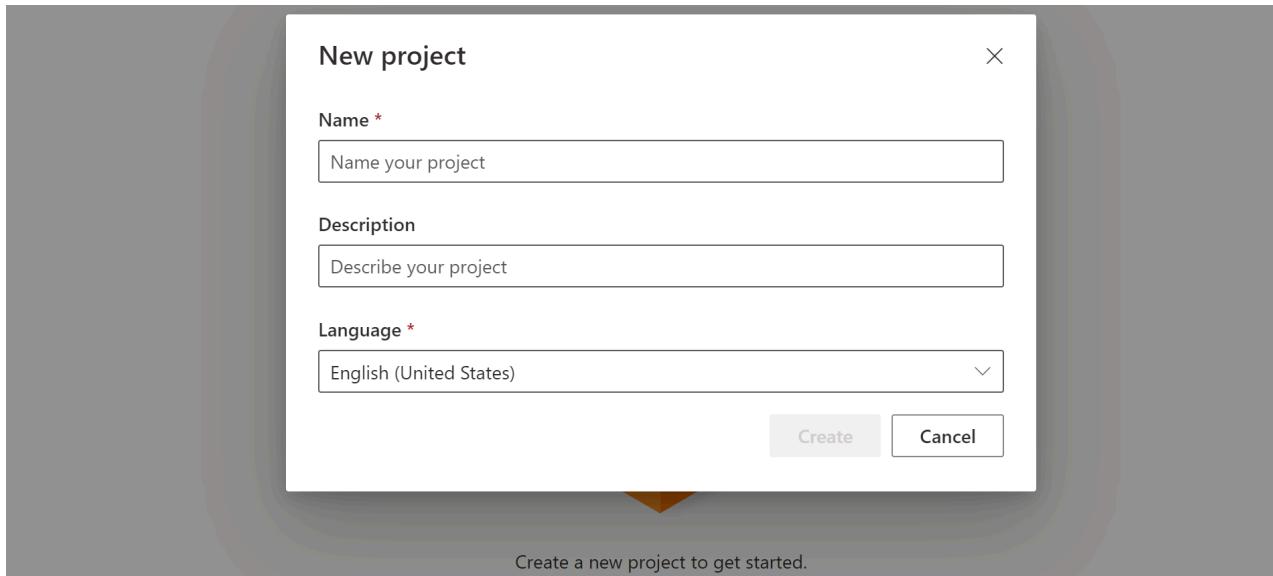
Create a keyword in Speech Studio

Before you can use a custom keyword, you need to create a keyword using the [Custom Keyword](#) ↗ page on [Speech Studio](#) ↗. After you provide a keyword, it produces a `.table` file that you can use with the Speech SDK.

Important

Custom keyword models, and the resulting `.table` files, can **only** be created in Speech Studio. You cannot create custom keywords from the SDK or with REST calls.

1. Go to the [Speech Studio](#) ↗ and **Sign in**. If you don't have a speech subscription, go to [Create Speech Services](#) ↗.
2. On the [Custom Keyword](#) ↗ page, select **Create a new project**.
3. Enter a **Name**, **Description**, and **Language** for your custom keyword project. You can only choose one language per project, and support is currently limited to English (United States) and Chinese (Mandarin, Simplified).



4. Select your project's name from the list.

Speech Studio > Custom Keyword

Select the project you want to work with

Name	Description	Language	Training	Created ↓
My custom keyword project	This is an example for the documentation	English (United States)	1	11/11/2021 6:46 AM

5. To create a custom keyword for your virtual assistant, select **Create a new model**.

6. Enter a **Name** for the model, **Description**, and **Keyword** of your choice, then select **Next**. See the [guidelines](#) on choosing an effective keyword.

Speech Studio > Custom Keyword > My custom keyword project > Models

Create a custom keyword

New model

Name and keyword
 Select pronunciations
 Choose model type

Name and keyword

Consider the following guidelines when choosing your keyword:
The word should be 4 to 7 syllables. For example, "Hey Computer" is a great choice.
Do not choose a common word. For example, "eat" and "go". Do not use special characters, symbols, or digits.

For a full set of guidelines, please visit the [documentation](#).

Name *

Description

Keyword *

Next **Cancel**

7. The portal creates candidate pronunciations for your keyword. Listen to each candidate by selecting the play buttons and remove the checks next to any pronunciations that are incorrect. Select all pronunciations that correspond to how you expect your users to say the keyword and then select **Next** to begin generating the keyword model.



8. Select a model type, then select **Create**. You can view a list of regions that support the **Advanced** model type in the [Keyword recognition region support](#) documentation.
9. Because of high demand, training the basic model might take several hours. Training the advanced model could take up to a day to finish. The status changes from **Processing** to **Succeeded** when the training is complete.

Name	Model type	Description	Keyword	Created	Status
My custom keyword model	Basic	This is an example for the documentation	Hey Computer	11/11/2021 11:50 AM	Succeeded

10. From the collapsible menu on the left, select **Tune** for options to tune and download your model. The downloaded file is a **.zip** archive. Extract the archive, and you see a file with the **.table** extension. You use the **.table** file with the SDK, so make sure to note its path.

Choose a model	Configuration
My custom keyword model... ▾	Default ▾

Download

Use a keyword model with the Speech SDK

First, load your keyword model file using the `FromFile()` static function, which returns a `KeywordRecognitionModel`. Use the path to the **.table** file you downloaded from Speech Studio. Additionally, you create an `AudioConfig` using the default microphone, then instantiate a new `KeywordRecognizer` using the audio configuration.

C#

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
```

```
var keywordModel =  
    KeywordRecognitionModel.FromFile("your/path/to/Activate_device.table");  
using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();  
using var keywordRecognizer = new KeywordRecognizer(audioConfig);
```

ⓘ Important

If you prefer testing a keyword model directly with audio samples via the `AudioConfig.fromStreamInput()` method, make sure you use samples that have at least 1.5 seconds of silence before the first keyword. This is to provide an adequate time for the Keyword recognition engine to initialize and to get to the listening state prior to detecting the first keyword.

Next, running keyword recognition is done with one call to `RecognizeOnceAsync()` by passing your model object. This method starts a keyword recognition session that lasts until the keyword is recognized. Thus, you generally use this design pattern in multi-threaded applications, or in use cases where you might be waiting for a wake-word indefinitely.

C#

```
KeywordRecognitionResult result = await  
    keywordRecognizer.RecognizeOnceAsync(keywordModel);
```

! Note

The example shown here uses local keyword recognition, since it does not require a `SpeechConfig` object for authentication context, and does not contact the back-end.

Continuous recognition

Other classes in the Speech SDK support continuous recognition (for both speech and intent recognition) with keyword recognition. The SDK allows you to use the same code you would normally use for continuous recognition, with the ability to reference a `.table` file for your keyword model.

For speech to text, follow the same design pattern shown in the [recognize speech guide](#) to set up continuous recognition. Then, replace the call to

```
recognizer.StartContinuousRecognitionAsync()
```

```
with recognizer.StartKeywordRecognitionAsync(
```

```
KeywordRecognitionModel), and pass your
```

```
KeywordRecognitionModel object. To stop continuous recognition with keyword recognition, use
```

```
recognizer.StopKeywordRecognitionAsync() instead of  
recognizer.StopContinuousRecognitionAsync().
```

Intent recognition uses an identical pattern with the [StartKeywordRecognitionAsync](#) and [StopKeywordRecognitionAsync](#) functions.

Next steps

[Get the Speech SDK](#)

Recommendations and guidelines for keyword recognition

08/07/2025

This article outlines how to choose your keyword optimize its accuracy characteristics and how to design your user experiences with keyword verification.

Choosing an effective keyword

Creating an effective keyword is vital to ensuring your product responds consistently and accurately. Consider the following guidelines when you choose a keyword.

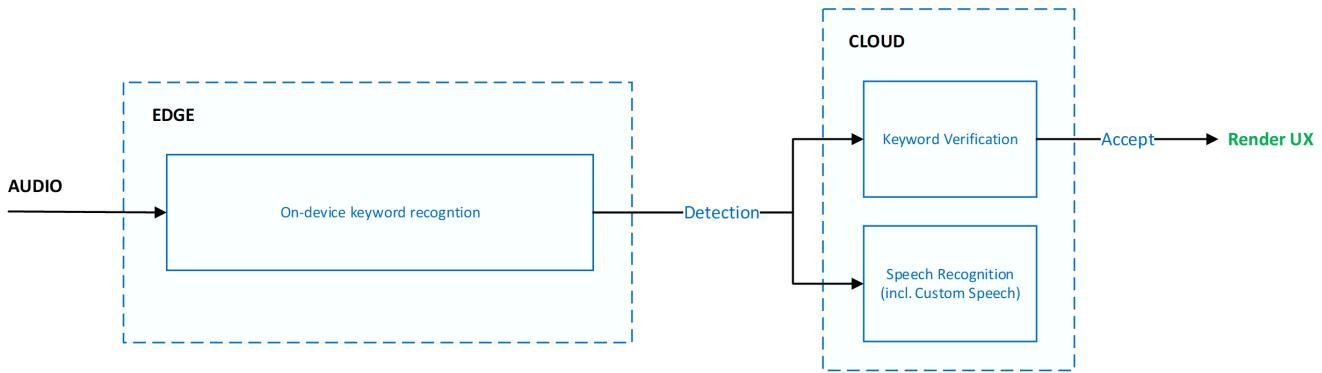
Note

The examples in this section are in English but the guidelines apply to all languages supported by custom keyword. For a list of all supported languages, see [Language support](#).

- It should take no longer than two seconds to say.
- Words of 4 to 7 syllables work best. For example, "Hey, Computer" is a good keyword. Just "Hey" is a poor one.
- Keywords should follow common pronunciation rules specific to the native language of your end-users.
- A unique or even a made-up word that follows common pronunciation rules might reduce false positives. For example, "computerama" might be a good keyword.
- Don't choose a common word. For example, "eat" and "go" are words that people say frequently in ordinary conversation. They might lead to higher than desired false accept rates for your product.
- Avoid using a keyword that might have alternative pronunciations. Users would have to know the "right" pronunciation to get their product to voice activate. For example, "509" can be pronounced "five zero nine," "five oh nine," or "five hundred and nine." "R.E.I." can be pronounced "r-e-i" or "ray." "Live" can be pronounced "/līv/" or "/liv/".
- Don't use special characters, symbols, or digits. For example, "Go#" and "20 + cats" could be problematic keywords. However, "go sharp" or "twenty plus cats" might work. You can still use the symbols in your branding and use marketing and documentation to reinforce the proper pronunciation.

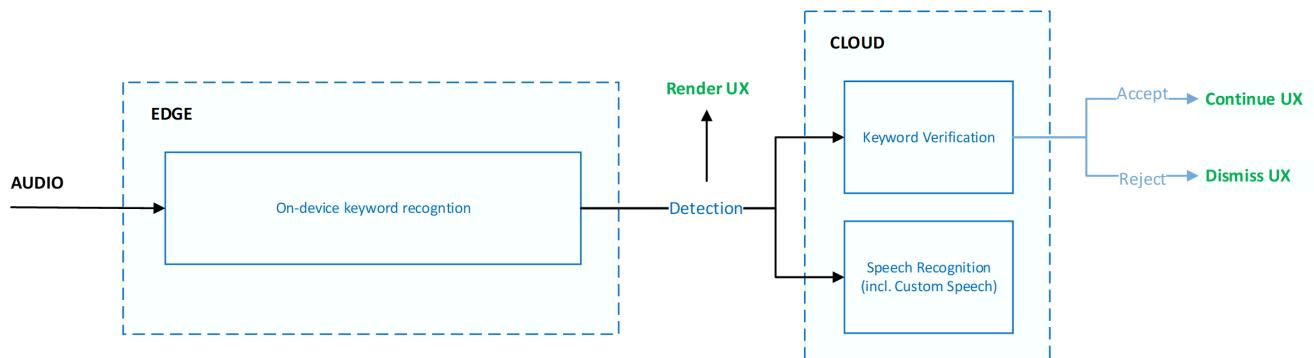
User experience recommendations with keyword verification

With a multi-stage keyword recognition scenario where [keyword verification](#) is used, applications can choose when the end-user is notified of a keyword detection. The recommendation for rendering any visual or audible indicator is to rely upon responses from the keyword verification service:



This ensures the optimal experience in terms of accuracy to minimize the user-perceived effect of false accepts but incurs extra latency.

For applications that require latency optimization, applications can provide light and unobtrusive indicators to the end-user based on the on-device keyword recognition. For example, lighting an LED pattern or pulsing an icon. The indicators can continue to exist if keyword verification accepts a keyword, or can be dismissed if the keyword is rejected.



Next steps

- [Get the Speech SDK](#).
- [Learn more about voice agents](#)

Captioning with speech to text

08/07/2025

In this guide, you learn how to create captions with speech to text. Captioning is the process of converting the audio content of a television broadcast, webcast, film, video, live event, or other production into text, and then displaying the text on a screen, monitor, or other visual display system.

Concepts include how to synchronize captions with your input audio, apply profanity filters, get partial results, apply customizations, and identify spoken languages for multilingual scenarios. This guide covers captioning for speech, but doesn't include speaker ID or sound effects such as bells ringing.

Here are some common captioning scenarios:

- Online courses and instructional videos
- Sporting events
- Voice and video calls

The following are aspects to consider when using captioning:

- Let your audience know that captions are generated by an automated service.
- Center captions horizontally on the screen, in a large and prominent font.
- Consider whether to use partial results, when to start displaying captions, and how many words to show at a time.
- Learn about captioning protocols such as [SMPTE-TT](#).
- Consider output formats such as SRT (SubRip Text) and WebVTT (Web Video Text Tracks). These can be loaded onto most video players such as VLC, automatically adding the captions on to your video.

💡 Tip

Try the [Speech Studio](#) and choose a sample video clip to see real-time or offline processed captioning results.

Try the [Azure AI Video Indexer](#) as a demonstration of how you can get captions for videos that you upload.

Captioning can accompany real-time or prerecorded speech. Whether you're showing captions in real-time or with a recording, you can use the [Speech SDK](#) or [Speech CLI](#) to recognize speech and get transcriptions. You can also use the [Batch transcription API](#) for pre-recorded video.

Caption output format

The Speech service supports output formats such as SRT (SubRip Text) and WebVTT (Web Video Text Tracks). These can be loaded onto most video players such as VLC, automatically adding the captions on to your video.

Tip

The Speech service provides [profanity filter](#) options. You can specify whether to mask, remove, or show profanity.

The [SRT](#) (SubRip Text) timespan output format is `hh:mm:ss,fff`.

```
srt

1
00:00:00,180 --> 00:00:03,230
Welcome to applied Mathematics course 201.
```

The [WebVTT](#) (Web Video Text Tracks) timespan output format is `hh:mm:ss.fff`.

```
WEBVTT

00:00:00.180 --> 00:00:03.230
Welcome to applied Mathematics course 201.
{
  "ResultId": "8e89437b4b9349088a933f8db4ccc263",
  "Duration": "00:00:03.050000"
}
```

Input audio to the Speech service

For real-time captioning, use a microphone or audio input stream instead of file input. For examples of how to recognize speech from a microphone, see the [Speech to text quickstart](#) and [How to recognize speech](#) documentation. For more information about streaming, see [How to use the audio input stream](#).

For captioning of a prerecording, send file input to the Speech service. For more information, see [How to use compressed input audio](#).

Caption and speech synchronization

You want to synchronize captions with the audio track, whether it's in real-time or with a prerecording.

The Speech service returns the offset and duration of the recognized speech.

- **Offset:** The offset into the audio stream being recognized, expressed as duration. Offset is measured in ticks, starting from `0` (zero) tick, associated with the first audio byte processed by the SDK. For example, the offset begins when you start recognition, since that's when the SDK starts processing the audio stream. One tick represents one hundred nanoseconds or one ten-millionth of a second.
- **Duration:** Duration of the utterance that is being recognized. The duration in ticks doesn't include trailing or leading silence.

For more information, see [Get speech recognition results](#).

Get partial results

Consider when to start displaying captions, and how many words to show at a time. Speech recognition results are subject to change while an utterance is still being recognized. Partial results are returned with each `Recognizing` event. As each word is processed, the Speech service re-evaluates an utterance in the new context and again returns the best result. The new result isn't guaranteed to be the same as the previous result. The complete and final transcription of an utterance is returned with the `Recognized` event.

 Note

Punctuation of partial results is not available.

For captioning of prerecorded speech or wherever latency isn't a concern, you could wait for the complete transcription of each utterance before displaying any words. Given the final offset and duration of each word in an utterance, you know when to show subsequent words at pace with the soundtrack.

Real-time captioning presents tradeoffs with respect to latency versus accuracy. You could show the text from each `Recognizing` event as soon as possible. However, if you can accept some latency, you can improve the accuracy of the caption by displaying the text from the `Recognized` event. There's also some middle ground, which is referred to as "stable partial results".

You can request that the Speech service return fewer `Recognizing` events that are more accurate. This is done by setting the `SpeechServiceResponse_StablePartialResultThreshold` property to a value between `0` and `2147483647`. The value that you set is the number of times a word has to be recognized before the Speech service returns a `Recognizing` event. For example, if you set the `SpeechServiceResponse_StablePartialResultThreshold` property value to `5`, the Speech service affirms recognition of a word at least five times before returning the partial results to you with a `Recognizing` event.

C#

```
speechConfig SetProperty(PropertyId.SpeechServiceResponse_StablePartialResultThreshold, 5);
```

Requesting more stable partial results reduce the "flickering" or changing text, but it can increase latency as you wait for higher confidence results.

Stable partial threshold example

In the following recognition sequence without setting a stable partial threshold, "math" is recognized as a word, but the final text is "mathematics". At another point, "course 2" is recognized, but the final text is "course 201".

Console

```
RECOGNIZING: Text=welcome to
RECOGNIZING: Text=welcome to applied math
RECOGNIZING: Text=welcome to applied mathematics
RECOGNIZING: Text=welcome to applied mathematics course 2
RECOGNIZING: Text=welcome to applied mathematics course 201
RECOGNIZED: Text=Welcome to applied Mathematics course 201.
```

In the previous example, the transcriptions were additive and no text was retracted. But at other times you might find that the partial results were inaccurate. In either case, the unstable partial results can be perceived as "flickering" when displayed.

For this example, if the stable partial result threshold is set to `5`, no words are altered or backtracked.

Console

```
RECOGNIZING: Text=welcome to
RECOGNIZING: Text=welcome to applied
```

RECOGNIZING: Text=welcome to applied mathematics

RECOGNIZED: Text=Welcome to applied Mathematics course 201.

Language identification

If the language in the audio could change, use continuous [language identification](#). Language identification is used to identify languages spoken in audio when compared against a list of [supported languages](#). You provide up to 10 candidate languages, at least one of which is expected in the audio. The Speech service returns the most likely language in the audio.

Customizations to improve accuracy

A [phrase list](#) is a list of words or phrases that you provide right before starting speech recognition. Adding a phrase to a phrase list increases its importance, thus making it more likely to be recognized.

Examples of phrases include:

- Names
- Geographical locations
- Homonyms
- Words or acronyms unique to your industry or organization

There are some situations where [training a custom model](#) is likely the best option to improve accuracy. For example, if you're captioning orthodontic lectures, you might want to train a custom model with the corresponding domain data.

Next steps

- [Captioning quickstart](#)
- [Get speech recognition results](#)

Quickstart: Create captions with speech to text

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) | [Additional samples on GitHub](#)

In this quickstart, you run a console app to create [captions](#) with speech to text.

💡 Tip

Try out the [Speech Studio](#) and choose a sample video clip to see real-time or offline processed captioning results.

💡 Tip

Try out the [Azure AI Speech Toolkit](#) to easily build and run captioning samples on Visual Studio Code.

Prerequisites

- ✓ An Azure subscription. You can [create one for free](#).
- ✓ [Create an AI Foundry resource for Speech](#) 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.

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.

You must also install [GStreamer](#) for compressed input audio.

Set environment variables

You need to authenticate your application to access Azure AI services. This article shows you how to use environment variables to store your credentials. You can then access the

environment variables from your code to authenticate your application. For production, use a more secure way to store and access your credentials.

Important

We recommend Microsoft Entra ID authentication with [managed identities for Azure resources](#) to avoid storing credentials with your applications that run in the cloud.

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If using API keys, store them securely in Azure Key Vault, rotate the keys regularly, and restrict access to Azure Key Vault using role based access control and network access restrictions. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

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

- To set the `SPEECH_KEY` environment variable, replace *your-key* with one of the keys for your resource.
- To set the `SPEECH_REGION` environment variable, replace *your-region* with one of the regions for your resource.

Windows

Console

```
setx SPEECH_KEY your-key  
setx SPEECH_REGION your-region
```

Note

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

After you add the environment variables, you might need to restart any programs that need to read the environment variables, including the console window. For example, if you're using Visual Studio as your editor, restart Visual Studio before you run the example.

Create captions from speech

Follow these steps to build and run the captioning quickstart code example.

1. Copy the [scenarios/csharp/dotnetcore/captioning/](#) sample files from GitHub. If you have [Git installed](#), open a command prompt and run the `git clone` command to download the Speech SDK samples repository.

```
.NET CLI
```

```
git clone https://github.com/Azure-Samples/cognitive-services-speech-sdk.git
```

2. Open a command prompt and change to the project directory.

```
.NET CLI
```

```
cd <your-local-path>/scenarios/csharp/dotnetcore/captioning/captioning/
```

3. Build the project with the .NET CLI.

```
.NET CLI
```

```
dotnet build
```

4. Run the application with your preferred command line arguments. See [usage and arguments](#) for the available options. Here is an example:

```
.NET CLI
```

```
dotnet run --input caption.this.mp4 --format any --output caption.output.txt  
--srt --realTime --threshold 5 --delay 0 --profanity mask --phrases  
"Contoso;Jessie;Rehaan"
```

ⓘ Important

Make sure that the paths specified by `--input` and `--output` are valid. Otherwise you must change the paths.

Make sure that you set the `SPEECH_KEY` and `SPEECH_REGION` environment variables as described [above](#). Otherwise use the `--key` and `--region` arguments.

Check results

When you use the `realTime` option in the example above, the partial results from `Recognizing` events are included in the output. In this example, only the final `Recognized` event includes the commas. Commas aren't the only differences between `Recognizing` and `Recognized` events. For more information, see [Get partial results](#).

```
srt

1
00:00:00,170 --> 00:00:00,380
The

2
00:00:00,380 --> 00:00:01,770
The rainbow

3
00:00:01,770 --> 00:00:02,560
The rainbow has seven

4
00:00:02,560 --> 00:00:03,820
The rainbow has seven colors

5
00:00:03,820 --> 00:00:05,050
The rainbow has seven colors red

6
00:00:05,050 --> 00:00:05,850
The rainbow has seven colors red
orange

7
00:00:05,850 --> 00:00:06,440
The rainbow has seven colors red
orange yellow

8
00:00:06,440 --> 00:00:06,730
The rainbow has seven colors red
orange yellow green

9
00:00:06,730 --> 00:00:07,160
orange, yellow, green, blue,
indigo and Violet.
```

When you use the `--offline` option, the results are stable from the final `Recognized` event.

Partial results aren't included in the output:

```
srt

1
00:00:00,170 --> 00:00:05,540
The rainbow has seven colors, red,
orange, yellow, green, blue,

2
00:00:05,540 --> 00:00:07,160
indigo and Violet.
```

The [SRT](#) (SubRip Text) timespan output format is `hh:mm:ss,fff`. For more information, see [Caption output format](#).

Usage and arguments

Usage: `captioning --input <input file>`

Connection options include:

- `--key`: Your AI Foundry resource key. Overrides the `SPEECH_KEY` environment variable. You must set the environment variable (recommended) or use the `--key` option.
- `--region REGION`: Your AI Foundry resource region. Overrides the `SPEECH_REGION` environment variable. You must set the environment variable (recommended) or use the `--region` option. Examples: `westus`, `northeurope`

Important

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If you use an API key, store it securely in Azure Key Vault. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

Input options include:

- `--input FILE`: Input audio from file. The default input is the microphone.
- `--format FORMAT`: Use compressed audio format. Valid only with `--file`. Valid values are `alaw`, `any`, `flac`, `mp3`, `mulaw`, and `ogg_opus`. The default value is `any`. To use a `wav` file,

don't specify the format. This option is not available with the JavaScript captioning sample. For compressed audio files such as MP4, install GStreamer and see [How to use compressed input audio](#).

Language options include:

- `--language LANG`: Specify a language using one of the corresponding [supported locales](#). This is used when breaking captions into lines. Default value is `en-US`.

Recognition options include:

- `--offline`: Output offline results. Overrides `--realTime`. Default output mode is offline.
- `--realTime`: Output real-time results.

Real-time output includes `Recognizing` event results. The default offline output is `Recognized` event results only. These are always written to the console, never to an output file. The `--quiet` option overrides this. For more information, see [Get speech recognition results](#).

Accuracy options include:

- `--phrases PHRASE1;PHRASE2`: You can specify a list of phrases to be recognized, such as `Contoso; Jessie; Rehaan`. For more information, see [Improve recognition with phrase list](#).

Output options include:

- `--help`: Show this help and stop
- `--output FILE`: Output captions to the specified `file`. This flag is required.
- `--srt`: Output captions in SRT (SubRip Text) format. The default format is WebVTT (Web Video Text Tracks). For more information about SRT and WebVTT caption file formats, see [Caption output format](#).
- `--maxLengthLength LENGTH`: Set the maximum number of characters per line for a caption to `LENGTH`. Minimum is 20. Default is 37 (30 for Chinese).
- `--lines LINES`: Set the number of lines for a caption to `LINES`. Minimum is 1. Default is 2.
- `--delay MILLISECONDS`: How many MILLISECONDS to delay the display of each caption, to mimic a real-time experience. This option is only applicable when you use the `realTime` flag. Minimum is 0.0. Default is 1000.
- `--remainTime MILLISECONDS`: How many MILLISECONDS a caption should remain on screen if it is not replaced by another. Minimum is 0.0. Default is 1000.
- `--quiet`: Suppress console output, except errors.
- `--profanity OPTION`: Valid values: raw, remove, mask. For more information, see [Profanity filter](#) concepts.

- `--threshold NUMBER`: Set stable partial result threshold. The default value is `3`. This option is only applicable when you use the `realTime` flag. For more information, see [Get partial results](#) concepts.

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Speech resource you created.

Next steps

[Learn more about speech recognition](#)

Call center overview

08/07/2025

Azure AI Language and Azure AI Speech can help you realize partial or full automation of telephony-based customer interactions, and provide accessibility across multiple channels. With the Language and Speech services, you can further analyze call center transcriptions, extract and redact conversation (PII), summarize the transcription, and detect the sentiment.

Some example scenarios for the implementation of Azure AI services in call and contact centers are:

- Virtual agents: Conversational AI-based telephony-integrated voice bots and voice-enabled chatbots
- Agent-assist: Real-time transcription and analysis of a call to improve the customer experience by providing insights and suggest actions to agents
- Post-call analytics: Post-call analysis to create insights into customer conversations to improve understanding and support continuous improvement of call handling, optimization of quality assurance and compliance control as well as other insight driven optimizations.

💡 Tip

Try the [Language Studio](#) or [Speech Studio](#) for a demonstration on how to use the Language and Speech services to analyze call center conversations.

To deploy a call center transcription solution to Azure with a no-code approach, try the [Ingestion Client](#).

Azure AI services features for call centers

A holistic call center implementation typically incorporates technologies from the Language and Speech services.

Audio data typically used in call centers generated through landlines, mobile phones, and radios are often narrowband, in the range of 8 KHz, which can create challenges when you're converting speech to text. The Speech service recognition models are trained to ensure that you can get high-quality transcriptions, however you choose to capture the audio.

Once you transcribe your audio with the Speech service, you can use the Language service to perform analytics on your call center data such as: sentiment analysis, summarizing the reason

for customer calls, how they were resolved, extracting and redacting conversation PII, and more.

Speech service

The Speech service offers the following features that can be used for call center use cases:

- [Real-time speech to text](#): Recognize and transcribe audio in real-time from multiple inputs. For example, with virtual agents or agent-assist, you can continuously recognize audio input and control how to process results based on multiple events.
- [Batch speech to text](#): Transcribe large amounts of audio files asynchronously including speaker diarization and is typically used in post-call analytics scenarios. Diarization is the process of recognizing and separating speakers in mono channel audio data.
- [Text to speech](#): Text to speech enables your applications, tools, or devices to convert text into human like synthesized speech.
- [Speaker identification](#): Helps you determine an unknown speaker's identity within a group of enrolled speakers and is typically used for call center customer verification scenarios or fraud detection.
- [Language Identification](#): Identify languages spoken in audio and can be used in real-time and post-call analysis for insights or to control the environment (such as output language of a virtual agent).

You might want to further customize and fine-tune the experience for your product or environment. Typical examples for Speech fine-tuning include:

 Expand table

Speech customization	Description
Custom speech	A speech to text feature used to evaluate and improve the speech recognition accuracy of use-case specific entities (such as alpha-numeric customer, case, and contract IDs, license plates, and names). You can also train a custom model with your own product names and industry terminology.
Custom voice	A text to speech feature that lets you create a one-of-a-kind, customized, synthetic voice for your applications.

Language service

The Language service offers the following features that can be used for call center use cases:

- **Personally Identifiable Information (PII) extraction and redaction:** Identify, categorize, and redact sensitive information in conversation transcription.
- **Conversation summarization:** Summarize in abstract text what each conversation participant said about the issues and resolutions. For example, a call center can group product issues that have a high volume.
- **Sentiment analysis and opinion mining:** Analyze transcriptions and associate positive, neutral, or negative sentiment at the utterance and conversation-level.

You might want to further customize and fine-tune models to extract more information from your data. Typical examples for Language customization include:

 Expand table

Language customization	Description
Custom NER (named entity recognition)	Improve the detection and extraction of entities in transcriptions.
Custom text classification	Classify and label transcribed utterances with either single or multiple classifications.

You can find an overview of all Language service features and customization options [here](#).

Next steps

- [Post-call transcription and analytics quickstart](#)
- [Try out the Language Studio ↗](#)
- [Try out the Speech Studio ↗](#)

Quickstart: Post-call transcription and analytics

08/07/2025

[Language service documentation](#) | [Language Studio](#) ↗ | [Speech service documentation](#) | [Speech Studio](#) ↗

In this C# quickstart, you perform sentiment analysis and conversation summarization of [call center](#) transcriptions. The sample will automatically identify, categorize, and redact sensitive information. The quickstart implements a cross-service scenario that uses features of the [Azure Cognitive Speech](#) and [Azure Cognitive Language](#) services.

💡 Tip

Try the [Language Studio](#) ↗ or [Speech Studio](#) ↗ for a demonstration on how to use the Language and Speech services to analyze call center conversations.

To deploy a call center transcription solution to Azure with a no-code approach, try the [Ingestion Client](#).

The following Azure AI services for Speech features are used in the quickstart:

- [Batch transcription](#): Submit a batch of audio files for transcription.
- [Speaker separation](#): Separate multiple speakers through diarization of mono 16khz 16 bit PCM wav files.

The Language service offers the following features that are used:

- [Personally Identifiable Information \(PII\) extraction and redaction](#): Identify, categorize, and redact sensitive information in conversation transcription.
- [Conversation summarization](#): Summarize in abstract text what each conversation participant said about the issues and resolutions. For example, a call center can group product issues that have a high volume.
- [Sentiment analysis and opinion mining](#): Analyze transcriptions and associate positive, neutral, or negative sentiment at the utterance and conversation-level.

Prerequisites

- ✓ Azure subscription - [Create one for free](#) ↗

- ✓ Create a multi-service resource [↗](#) in the Azure portal. This quickstart only requires one Azure AI services [multi-service resource](#). The sample code allows you to specify separate Language and Speech resource keys.
- ✓ Get the resource key and region. After your Azure AI Foundry resource is deployed, select **Go to resource** to view and manage keys.

Important

This quickstart requires access to [conversation summarization](#). To get access, you must submit an [online request](#) [↗](#) and have it approved.

The `--languageKey` and `--languageEndpoint` values in this quickstart must correspond to a resource that's in one of the regions supported by the [conversation summarization API](#) [↗](#): `eastus`, `northeurope`, and `uksouth`.

Run post-call transcription analysis with C#

Follow these steps to build and run the post-call transcription analysis quickstart code example.

1. Copy the [scenarios/csharp/dotnetcore/call-center/](#) [↗](#) sample files from GitHub. If you have [Git installed](#) [↗](#), open a command prompt and run the `git clone` command to download the Speech SDK samples repository.

.NET CLI

```
git clone https://github.com/Azure-Samples/cognitive-services-speech-sdk.git
```

2. Open a command prompt and change to the project directory.

.NET CLI

```
cd <your-local-path>/scenarios/csharp/dotnetcore/call-center/call-center/
```

3. Build the project with the .NET CLI.

.NET CLI

```
dotnet build
```

4. Run the application with your preferred command line arguments. See [usage and arguments](#) for the available options.

Here's an example that transcribes from an example audio file at [GitHub](#):

.NET CLI

```
dotnet run --languageKey YourResourceKey --languageEndpoint  
YourResourceEndpoint --speechKey YourResourceKey --speechRegion  
YourResourceRegion --input "https://github.com/Azure-Samples/cognitive-  
services-speech-sdk/raw/master/scenarios/call-  
center/sampleddata/Call1_separated_16k_health_insurance.wav" --stereo --  
output summary.json
```

If you already have a transcription for input, here's an example that only requires a Language resource:

.NET CLI

```
dotnet run --languageKey YourResourceKey --languageEndpoint  
YourResourceEndpoint --jsonInput "YourTranscriptionFile.json" --stereo --  
output summary.json
```

Replace `YourResourceKey` with your Azure AI Foundry resource key, replace `YourResourceRegion` with your Azure AI Foundry resource `region` (such as `eastus`), and replace `YourResourceEndpoint` with your Azure AI services endpoint. Make sure that the paths specified by `--input` and `--output` are valid. Otherwise you must change the paths.

 **Important**

Remember to remove the key from your code when you're done, and never post it publicly. For production, use a secure way of storing and accessing your credentials like [Azure Key Vault](#). See the Azure AI services [security](#) article for more information.

Check results

The console output shows the full conversation and summary. Here's an example of the overall summary, with redactions for brevity:

Output

Conversation summary:

issue: Customer wants to sign up for insurance.

resolution: Customer was advised that customer would be contacted by the insurance company.

If you specify the `--output FILE` optional **argument**, a JSON version of the results are written to the file. The file output is a combination of the JSON responses from the [batch transcription](#) (Speech), [sentiment](#) (Language), and [conversation summarization](#) (Language) APIs.

The `transcription` property contains a JSON object with the results of sentiment analysis merged with batch transcription. Here's an example, with redactions for brevity:

JSON

```
{  
    "source": "https://github.com/Azure-Samples/cognitive-services-speech-  
    sdk/raw/master/scenarios/call-  
    center/sampleddata/Call1_separated_16k_health_insurance.wav",  
    // Example results redacted for brevity  
    "nBest": [  
        {  
            "confidence": 0.77464247,  
            "lexical": "hello thank you for calling contoso who am i speaking with  
today",  
            "itn": "hello thank you for calling contoso who am i speaking with  
today",  
            "maskedITN": "hello thank you for calling contoso who am i speaking  
with today",  
            "display": "Hello, thank you for calling Contoso. Who am I speaking  
with today?",  
            "sentiment": {  
                "positive": 0.78,  
                "neutral": 0.21,  
                "negative": 0.01  
            }  
        },  
    ]  
    // Example results redacted for brevity  
}
```

The `conversationAnalyticsResults` property contains a JSON object with the results of the conversation PII and conversation summarization analysis. Here's an example, with redactions for brevity:

JSON

```
{  
    "conversationAnalyticsResults": {  
        "conversationSummaryResults": {  
            "conversations": [  
                {  
                    "id": "conversation1",  
                    "summaries": [  
                        {  
                            "aspect": "issue",  
                            "text": "I'm having trouble with my account.  
                            It's not letting me log in."  
                        }  
                    ]  
                }  
            ]  
        }  
    }  
}
```

```
        "text": "Customer wants to sign up for insurance"
    },
    {
        "aspect": "resolution",
        "text": "Customer was advised that customer would be contacted by
the insurance company"
    }
],
"warnings": []
},
],
"errors": [],
"modelVersion": "2022-05-15-preview"
},
"conversationPiiResults": {
    "combinedRedactedContent": [
        {
            "channel": "0",
            "display": "Hello, thank you for calling Contoso. Who am I speaking with
today? Hi, ****. Uh, are you calling because you need health insurance?", // Example results redacted for brevity
            "itn": "hello thank you for calling contoso who am i speaking with today
hi **** uh are you calling because you need health insurance", // Example results redacted for brevity
            "lexical": "hello thank you for calling contoso who am i speaking with
today hi **** uh are you calling because you need health insurance" // Example results redacted for brevity
        },
        {
            "channel": "1",
            "display": "Hi, my name is *****. I'm trying to enroll myself with
Contoso. Yes. Yeah, I'm calling to sign up for insurance.", // Example results redacted for brevity
            "itn": "hi my name is ***** i'm trying to enroll myself with
contoso yes yeah i'm calling to sign up for insurance", // Example results redacted for brevity
            "lexical": "hi my name is ***** i'm trying to enroll myself with
contoso yes yeah i'm calling to sign up for insurance" // Example results redacted for brevity
        }
    ],
    "conversations": [
        {
            "id": "conversation1",
            "conversationItems": [
                {
                    "id": "0",
                    "redactedContent": {
                        "itn": "hello thank you for calling contoso who am i speaking with
today",
                        "lexical": "hello thank you for calling contoso who am i speaking
with today",
                        "text": "Hello, thank you for calling Contoso. Who am I speaking
with today?"
                    },
                    "text": "Hello, thank you for calling Contoso. Who am I speaking
with today?"
                }
            ]
        }
    ]
}
```

```
        "entities": [],
        "channel": "0",
        "offset": "PT0.77S"
    },
    {
        "id": "1",
        "redactedContent": {
            "itn": "hi my name is ***** i'm trying to enroll myself with contoso",
            "lexical": "hi my name is ***** i'm trying to enroll myself with contoso",
            "text": "Hi, my name is *****. I'm trying to enroll myself with Contoso."
        },
        "entities": [
            {
                "text": "Mary Rondo",
                "category": "Name",
                "offset": 15,
                "length": 10,
                "confidenceScore": 0.97
            }
        ],
        "channel": "1",
        "offset": "PT4.55S"
    },
    {
        "id": "2",
        "redactedContent": {
            "itn": "hi **** uh are you calling because you need health insurance",
            "lexical": "hi **** uh are you calling because you need health insurance",
            "text": "Hi, ****. Uh, are you calling because you need health insurance?"
        },
        "entities": [
            {
                "text": "Mary",
                "category": "Name",
                "offset": 4,
                "length": 4,
                "confidenceScore": 0.93
            }
        ],
        "channel": "0",
        "offset": "PT9.55S"
    },
    {
        "id": "3",
        "redactedContent": {
            "itn": "yes yeah i'm calling to sign up for insurance",
            "lexical": "yes yeah i'm calling to sign up for insurance",
            "text": "Yes. Yeah, I'm calling to sign up for insurance."
        },
    }
```

```
        "entities": [],
        "channel": "1",
        "offset": "PT13.09S"
    },
// Example results redacted for brevity
    ],
    "warnings": []
}
]
}
}
}
```

Usage and arguments

Usage: `call-center -- [...]`

Important

You can use a [multi-service](#) resource or separate [Language](#) and [Speech](#) resources. In either case, the `--languageKey` and `--languageEndpoint` values must correspond to a resource that's in one of the regions supported by the [conversation summarization API](#): `eastus`, `northeurope`, and `uksouth`.

Connection options include:

- `--speechKey KEY`: Your [AI Foundry](#) resource key. Required for audio transcriptions with the `--input` from URL option.
- `--speechRegion REGION`: Your [AI Foundry](#) resource region. Required for audio transcriptions with the `--input` from URL option. Examples: `eastus`, `northeurope`
- `--languageKey KEY`: Your [AI Foundry](#) resource key. Required.
- `--languageEndpoint ENDPOINT`: Your [AI Foundry](#) resource endpoint. Required. Example:
`https://YourResourceName.cognitiveservices.azure.com`

Input options include:

- `--input URL`: Input audio from URL. You must set either the `--input` or `--jsonInput` option.
- `--jsonInput FILE`: Input an existing batch transcription JSON result from FILE. With this option, you only need a Language resource to process a transcription that you already

have. With this option, you don't need an audio file or an AI Foundry resource for Speech. Overrides `--input`. You must set either the `--input` or `--jsonInput` option.

- `--stereo`: Indicates that the audio via ``input URL`` should be in stereo format. If stereo isn't specified, then mono 16khz 16 bit PCM wav files are assumed. Diarization of mono files is used to separate multiple speakers. Diarization of stereo files isn't supported, since 2-channel stereo files should already have one speaker per channel.
- `--certificate`: The PEM certificate file. Required for C++.

Language options include:

- `--language LANGUAGE`: The language to use for sentiment analysis and conversation analysis. This value should be a two-letter ISO 639-1 code. The default value is `en`.
- `--locale LOCALE`: The locale to use for batch transcription of audio. The default value is `en-US`.

Output options include:

- `--help`: Show the usage help and stop
- `--output FILE`: Output the transcription, sentiment, conversation PII, and conversation summaries in JSON format to a text file. For more information, see [output examples](#).

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Azure AI Foundry resource you created.

Next steps

[Try the Ingestion Client](#)

Ingestion Client with Azure AI services

08/07/2025

The Ingestion Client is a tool released by Microsoft on GitHub that helps you quickly deploy a call center transcription solution to Azure with a no-code approach.

💡 Tip

You can use the tool and resulting solution in production to process a high volume of audio.

Ingestion Client uses the [Azure AI Language](#), [Azure AI Speech](#), [Azure storage ↗](#), and [Azure Functions ↗](#).

Get started with the Ingestion Client

An Azure account and a multi-service Azure AI Foundry resource are needed to run the Ingestion Client.

- Azure subscription - [Create one for free ↗](#)
- [Create an AI Foundry resource ↗](#) in the Azure portal.
- Get the resource key and region. After your resource is deployed, select **Go to resource** to view and manage keys. For more information about Azure AI Foundry resources, see [this quickstart](#).

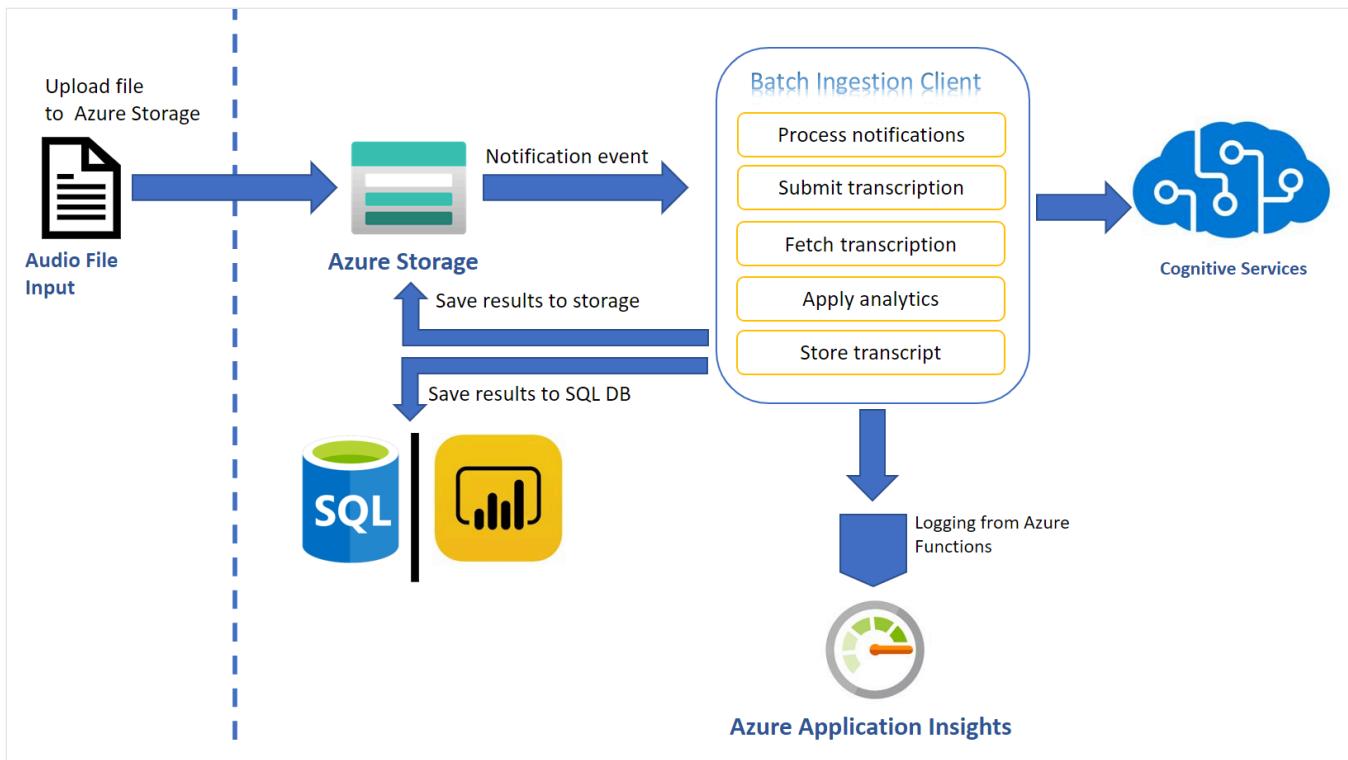
Ingestion Client Features

The Ingestion Client works by connecting a dedicated [Azure storage ↗](#) account to custom [Azure Functions ↗](#) in a serverless fashion to pass transcription requests to the service. The transcribed audio files land in the dedicated [Azure Storage container ↗](#).

ⓘ Important

Pricing varies depending on the mode of operation (batch vs real-time) as well as the Azure Function SKU selected. By default the tool will create a Premium Azure Function SKU to handle large volume. Visit the [Pricing ↗](#) page for more information.

Internally, the tool uses Speech and Language services, and follows best practices to handle scale-up, retries and failover. The following schematic describes the resources and connections.



The following Speech service feature is used by the Ingestion Client:

- **Batch speech to text:** Transcribe large amounts of audio files asynchronously including speaker diarization and is typically used in post-call analytics scenarios. Diarization is the process of recognizing and separating speakers in mono channel audio data.

Here are some Language service features that are used by the Ingestion Client:

- **Personally Identifiable Information (PII) extraction and redaction:** Identify, categorize, and redact sensitive information in conversation transcription.
- **Sentiment analysis and opinion mining:** Analyze transcriptions and associate positive, neutral, or negative sentiment at the utterance and conversation-level.

Besides Azure AI services, these Azure products are used to complete the solution:

- **Azure storage ↗:** Used for storing telephony data and the transcripts that batch transcription API returns. This storage account should use notifications, specifically for when new files are added. These notifications are used to trigger the transcription process.
- **Azure Functions ↗:** Used for creating the shared access signature (SAS) URI for each recording, and triggering the HTTP POST request to start a transcription. Additionally, you use Azure Functions to create requests to retrieve and delete transcriptions by using the Batch Transcription API.

Tool customization

The tool is built to show customers results quickly. You can customize the tool to your preferred SKUs and setup. The SKUs can be edited from the [Azure portal](#) and [the code itself is available on GitHub](#).

 **Note**

We suggest creating the resources in the same dedicated resource group to understand and track costs more easily.

Next steps

- [Learn more about Azure AI services features for call center](#)
- [Explore the Language service features](#)
- [Explore the Speech service features](#)

Telephony integration

08/07/2025

To support real-time scenarios, like Virtual Agent and Agent Assist in call centers, an integration with the call center's telephony system is required.

Typically, integration with the Speech service is handled by a telephony client connected to the customers SIP/RTP processor, for example, to a Session Border Controller (SBC).

Usually the telephony client handles the incoming audio stream from the SIP/RTP processor, the conversion to PCM and connects the streams using continuous recognition. It also triages the processing of the results. For example, analysis of speech transcripts for Agent Assist or connect with a dialog processing engine (for example, Azure Botframework or Power Virtual Agent) for Virtual Agent.

For easier integration the Speech service also supports "ALAW in WAV container" and "MULAW in WAV container" for audio streaming. To build this integration, we recommend using the [Speech SDK](#).

Azure Communication Services

[Azure Communication Services](#) calls automation APIs provide telephony integration, real-time event triggers to perform actions based on custom business logic specific to their domain. Within the call automation APIs developers can use simple AI powered APIs, which can be used to play personalized greeting messages, recognize conversational voice inputs to gather information on contextual questions to drive a more self-service model with customers, use sentiment analysis to improve customer service overall. These content specific APIs are orchestrated through Azure AI services with support for customization of AI models without developers needing to terminate media streams on their services and streaming back to Azure for AI functionality. For more information, see [Azure Communication Services](#).

Next steps

- [Learn about Speech SDK](#)
- [How to lower speech synthesis latency](#)

Connect Azure Communication Services with Azure AI services

06/04/2025

Azure Communication Services Call Automation APIs allow developers to steer and control calls made through Azure Communication Services, including telephony, VoIP, and WebRTC. These APIs use real-time event triggers, which enable actions based on custom business logic that is specific to each developer's domain. With Call Automation APIs, developers can use simple AI-powered features. For example, they can play personalized greetings, recognize spoken responses to gather information from customers, and analyze sentiment to improve service. These targeted APIs are managed through **Azure AI Foundry**, which allow developers to customize AI models. Importantly, developers do not need to deal with media streams or send them back to Azure for these AI functions—the processing happens seamlessly.

All this functionality is possible with one click, allowing enterprises to access a secure solution and link their models through the portal. Furthermore, developers and enterprises don't need to manage credentials. Connecting your Azure AI services uses managed identities to access user-owned resources. Developers can use managed identities to authenticate any resource that supports Microsoft Entra authentication.

Azure AI services can be easily integrated into any application regardless of the programming language. When creating an Azure Resource in Azure portal, enable the option and provide the URL to the Azure AI services. This simple experience allows developers to meet their needs, scale, and avoid investing time and resources into designing and maintaining a custom solution.

!**Note**

This integration only supports Multi-service Cognitive Service resource, we recommend if you're creating a new Azure AI Service resource you create a Multi-service Cognitive Service resource or when you're connecting an existing resource confirm that it's a Multi-service Cognitive Service resource.

Common use cases

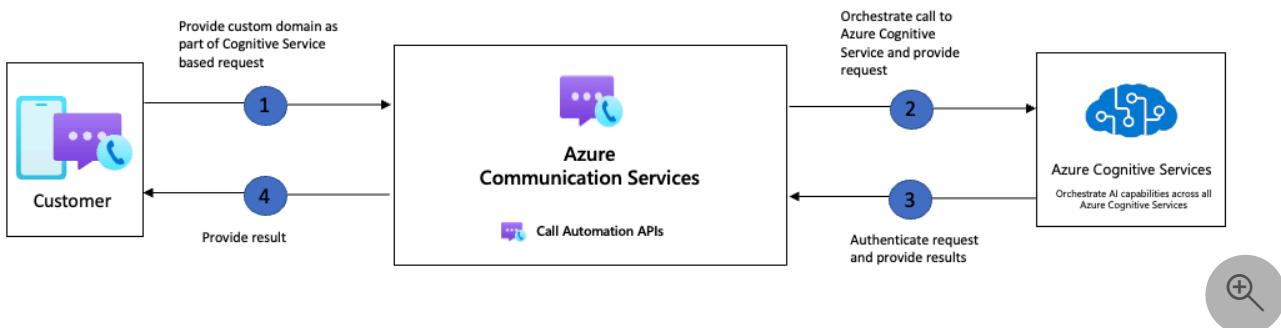
Build applications that can play and recognize speech

With the ability to connect your Azure AI services to Azure Communication Services. You can enable custom play functionality, using [Text-to-Speech](#) and [Speech Synthesis Markup](#)

Language (SSML) configuration, to play more customized and natural sounding audio to users. Through the Azure AI services connection, you can also use the Speech-To-Text service to incorporate recognition of voice responses that can be converted into actionable tasks through business logic in the application. These functions can be further enhanced within Azure AI services by:

- Creating custom models tailored to your domain and region
- Selecting which languages are spoken and recognized
- Designing custom voices
- Building additional models based on your experience

Runtime flow



Azure portal experience

You need to connect your Azure Communication Services resource with the Azure AI resource through the Azure portal. There are two ways you can accomplish this step:

- Navigating through the steps of the Cognitive Services tab in your Azure Communication Services (recommended).
- Manually adding the Managed Identity to your Azure Communication Services resource. This step is more advanced and requires a little more effort to connect your Azure Communication Services to your Azure AI services.

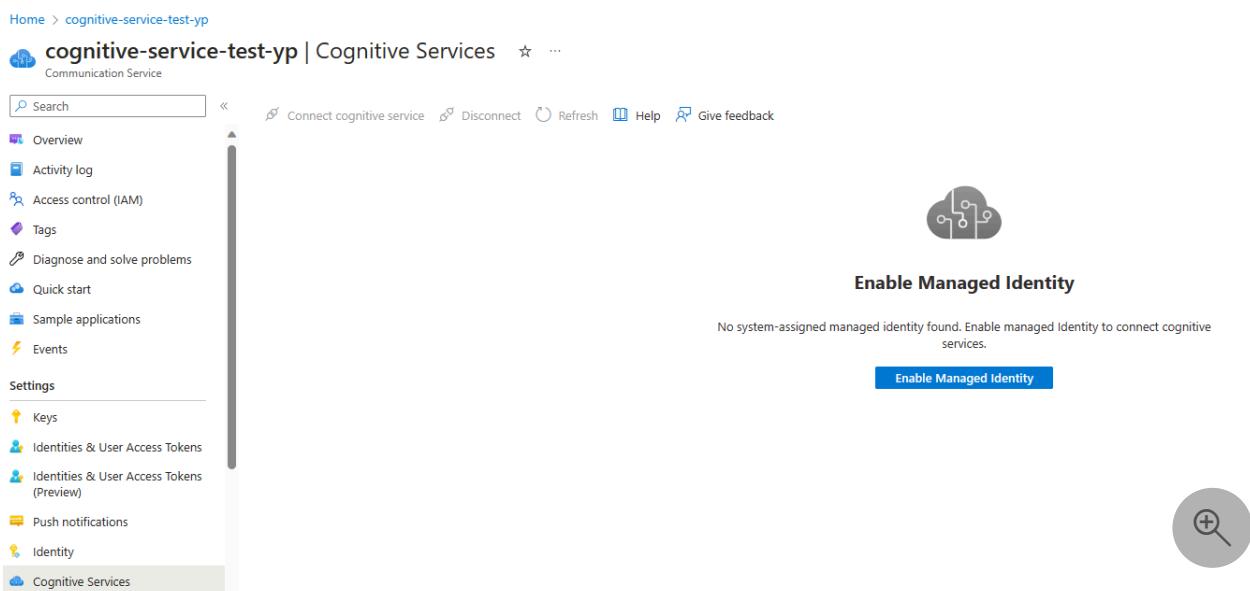
Prerequisites

- Azure account with an active subscription and access to Azure portal, for details see [Create an account for free ↗](#).
- Azure Communication Services resource. See [Create an Azure Communication Services resource](#).

- Azure Communication Service [Microsoft.Authorization/roleAssignments/write](#) permissions, commonly done through Azure RBAC. See [Assign Azure roles using the Azure portal](#).
- An [Azure AI Services resource](#) .

Connecting through the Azure portal

1. Open your Azure Communication Services resource and click on the Cognitive Services tab.
2. If system-assigned managed identity isn't enabled, you need to enable it.
3. In the Cognitive Services tab, click on "Enable Managed Identity" button.



4. Enable system assigned identity. This action begins the creation of the identity; A pop-up notification appears notifying you that the request is being processed.

Home > my-acs-test-resource | Overview > my-acs-test-resource

my-acs-test-resource | Identity ⋮

Search

Overview Activity log Access control (IAM) Tags Diagnose and solve problems Quick start Sample applications Events

Settings

- Keys
- Identities & User Access Tokens
- Identities & User Access Tokens (Preview)
- Push notifications
- Identity**
- Properties
- Locks

Telephony and SMS

- Try SMS
- Phone numbers

Save Discard Refresh Got feedback?

Status **On**

5. Once the identity is enabled, you should see something similar.

Identity ⋮

System assigned (preview)

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle credentials in code. [Learn more about Managed identities.](#)

Save Discard Refresh Got feedback?

Status **On**

Object (principal) ID

Permissions
Azure role assignments

This resource is registered with Azure Active Directory. The managed identity can be configured

6. When managed identity is enabled, the Cognitive Service tab should show a button 'Connect cognitive service' to connect the two services.

cognitive-service-test-yp | Cognitive Services

Communication Service

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Quick start

Sample applications

Events

Settings

- Keys
- Identities & User Access Tokens
- Identities & User Access Tokens (Preview)
- Push notifications
- Identity
- Cognitive Services

Connect cognitive service

Disconnect Refresh Help Give feedback

Connect cognitive Service

Connect cognitive services to embed the ability to see, hear, speak, search, understand, and accelerate advanced decision-making into your communication apps.

Connect cognitive service

Don't have any cognitive services? Learn how to get started [Get started](#)

Need help connecting a cognitive service? View the guide [View the guide](#)

Cloud icon with a person icon inside

- Click on 'Connect cognitive service', select the Subscription, Resource Group and Resource and click 'Connect' in the context pane that opens up.

cognitive-service-test-yp | Cognitive Services

Communication Service

Search

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Quick start

Sample applications

Events

Settings

- Keys
- Identities & User Access Tokens
- Identities & User Access Tokens (Preview)
- Push notifications
- Identity
- Cognitive Services
- Properties
- Locks

Telephony and SMS

- Try SMS
- Phone numbers
- Regulatory Documents
- Registered countries/regions
- Operators
- Alphanumeric Sender ID

Connect cognitive service

Disconnect Refresh Help Give feedback

Connect cognitive service

Connect to a cognitive service for use in your application. You can modify the cognitive service from the resource at any time.

Subscription *

Resource Group *

Resource *

Connect

Don't have any cognitive services? Learn how to get started [Get started](#)

Need help connecting a cognitive service? View the guide [View the guide](#)

Cloud icon with a person icon inside

- If connection is successful, you should see a green banner confirming successful connection.

9. Now in the Cognitive Service tab you should see your connected services showing up.

Resource Name	Subscription	Kind	Location
my-cognitiveservice-test-resource	Teams-SpoolGateway-DEV	CognitiveServices	westus

Advanced option: Manually adding Managed Identity to Azure Communication Services resource

Alternatively if you would like to go through the manual process of connecting your resources you can follow these steps.

Enable system assigned identity

1. Navigate to your Azure Communication Services resource in the Azure portal.
2. Select the Identity tab.

3. Enable system assigned identity. This action begins the creation of the identity. A pop-up notification appears notifying you that the request is being processed.

Enable system assigned managed identity
'my-acss-test-resource' will be registered with Azure Active Directory. Once it is registered, 'my-acss-test-resource' can be granted permissions to access resources protected by Azure AD. Do you want to enable the system assigned managed identity for 'my-acss-test-resource'?

Save Discard Refresh Got feedback?

Status:

Option 1: Add role from Azure Cognitive Services in the Azure portal

1. Navigate to your Azure Cognitive Services resource.
2. Select the "Access control (IAM)" tab.
3. Click the "+ Add" button.
4. Select "Add role assignments" from the menu.

Home > my-cognitiveservice-test-resource

my-cognitiveservice-test-resource | Access control (IAM)

Add role assignment Add co-administrator My access Check access Grant access to this resource View access to this resource View deny assignments

5. Choose the "Cognitive Services User" role to assign, then click "Next."

The screenshot shows the 'Add role assignment' page in the Azure portal. At the top, there are tabs for 'Role', 'Members' (which is selected), and 'Review + assign'. Below this, a search bar contains the text 'cognitive service user'. To the right of the search bar are filters for 'Type: All' and 'Category: All'. A table lists various roles, each with a 'Name', 'Description', 'Type', 'Category', and 'Details' column. The 'Cognitive Services User' role is highlighted in the list.

Name ↑↓	Description ↑↓	Type ↑↓	Category ↑↓	Details
Cognitive Services Custom Vision Con...	Full access to the project, including the ability to view, create, edit, or delete projects.	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vision Dep...	Publish, unpublish or export models. Deployment can view the project but can't update.	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vision Lab...	View, edit training images and create, add, remove, or delete the image tags. Labelers can view th...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vision Rea...	Read-only actions in the project. Readers can't create or update the project.	BuiltInRole	AI + Machine Learning	View
Cognitive Services Custom Vision Trai...	View, edit projects and train the models, including the ability to publish, unpublish, export the mo...	BuiltInRole	AI + Machine Learning	View
Cognitive Services Data Reader (Previe...	Lets you read Cognitive Services data.	BuiltInRole	Preview	View
Cognitive Services Immersive Reader ...	Provides access to create Immersive Reader sessions and call APIs	BuiltInRole	None	View
Cognitive Services LUIS Owner	Has access to all Read, Test, Write, Deploy and Delete functions under LUIS	BuiltInRole	None	View
Cognitive Services LUIS Reader	Has access to Read and Test functions under LUIS.	BuiltInRole	None	View
Cognitive Services LUIS Writer	Has access to all Read, Test, and Write functions under LUIS	BuiltInRole	None	View
Cognitive Services Metrics Advisor User	Access to the project.	BuiltInRole	AI + Machine Learning	View
Cognitive Services OpenAI User	Ability to view files, models, deployments. Readers can't make any changes They can inference	BuiltInRole	None	View
Cognitive Services Speech User	Access to the real-time speech recognition and batch transcription APIs, real-time speech synthesi...	BuiltInRole	AI + Machine Learning	View
Cognitive Services User	Lets you read and list keys of Cognitive Services.	BuiltInRole	AI + Machine Learning	View

Below the table are navigation buttons: '< Previous', 'Page 1 of 1', and 'Next >'. At the bottom of the main page are buttons for 'Review + assign', 'Previous', and 'Next'.

6. For the field "Assign access to" choose the "User, group or service principal."

7. Press "+ Select members" and a side tab opens.

8. Search for your Azure Communication Services resource name in the text box and click it when it shows up, then click "Select."

The screenshot shows the 'Add role assignment' page with the 'Members' tab selected. The 'Selected role' is set to 'Cognitive Services User'. Under 'Assign access to', the radio button for 'User, group, or service principal' is selected. In the 'Members' section, there is a button '+ Select members'. A side panel titled 'Select members' is open, showing a search bar with the text 'my-acs-test-resource'. Below the search bar, the result 'my-acs-test-resource' is listed with a small icon. At the bottom of the side panel, there is a note about selected members and a link to 'Learn more about RBAC'.

9. Click "Review + assign," this assigns the role to the managed identity.

Option 2: Add role through Azure Communication Services Identity tab

1. Navigate to your Azure Communication Services resource in the Azure portal.
2. Select Identity tab.
3. Click on "Azure role assignments."

The screenshot shows the Azure portal interface. On the left, there is a sidebar titled 'Azure role assignments' with a table header 'Role' and 'Resource Name'. Below the table, it says 'No role assignments found for the selected subscription.' On the right, a modal window titled 'Add role assignment (Preview)' is open. It has fields for 'Scope' (set to 'Select a scope'), 'Subscription' (set to 'Contoso subscription'), and 'Role' (set to 'Select a role'). A small circular icon with a magnifying glass and a plus sign is located in the bottom right corner of the modal.

4. Click the "Add role assignment (Preview)" button, which opens the "Add role assignment (Preview)" tab.
5. Select the "Resource group" for "Scope."
6. Select the "Subscription."
7. Select the "Resource Group" containing the Cognitive Service.
8. Select the Role "Cognitive Services User."

The screenshot shows the 'Add role assignment (Preview)' dialog with the following selected values:

- Scope: Resource group
- Subscription: Contoso subscription
- Resource group: Contoso resource group
- Role: Cognitive Services User

A small circular icon with a magnifying glass and a plus sign is located in the bottom right corner of the modal.

9. Click Save.

Your Azure Communication Service has now been linked to your Azure Cognitive Service resource.

Azure AI services regions supported

Our integration between Azure Communication Services (ACS) and Azure AI is fully aligned with the regional availability of Azure AI Foundry. This means that ACS to Azure AI integration is supported in all regions where Azure AI Foundry is available.

For the most up-to-date list of supported regions, refer to the official [Azure AI Foundry documentation](#).

Known limitations

- Text-to-Speech text prompts support a maximum of 400 characters, if your prompt is longer than this we suggest using SSML for Text-to-Speech based play actions.
- For scenarios where you exceed your Speech service quota limit, you can request to increase this limit by following the steps outlined [here](#).

Next steps

- Learn about [playing audio](#) to callers using Text-to-Speech.
- Learn about [gathering user input](#) with Speech-to-Text.

Game development with Azure AI Speech

08/07/2025

Azure AI Speech can be used to improve various gaming scenarios, both in-game and out-of-game.

Here are a few Speech features to consider for flexible and interactive game experiences:

- Bring everyone into the conversation by synthesizing audio from text. Or by displaying text from audio.
- Make the game more accessible for players who are unable to read text in a particular language, including young players who don't read or write. Players can listen to storylines and instructions in their preferred language.
- Create game avatars and nonplayable characters (NPC) that can initiate or participate in a conversation in-game.
- Standard voice can provide highly natural out-of-box voices with leading voice variety in terms of a large portfolio of languages and voices.
- Custom voice for creating a voice that stays on-brand with consistent quality and speaking style. You can add emotions, accents, nuances, laughter, and other para linguistic sounds and expressions.
- Use game dialogue prototyping to shorten the amount of time and money spent in product to get the game to market sooner. You can rapidly swap lines of dialog and listen to variations in real-time to iterate the game content.

You can use the [Speech SDK](#) or [Speech CLI](#) for real-time low latency speech to text, text to speech, language identification, and speech translation. You can also use the [Batch transcription API](#) to transcribe prerecorded speech to text. To synthesize a large volume of text input (long and short) to speech, use the [Batch synthesis API](#).

For information about locale and regional availability, see [Language and voice support](#) and [Region support](#).

Text to speech

Help bring everyone into the conversation by converting text messages to audio using [Text to speech](#) for scenarios, such as game dialogue prototyping, greater accessibility, or nonplayable character (NPC) voices. Text to speech includes [standard voice](#) and [custom voice](#) features. Standard voice can provide highly natural out-of-box voices with leading voice variety in terms of a large portfolio of languages and voices. Custom voice is an easy-to-use self-service for creating a highly natural custom voice.

When enabling this functionality in your game, keep in mind the following benefits:

- Voices and languages supported - A large portfolio of [locales and voices](#) are supported. You can also [specify multiple languages](#) for Text to speech output. For [custom voice](#), you can [choose to create](#) different languages from single language training data.
- Emotional styles supported - [Emotional tones](#), such as cheerful, angry, sad, excited, hopeful, friendly, unfriendly, terrified, shouting, and whispering. You can [adjust the speaking style](#), style degree, and role at the sentence level.
- Visemes supported - You can use visemes during real-time synthesizing to control the movement of 2D and 3D avatar models, so that the mouth movements are perfectly matched to synthetic speech. For more information, see [Get facial position with viseme](#).
- Fine-tuning Text to speech output with Speech Synthesis Markup Language (SSML) - With SSML, you can customize Text to speech outputs, with richer voice tuning supports. For more information, see [Speech Synthesis Markup Language \(SSML\) overview](#).
- Audio outputs - Each standard voice model is available at 24 kHz and high-fidelity 48 kHz. If you select 48-kHz output format, the high-fidelity voice model with 48 kHz is invoked accordingly. The sample rates other than 24 kHz and 48 kHz can be obtained through upsampling or downsampling when synthesizing. For example, 44.1 kHz is downsampled from 48 kHz. Each audio format incorporates a bitrate and encoding type. For more information, see the [supported audio formats](#). For more information on 48-kHz high-quality voices, see [this introduction blog](#) ↗.

For an example, see the [text to speech quickstart](#).

Speech to text

You can use [speech to text](#) to display text from the spoken audio in your game. For an example, see the [Speech to text quickstart](#).

Language identification

With [language identification](#), you can detect the language of the chat string submitted by the player.

Speech translation

It's not unusual that players in the same game session natively speak different languages and might appreciate receiving both the original message and its translation. You can use [speech translation](#) to translate text between languages so players across the world can communicate with each other in their native language.

For an example, see the [Speech translation quickstart](#).

 **Note**

Besides the Speech service, you can also use the [Translator service](#). To execute text translation between supported source and target languages in real-time see [Text translation](#).

Next steps

- [Azure gaming documentation](#)
- [Text to speech quickstart](#)
- [Speech to text quickstart](#)
- [Speech translation quickstart](#)

Implement language identification

05/25/2025

Language identification is used to identify languages spoken in audio when compared against a list of supported languages.

Language identification (LID) use cases include:

- Speech to text recognition when you need to identify the language in an audio source and then transcribe it to text.
- Speech translation when you need to identify the language in an audio source and then translate it to another language.

For speech recognition, the initial latency is higher with language identification. You should only include this optional feature as needed.

Set configuration options

Whether you use language identification with [speech to text](#) or with [speech translation](#), there are some common concepts and configuration options.

- Define a list of [candidate languages](#) that you expect in the audio.
- Decide whether to use [at-start](#) or [continuous](#) language identification.

Then you make a [recognize once](#) or [continuous recognition](#) request to the Speech service.

This article provides code snippets to describe the concepts. Links to complete samples for each use case are provided.

Candidate languages

You provide candidate languages with the `AutoDetectSourceLanguageConfig` object. You expect that at least one of the candidates is in the audio. You can include up to four languages for [at-start LID](#) or up to 10 languages for [continuous LID](#). The Speech service returns one of the candidate languages provided even if those languages weren't in the audio. For example, if `fr-FR` (French) and `en-US` (English) are provided as candidates, but German is spoken, the service returns either `fr-FR` or `en-US`.

You must provide the full locale with dash (-) separator, but language identification only uses one locale per base language. Don't include multiple locales for the same language, for example, `en-US` and `en-GB`.

C#

```
var autoDetectSourceLanguageConfig =
    AutoDetectSourceLanguageConfig.FromLanguages(new string[] { "en-US", "de-DE",
"zh-CN" });
```

For more information, see [supported languages](#).

At-start and Continuous language identification

Speech supports both at-start and continuous language identification (LID).

Note

Continuous language identification is only supported with Speech SDKs in C#, C++, Java ([for speech to text only](#)), JavaScript ([for speech to text only](#)), and Python.

- At-start LID identifies the language once within the first few seconds of audio. Use at-start LID if the language in the audio doesn't change. With at-start LID, a single language is detected and returned in less than 5 seconds.
- Continuous LID can identify multiple languages during the audio. Use continuous LID if the language in the audio could change. Continuous LID doesn't support changing languages within the same sentence. For example, if you're primarily speaking Spanish and insert some English words, it doesn't detect the language change per word.

You implement at-start LID or continuous LID by calling methods for [recognize once](#) or [continuous](#). Continuous LID is only supported with continuous recognition.

Recognize once or continuous

Language identification is completed with recognition objects and operations. Make a request to the Speech service for recognition of audio.

Note

Don't confuse recognition with identification. Recognition can be used with or without language identification.

Either call the "recognize once" method, or the start and stop continuous recognition methods. You choose from:

- Recognize once with At-start LID. Continuous LID isn't supported for recognize once.
- Use continuous recognition with at-start LID.
- Use continuous recognition with continuous LID.

The `SpeechServiceConnection_LanguageIdMode` property is only required for continuous LID.

Without it, the Speech service defaults to at-start LID. The supported values are `AtStart` for at-start LID or `Continuous` for continuous LID.

C#

```
// Recognize once with At-start LID. Continuous LID isn't supported for recognize once.  
var result = await recognizer.RecognizeOnceAsync();  
  
// Start and stop continuous recognition with At-start LID  
await recognizer.StartContinuousRecognitionAsync();  
await recognizer.StopContinuousRecognitionAsync();  
  
// Start and stop continuous recognition with Continuous LID  
speechConfig SetProperty(PropertyId.SpeechServiceConnection_LanguageIdMode,  
"Continuous");  
await recognizer.StartContinuousRecognitionAsync();  
await recognizer.StopContinuousRecognitionAsync();
```

Use speech to text

You use Speech to text recognition when you need to identify the language in an audio source and then transcribe it to text. For more information, see [Speech to text overview](#).

Note

Speech to text recognition with at-start language identification is supported with Speech SDKs in C#, C++, Python, Java, JavaScript, and Objective-C. Speech to text recognition with continuous language identification is only supported with Speech SDKs in C#, C++, Java, JavaScript, and Python.

Currently for speech to text recognition with continuous language identification, you must create a `SpeechConfig` from endpoint, as shown in code examples.

See more examples of speech to text recognition with language identification on [GitHub](#)↗.

Recognize once

C#

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
"YourSpeechKey");

var autoDetectSourceLanguageConfig =
    AutoDetectSourceLanguageConfig.FromLanguages(
        new string[] { "en-US", "de-DE", "zh-CN" });

using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
using (var recognizer = new SpeechRecognizer(
    speechConfig,
    autoDetectSourceLanguageConfig,
    audioConfig))
{
    var speechRecognitionResult = await recognizer.RecognizeOnceAsync();
    var autoDetectSourceLanguageResult =
        AutoDetectSourceLanguageResult.FromResult(speechRecognitionResult);
    var detectedLanguage = autoDetectSourceLanguageResult.Language;
}
```

Speech to text custom models

 Note

Language detection with custom models can only be used with real-time speech to text and speech translation. Batch transcription only supports language detection for default base models.

This sample shows how to use language detection with a custom endpoint. If the detected language is `en-US`, the example uses the default model. If the detected language is `fr-FR`, the example uses the custom model endpoint. For more information, see [Deploy a custom speech model](#).

C#

```
var sourceLanguageConfigs = new SourceLanguageConfig[]
{
    SourceLanguageConfig.FromLanguage("en-US"),
    SourceLanguageConfig.FromLanguage("fr-FR", "The Endpoint Id for custom model")
```

```
of fr-FR")
};

var autoDetectSourceLanguageConfig =
    AutoDetectSourceLanguageConfig.FromSourceLanguageConfigs(
        sourceLanguageConfigs);
```

Run speech translation

Use Speech translation when you need to identify the language in an audio source and then translate it to another language. For more information, see [Speech translation overview](#).

(!) Note

Speech translation with language identification is only supported with Speech SDKs in C#, C++, JavaScript, and Python.

See more examples of speech translation with language identification on [GitHub](#).

Recognize once

C#

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Microsoft.CognitiveServices.Speech.Translation;

public static async Task RecognizeOnceSpeechTranslationAsync()
{
    var endpointUrl = new Uri("YourSpeechResourceEndpoint");

    var config = SpeechTranslationConfig.FromEndpoint(endpointUrl,
    "YourSpeechResourceKey");

    // Source language is required, but currently ignored.
    string fromLanguage = "en-US";
    speechTranslationConfig.SpeechRecognitionLanguage = fromLanguage;

    speechTranslationConfig.AddTargetLanguage("de");
    speechTranslationConfig.AddTargetLanguage("fr");

    var autoDetectSourceLanguageConfig =
        AutoDetectSourceLanguageConfig.FromLanguages(new string[] { "en-US", "de-DE",
    "zh-CN" });

    using var audioConfig = AudioConfig.FromDefaultMicrophoneInput();

    using (var recognizer = new TranslationRecognizer(
```

```
        speechTranslationConfig,
        autoDetectSourceLanguageConfig,
        audioConfig))
    {

        Console.WriteLine("Say something or read from file...");
        var result = await
recognizer.RecognizeOnceAsync().ConfigureAwait(false);

        if (result.Reason == ResultReason.TranslatedSpeech)
        {
            var lidResult =
result.Properties.GetProperty(PropertyId.SpeechServiceConnection_AutoDetectSou
rceLanguageResult);

            Console.WriteLine($"RECOGNIZED in '{lidResult}': Text=
{result.Text}");
            foreach (var element in result.Translations)
            {
                Console.WriteLine($"      TRANSLATED into '{element.Key}':
{element.Value}");
            }
        }
    }
}
```

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 endpoint.

When you run language ID in a container, use the `SourceLanguageRecognizer` object instead of `SpeechRecognizer` or `TranslationRecognizer`.

For more information about containers, see the [language identification speech containers](#) how-to guide.

Implement speech to text batch transcription

To identify languages with [Batch transcription REST API](#), use `languageIdentification` property in the body of your [Transcriptions - Submit](#) request.



Warning

Batch transcription only supports language identification for default base models. If both language identification and a custom model are specified in the transcription request, the service falls back to use the base models for the specified candidate languages. This might result in unexpected recognition results.

If your speech to text scenario requires both language identification and custom models, use [real-time speech to text](#) instead of batch transcription.

The following example shows the usage of the `languageIdentification` property with four candidate languages. For more information about request properties, see [Create a batch transcription](#).

JSON

```
{  
    <...>  
  
    "properties": {  
        <...>  
  
        "languageIdentification": {  
            "candidateLocales": [  
                "en-US",  
                "ja-JP",  
                "zh-CN",  
                "hi-IN"  
            ]  
        },  
        <...>  
    }  
}
```

Related content

- [Try the speech to text quickstart](#)
- [Improve recognition accuracy with custom speech](#)
- [Use batch transcription](#)

Language learning with Azure AI Speech

08/07/2025

One of the most important aspects of learning a new language is speaking and listening. Azure AI Speech provides features that can be used to help language learners.

Pronunciation assessment

The [pronunciation assessment](#) feature provides comprehensive feedback to users on the accuracy, fluency, prosody, vocabulary usage, grammar correctness, and topic understanding of their speech. Language learners can apply the assessment results to speak and present in a new language with confidence. For information about availability of pronunciation assessment, see [supported languages](#) and [available regions](#).

The Pronunciation Assessment feature offers several benefits for educators, service providers, and students.

- For educators, it provides instant feedback, eliminates the need for time-consuming oral language assessments, and offers consistent and comprehensive assessments.
- For service providers, it offers high real-time capabilities, worldwide Azure AI Speech and supports growing global business.
- For students and learners, it provides a convenient way to practice and receive feedback, authoritative scoring to compare with native pronunciation and helps to follow the exact text order for long sentences or full documents.

Speech to text

[Speech to text](#) supports real-time language identification for multilingual language learning scenarios, help human-human interaction with better understanding and readable context.

Text to speech

[Text to speech](#) standard voices can read out learning materials natively and empower self-served learning. A broad portfolio of [languages and voices](#) are supported for AI teacher, content read aloud capabilities, and more. Microsoft is continuously working on bringing new languages to the world.

[Custom voice](#) is available for you to create a customized synthetic voice for your applications. Education companies are using this technology to personalize language learning, by creating

unique characters with distinct voices that match the culture and background of their target audience.

Next steps

- [How to use pronunciation assessment](#)
- [What is speech to text](#)
- [What is text to speech](#)
- [What is custom voice](#)

Pronunciation assessment in the Azure AI Foundry portal

09/16/2025

Important

Items marked (preview) in this article are currently in public preview. This preview is provided without a service-level agreement, and we don't recommend it for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Pronunciation assessment uses the speech-to-text capability to provide subjective and objective feedback for language learners. Practicing pronunciation and getting timely feedback are essential for improving language skills. Assessments conducted by experienced teachers can take much time and can be expensive for learners. Pronunciation assessment can help make the process more engaging and accessible to learners of all backgrounds.

This article describes how to use the pronunciation assessment tool without writing any code through the [Azure AI Foundry portal](#). For information about how to integrate pronunciation assessment in your speech applications, see [How to use pronunciation assessment](#).

Note

For information about availability of pronunciation assessment, see [supported languages](#) and [available regions](#).

Reading, speaking and gaming scenarios

For pronunciation assessment, there are three scenarios: Reading, Speaking and Gaming.

- Reading: This scenario is designed for [scripted assessment](#). It requires the learner to read a given text. The reference text is provided in advance.
- Speaking: This scenario is designed for [unscripted assessment](#). It requires the learner to speak on a given topic. The reference text isn't provided in advance.
- Gaming: This scenario is designed for [scripted assessment](#). It requires the learners to read a twister to receive scores for pronunciation and for each syllable. The reference text is provided in advance.

Conduct a reading assessment

Follow these steps to assess your pronunciation of the reference text:

1. Go to **Pronunciation assessment** in the [Azure AI Foundry portal](#).

Try out speech capabilities

Voice gallery

Browse expressive voices with humanlike speech to find the perfect speaker for your project.

[Try demo](#)

Real-time speech to text

Quickly test live transcription capabilities on your own audio without writing any code.

[Try demo](#)

Pronunciation assessment

Evaluate pronunciation and give speakers feedback on the accuracy and fluency of their speech. Language learners can practice and get instant feedback on their pronunciation.

[Try demo](#)

Fast transcription

Quickly test your audios by leveraging advanced speech recognition technology for rapid analysis and recognition.

[Try demo](#)

Other speech capabilities

View additional speech capabilities in the Speech Studio.

[View more capabilities](#)

2. On the Reading tab, choose a supported [language](#) that you want to evaluate the pronunciation.

Configure



Pronunciation mode

Reading

Read a prepared script to receive pronunciation scores.

Speaking

Talk about a chosen topic and receive scores for pronunciation and content.

Gaming

Read a tongue twister to receive scores for pronunciation and for each syllable.

Language to assess

[English \(United States\)](#)[Show advanced options](#)[Sample 1](#) [Sample 2](#) [Sample 3](#) [...](#)

Today was a beautiful day. We had a great time taking a long walk outside in the morning. The countryside was in full bloom, yet the air was crisp and cold. Towards the end of the day, clouds came in, forecasting much needed rain.

Drag and drop audio file(s) here, [browse files](#),

or

[record audio with a microphone](#)

00:00

Record

3. You can use provisioned text samples or enter your own script.

When reading the text, you should be close to microphone to make sure the recorded voice isn't too low.

The screenshot shows a user interface for configuring pronunciation mode. On the left, under 'Pronunciation mode', 'Reading' is selected (indicated by a blue circle). It includes a description: 'Read a prepared script to receive pronunciation scores.' Below it are two other options: 'Speaking' (described as talking about a chosen topic) and 'Gaming' (described as reading a tongue twister). Under 'Language to assess', 'English (United States)' is chosen. A 'Show advanced options' button is also present. On the right, there's a preview section titled 'Sample 1' showing a sample text: 'Today was a beautiful day. We had a great time taking a long walk outside in the morning. The countryside was in full bloom, yet the air was crisp and cold. Towards the end of the day, clouds came in, forecasting much needed rain.' Below this is a dashed box containing instructions to drag and drop files or record audio, with a 'Record' button at the bottom right. The 'Record' button has a microphone icon and the number '0'.

Otherwise you can upload recorded audio for pronunciation assessment. Once successfully uploaded, the audio is automatically evaluated by the system, as shown in the following screenshot.

Configure

Pronunciation mode

Reading
Read a prepared script to receive pronunciation scores.

Speaking
Talk about a chosen topic and receive scores for pronunciation and content.

Gaming
Read a tongue twister to receive scores for pronunciation and for each syllable.

Language to assess

English (United States) ▾

Show advanced options ▾

Sample 1 Sample 2 Sample 3 ...

Today was a beautiful day. We had a great time taking a long walk outside in the morning. The countryside was in full bloom, yet the air was crisp and cold. Towards the end of the day, clouds came in, forecasting much needed rain.

Drag and drop audio file(s) here, [browse files](#), or [record audio with a microphone](#)

00:00 

✓ 276cc014-d... upload Successfully!

Conduct a speaking assessment

If you want to conduct an unscripted assessment, select the Speaking tab. This feature allows you to conduct unscripted assessment without providing reference text in advance. Here's how to proceed:

1. Go to [Pronunciation assessment](#) in the [Azure AI Foundry portal](#).
2. On the Speaking tab, choose a supported [language](#) that you want to evaluate the pronunciation.

Configure



Pronunciation mode

Reading

Read a prepared script to receive pronunciation scores.

Speaking

Talk about a chosen topic and receive scores for pronunciation and content.

Gaming

Read a tongue twister to receive scores for pronunciation and for each syllable.

Language to assess

English (United States)

Sample topics

Talk about your day today

+ Add Topic

Show advanced options

Talk about your day today

Record or upload an audio file of your discussion on the topic above.

> 3 sentences

> 15s

> 50 words

Recommended

(Recommended)

Drag and drop audio file(s)
here, [browse files](#), or
[record audio with a
microphone](#)

00:00

Record

Assessment result



00:00



00:00



Errors

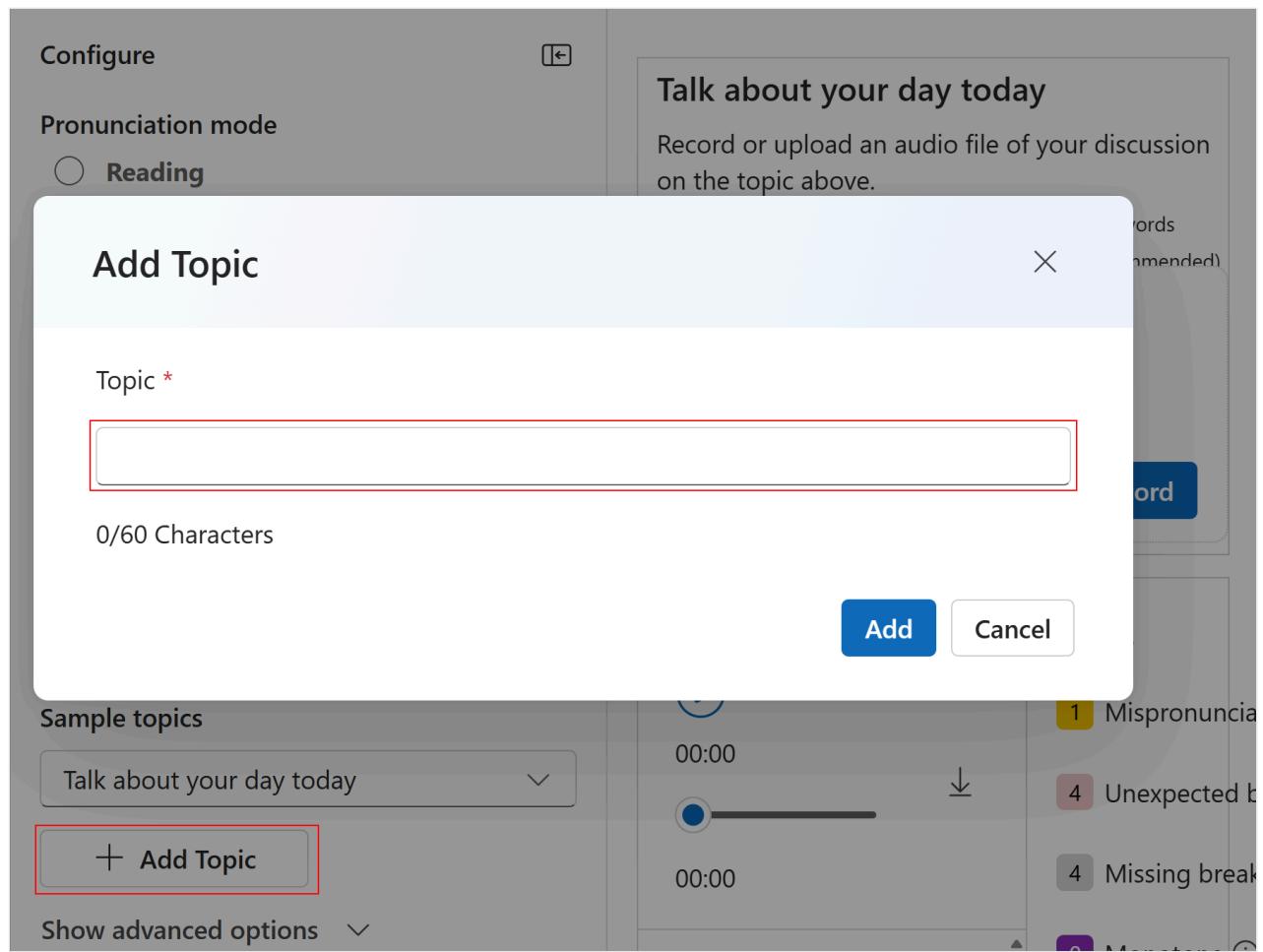
1 Mispronunc

4 Unexpected

4 Missing brea

0 Monotone C

3. Next, you can select from sample topics provided or enter your own topic. This choice allows you to assess your ability to speak on a given subject without a predefined script.



When recording your speech for pronunciation assessment, it's important to ensure that your recording time falls within the recommended range of 15 seconds (equivalent to more than 50 words) to 10 minutes. This time range is optimal for evaluating the content of your speech accurately. To receive a topic score, your spoken audio should contain at least three sentences.

You can also upload recorded audio for pronunciation assessment. Once successfully uploaded, the audio is automatically evaluated by the system.

Conduct a gaming assessment

If you want to practice language learning through a game, follow these steps:

1. Go to [Pronunciation assessment](#) in the [Azure AI Foundry portal](#).
2. On the Gaming tab, choose a supported [language](#) that you want to evaluate the pronunciation and generate new twister you want to practice.

Configure

Pronunciation mode

- Reading**
Read a prepared script to receive pronunciation scores.
- Speaking**
Talk about a chosen topic and receive scores for pronunciation and content.
- Gaming**
Read a tongue twister to receive scores for pronunciation and for each syllable.

Language to assess

English (United States)

Tiny Tim tiptoed through the tulips on tiptoe.

Get ready and start recording as you read the tongue twister aloud.

Generate new twister
 Record

Assessment result

00:00 00:00

Your assessment result will appear here once you record an audio

Errors

- 0 Mispronunciations
- 0 Omissions
- 0 Insertions
- 0 Unexpected break
- 0 Missing break
- 0 Monotone

3. Finally, you can start to record and practice the tongue twister to get the scores.

Configure

Pronunciation mode

- Reading**
Read a prepared script to receive pronunciation scores.
- Speaking**
Talk about a chosen topic and receive scores for pronunciation and content.
- Gaming**
Read a tongue twister to receive scores for pronunciation and for each syllable.

Language to assess

English (United States)

Jenny juggles jars of juicy jumbled jellies and jams.

Get ready and start recording as you read the tongue twister aloud.

Generate new twister
 Record

Assessment result

00:00 00:06

jenny juggles jars of juicy jumbled jellies and
jams.

Errors

- 0 Mispronunciations
- 0 Omissions
- 0 Insertions
- 0 Unexpected break
- 0 Missing break
- 0 Monotone

Pronunciation score

Score breakdown

	Accuracy score	86 / 100
Fluency score	99 / 100	
Completeness score	100 / 100	
Prosody score	81 / 100	

0 ~ 59 60 ~ 79 80 ~ 100

jenny 85 jen 92 ny 84	juggles 82 jug 64 gles 92	jars 88 jars 95	of 94 of 94	juicy 97 jui 91 cy 96	jumbled 73 jum 100 bled 61
jellies 82 jel 76 lies 75	and 88 and 82	jams 85 jams 75			

Pronunciation assessment results

Once you recorded your speech or uploaded the recorded audio, the **Assessment result** is output. The result includes your spoken audio and the feedback on your speech assessment. You can listen to your spoken audio and download it if necessary.

You can also check the pronunciation assessment result in JSON. The word-level, syllable-level, and phoneme-level accuracy scores are included in the JSON file.

Display

Assessment result

00:00 [play] 00:00 ↓

[Sample response]

today was a burst of adventures. i am [red] wake up bright and early to help my dad make pancakes. [red] later we set them [red] on a nature hike. [red] we spotted some **fascinating** birds. [red] after that [red] i joined my friends for a lively virtual dance class [red] and we had an absolute blast, [red] grooving and laughing together.

Errors

- 1 Mispronunciations ⓘ
- 4 Unexpected break ⓘ
- 4 Missing break ⓘ
- 0 Monotone ⓘ

Pronunciation score

Score breakdown

Score Type	Score / 100
Accuracy score ⓘ	88 / 100
Fluency score ⓘ	98 / 100
Prosody score ⓘ	88 / 100

Content score

Score breakdown

Score Type	Score / 100
Vocabulary score ⓘ	54 / 100
Grammar score ⓘ	50 / 100
Topic score ⓘ	76 / 100

The word is highlighted according to the error type. The error types in the pronunciation assessment are represented using different colors. This visual distinction makes it easier to identify and analyze specific errors. It provides a clear overview of the error types and frequencies in the spoken audio, helping you focus on areas that need improvement. You can toggle on/off each error type to focus on specific types of errors or exclude certain types from the display. This feature provides flexibility in how you review and analyze the errors in your spoken audio. While hovering over each word, you can see accuracy scores for the whole word or specific phonemes.

At the bottom of the Assessment result, scoring results are displayed. For scripted pronunciation assessment, only the pronunciation score (including accuracy score, fluency score, completeness score, and prosody score) is provided. For unscripted pronunciation assessment, both pronunciation score (including accuracy score, fluency score, and

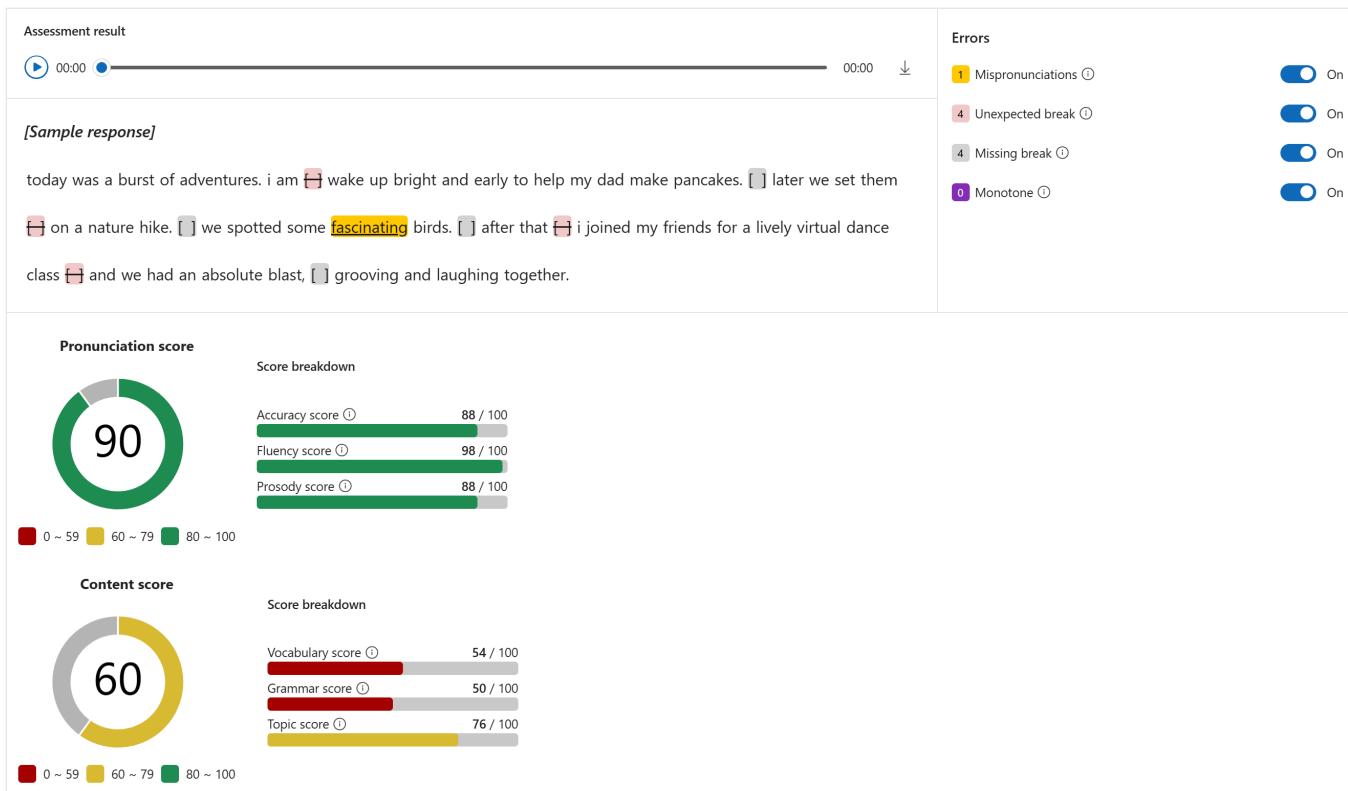
prosody score) and content score (including vocabulary score, grammar score, and topic score) are displayed.

Granularity of pronunciation assessment

Pronunciation assessment provides various assessment results in different granularities, from individual phonemes to the entire text input.

- At the full-text level, pronunciation assessment offers additional Fluency, Completeness, and Prosody scores: Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words; Completeness indicates how many words are correctly pronounced in the speech to the reference text input; Prosody indicates how well a speaker conveys elements of naturalness, expressiveness, and overall prosody in their speech. An overall score aggregated from Accuracy, Fluency, Completeness, and Prosody is then given to indicate the overall pronunciation quality of the given speech. Pronunciation assessment also offers content score (Vocabulary, Grammar, and Topic) at the full-text level.
- At the word level, pronunciation assessment can automatically detect miscues and provide accuracy score simultaneously, which provides more detailed information on omission, repetition, insertions, and mispronunciation in the given speech.
- Syllable-level accuracy scores are currently available via the [JSON file](#) or [Speech SDK](#).
- At the phoneme level, pronunciation assessment provides accuracy scores of each phoneme, helping learners to better understand the pronunciation details of their speech.

In addition to the baseline scores of accuracy, fluency, and completeness, the pronunciation assessment feature in the Azure AI Foundry includes more comprehensive scores to provide detailed feedback on various aspects of speech performance and understanding. The enhanced scores are as follows: Prosody score, Vocabulary score, Grammar score, and Topic score. These scores offer valuable insights into speech prosody, vocabulary usage, grammar correctness, and topic understanding.



At the bottom of the Assessment result, two overall scores are displayed: Pronunciation score and Content score. In the Reading tab, you find the Pronunciation score displayed. In the Speaking tab, both the Pronunciation score and the Content score are displayed.

Pronunciation score: This score represents an aggregated assessment of the pronunciation quality and includes four subaspects. These scores are available in both the reading and speaking tabs for both scripted and unscripted assessments.

- **Accuracy score:** Evaluates the correctness of pronunciation.
- **Fluency score:** Measures the level of smoothness and naturalness in speech.
- **Completeness score:** Reflects the number of words pronounced correctly.
- **Prosody score:** Assesses the use of appropriate intonation, rhythm, and stress. Several more error types related to prosody assessment are introduced, such as Unexpected break, Missing break, and Monotone. These error types provide more detailed information about pronunciation errors compared to the previous engine.

Content score: This score provides an aggregated assessment of the content of the speech and includes three subaspects. This score is only available in the speaking tab for an unscripted assessment.

- **Vocabulary score:** Evaluates the speaker's effective usage of words and their appropriateness within the given context to express ideas accurately, and the level of lexical complexity.
- **Grammar score:** Evaluates the correctness of grammar usage and variety of sentence patterns. It considers lexical accuracy, grammatical accuracy, and diversity of sentence structures, providing a more comprehensive evaluation of language proficiency.

- **Topic score:** Assesses the level of understanding and engagement with the topic discussed in the speech. It evaluates the speaker's ability to effectively express thoughts and ideas related to the given topic.

These overall scores offer a comprehensive assessment of both pronunciation and content, providing learners with valuable feedback on various aspects of their speech performance and understanding. With these enhanced features, language learners can gain deeper insights into their advantages and areas for improvement in both pronunciation and content expression.

 **Note**

Content and prosody assessments are only available in the [en-US](#) locale.

Assessment scores in streaming mode

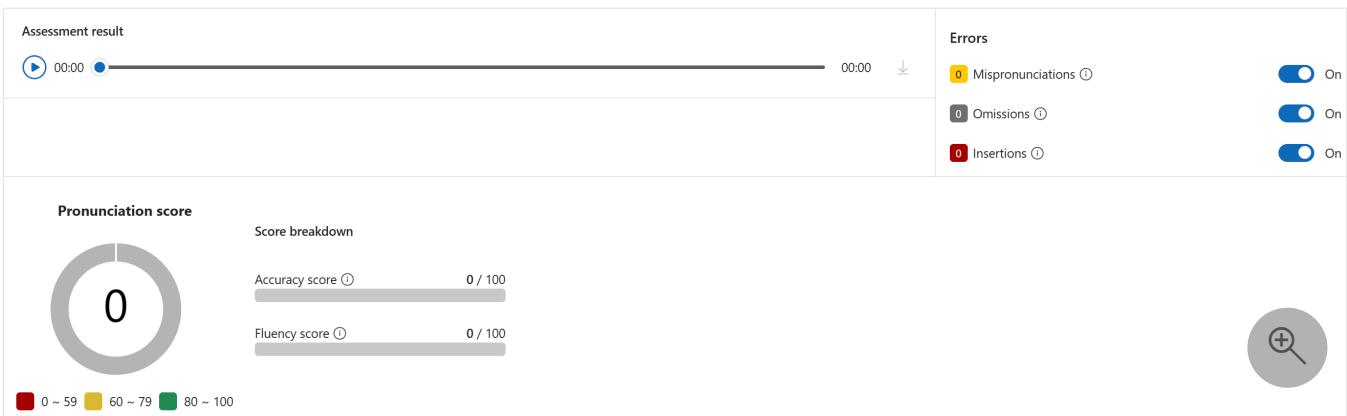
Pronunciation assessment supports uninterrupted streaming mode. The Azure AI Foundry demo allows for up to 60 minutes of recording in streaming mode for evaluation. As long as you don't press the stop recording button, the evaluation process doesn't finish and you can pause and resume evaluation conveniently.

Pronunciation assessment evaluates several aspects of pronunciation. At the bottom of **Assessment result**, you can see **Pronunciation score** as aggregated overall score, which includes 4 sub aspects: **Accuracy score**, **Fluency score**, **Completeness score**, and **Prosody score**. In streaming mode, since the **Accuracy score**, **Fluency score**, and **Prosody score** will vary over time throughout the recording process, we demonstrate an approach in Azure AI Foundry to display approximate overall score incrementally before the end of the evaluation, which weighted only with Accuracy score, Fluency score, and Prosody score. The **Completeness score** is only calculated at the end of the evaluation after you press the stop button, so the final pronunciation overall score is aggregated from **Accuracy score**, **Fluency score**, **Completeness score**, and **Prosody score** with weight.

Refer to the demo examples below for the whole process of evaluating pronunciation in streaming mode.

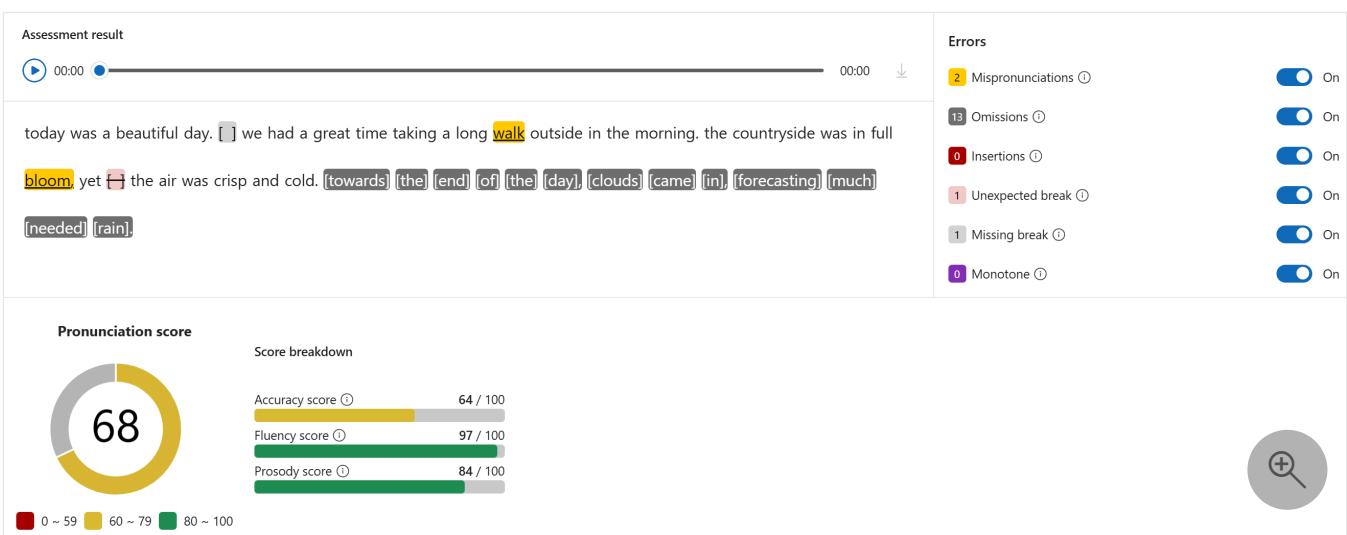
Start recording

As you start recording, the scores at the bottom begin to alter from 0.



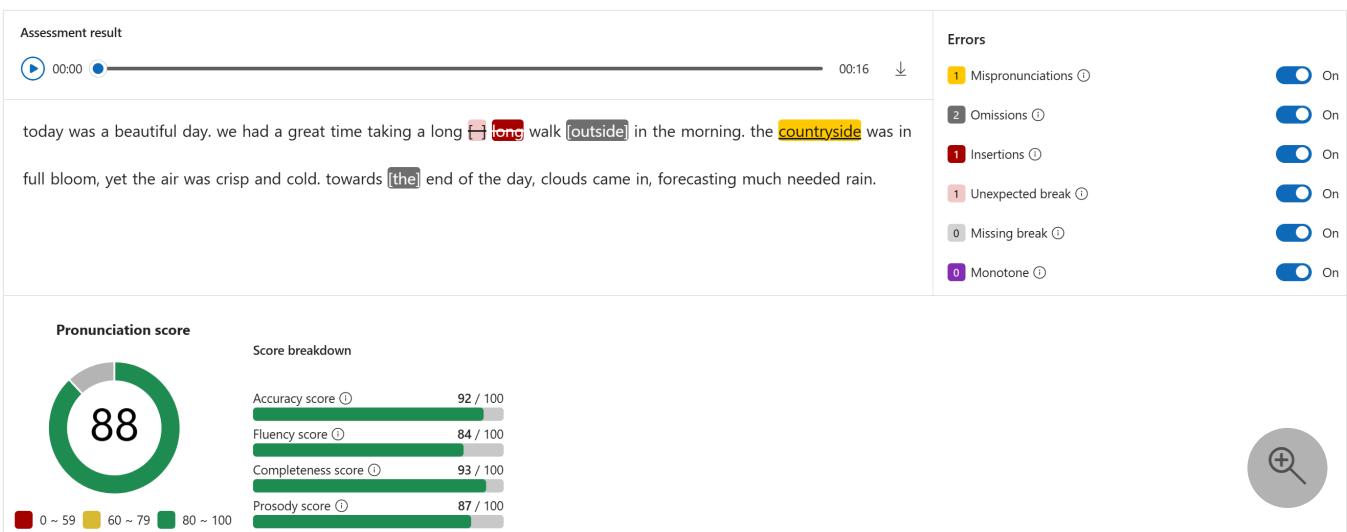
During recording

During recording a long paragraph, you can pause recording at any time. You can continue to evaluate your recording as long as you don't press the stop button.



Finish recording

After you press the stop button, you can see **Pronunciation score**, **Accuracy score**, **Fluency score**, **Completeness score**, and **Prosody score** at the bottom.



Pricing

As a baseline, usage of pronunciation assessment costs the same as speech to text for Standard or commitment tier [pricing ↗](#). If you [purchase a commitment tier](#) for speech to text, the spend for pronunciation assessment goes towards meeting the commitment.

The pronunciation assessment feature also offers other scores that aren't included in the baseline speech to text price: prosody, grammar, topic, and vocabulary. These scores are available as an add-on charge above the baseline speech to text price. For information about pricing, see [speech to text pricing ↗](#).

Here's a table of available pronunciation assessment scores, whether it's available in the [scripted](#) or [unscripted](#) assessments, and whether it's included in the baseline speech to text price or the add-on price.

 Expand table

Score	Scripted or unscripted	Included in baseline speech to text price?
Accuracy	Scripted and unscripted	Yes
Fluency	Scripted and unscripted	Yes
Completeness	Scripted	Yes
Misue	Scripted and unscripted	Yes
Prosody	Scripted and unscripted	No
Grammar	Unscripted only	No
Topic	Unscripted only	No
Vocabulary	Unscripted only	No

Responsible AI

An AI system includes not only the technology, but also the people who use it, the people who will be affected by it, and the environment in which it's deployed. Read the transparency notes to learn about responsible AI use and deployment in your systems.

- [Transparency note and use cases](#)
- [Characteristics and limitations](#)

Next steps

- Use pronunciation assessment with the Speech SDK
- Read the blog about [use cases ↗](#)

Interactive language learning with pronunciation assessment

08/07/2025

ⓘ Important

Items marked (preview) in this article are currently in public preview. This preview is provided without a service-level agreement, and we don't recommend it for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

Learning a new language is an exciting journey. Interactive language learning can make your learning experience more engaging and effective. By using pronunciation assessment effectively, you get instant feedback on pronunciation accuracy, fluency, prosody, grammar, and vocabulary through your interactive language learning experience.

ⓘ Note

The language learning feature currently supports only `en-us`. For available regions, refer to [available regions for pronunciation assessment](#). If you turn on the **Avatar** button to interact with a text to speech avatar, refer to the available [regions](#) for text to speech avatar.

If you have any feedback on the language learning feature, fill out [this form](#).

Common use cases

Here are some common scenarios where you can make use of the language learning feature to improve your language skills:

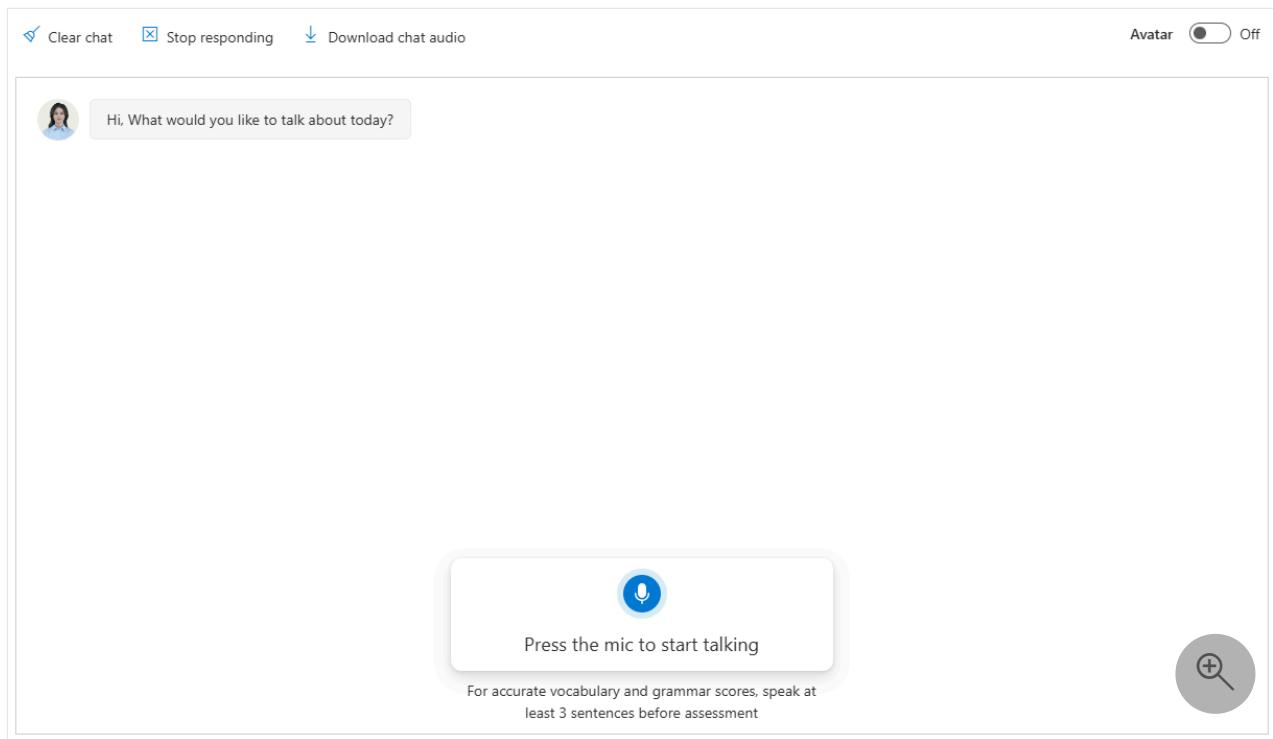
- **Assess pronunciations:** Practice your pronunciation and receive scores with detailed feedback to identify areas for improvement.
- **Improve speaking skills:** Engage in conversations with a native speaker (or a simulated one) to enhance your speaking skills and build confidence.
- **Learn new vocabulary:** Expand your vocabulary and work on advanced pronunciation by interacting with AI-driven language models.

Getting started

In this section, you can learn how to immerse yourself in dynamic conversations with a GPT-powered voice assistant to enhance your speaking skills.

To get started with language learning through chatting, follow these steps:

1. Go to [Language learning](#) in the [Speech Studio](#).
2. Decide on a scenario or context in which you'd like to interact with the voice assistant.
This can be a casual conversation, a specific topic, or a language learning exercise.



If you want to interact with an avatar, toggle the **Avatar** button in the upper right corner to **On**.

3. Press the microphone icon to start speaking naturally, as if you were talking to a real person.

Clear chat Stop responding Download chat audio Avatar Off

Hi, What would you like to talk about today?

Press the mic to start talking

For accurate vocabulary and grammar scores, speak at least 3 sentences before assessment

BZ

For accurate vocabulary and grammar scores, speak at least 3 sentences before assessment.

4. Press the stop button or **Assess my response** button to finish speaking. This action will trigger the assessment process.

Clear chat Stop responding Download chat audio Avatar Off

Hi, What would you like to talk about today?

Let's talk about online shopping. BZ

Sure! What aspect of online shopping interests you most? For example, benefits, security, or choosing the best platforms?

I would like to talk about the benefit of online shopping.

Online shopping offers convenience, a broader selection, often better prices, and the ability to compare products and reviews from the comfort of home.

Assess my response

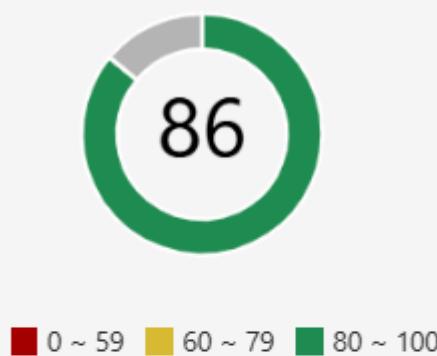
Recording... stop talking to send

For accurate vocabulary and grammar scores, speak at least 3 sentences before assessment

BZ

5. Wait for a moment, and you can get a detailed assessment report.

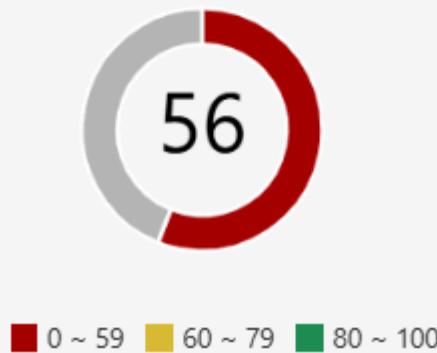
Pronunciation score ⓘ



Score breakdown

Accuracy Score ⓘ	87 / 100
Fluency Score ⓘ	92 / 100
Prosody Score ⓘ	83 / 100

Content score ⓘ



Score breakdown

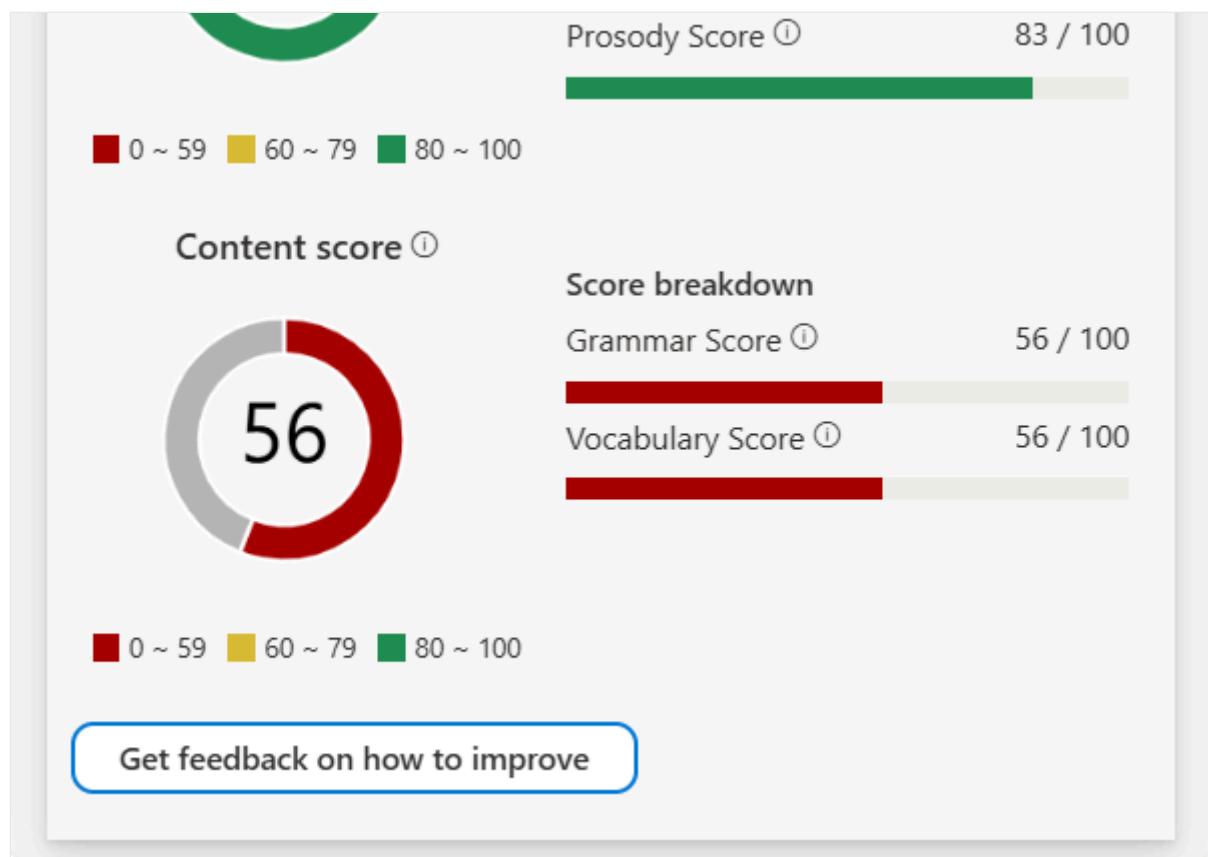
Grammar Score ⓘ	56 / 100
Vocabulary Score ⓘ	56 / 100

The assessment report may include feedback on:

- **Accuracy:** Accuracy indicates how closely the phonemes match a native speaker's pronunciation.
- **Fluency:** Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words.
- **Prosody:** Prosody indicates the nature of the given speech, including stress, intonation, speaking speed, and rhythm.
- **Grammar:** Grammar considers lexical accuracy, grammatical accuracy, and diversity of sentence structures, providing a more comprehensive evaluation of language proficiency.
- **Vocabulary:** Vocabulary evaluates the speaker's effective usage of words and their appropriateness within the given context to express ideas accurately, as well as the level of lexical complexity.

When recording your speech for pronunciation assessment, ensure your recording time falls within the recommended range of 20 seconds (equivalent to more than 50 words) to 10 minutes per session. This time range is optimal for evaluating the content of your speech accurately. Whether you have a short and focused conversation or a more extended dialogue, as long as the total recorded time falls within this range, you'll receive comprehensive feedback on your pronunciation, fluency, and content.

To get feedback on how to improve for each aspect of the assessment, select **Get feedback on how to improve**.



When you have completed the conversation, you can also download your chat audio. You can clear the current conversation by selecting **Clear chat**.

Next steps

- Use [pronunciation assessment with the Speech SDK](#)
- Try [pronunciation assessment in the studio](#).
- Read our [pronunciation assessment blog ↗](#) to learn more speech scenarios.

Azure OpenAI speech to speech chat

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this how-to guide, you can use Azure AI Speech to converse with Azure OpenAI in Azure AI Foundry Models. The text recognized by the Speech service is sent to Azure OpenAI. The Speech service synthesizes speech from the text response from Azure OpenAI.

Speak into the microphone to start a conversation with Azure OpenAI.

- The Speech service recognizes your speech and converts it into text (speech to text).
- Your request as text is sent to Azure OpenAI.
- The Speech service text to speech feature synthesizes the response from Azure OpenAI to the default speaker.

Although the experience of this example is a back-and-forth exchange, Azure OpenAI doesn't remember the context of your conversation.

Prerequisites

- ✓ Azure subscription - [Create one for free](#) ↗
- ✓ [Create a Microsoft Azure OpenAI in Azure AI Foundry Models 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.
- ✓ [Create an AI Foundry resource for Speech](#) ↗ 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.

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 `AZURE_OPENAI_API_KEY`, `AZURE_OPENAI_ENDPOINT`, `AZURE_OPENAI_CHAT_DEPLOYMENT`, `SPEECH_KEY`, and `SPEECH_REGION`.

Your application must be authenticated to access Azure AI Foundry resources. This article shows you how to use environment variables to store your credentials. You can then access the environment variables from your code to authenticate your application. For production, use a more secure way to store and access your credentials.

Important

We recommend Microsoft Entra ID authentication with [managed identities for Azure resources](#) to avoid storing credentials with your applications that run in the cloud.

Use API keys with caution. Don't include the API key directly in your code, and never post it publicly. If using API keys, store them securely in Azure Key Vault, rotate the keys regularly, and restrict access to Azure Key Vault using role based access control and network access restrictions. For more information about using API keys securely in your apps, see [API keys with Azure Key Vault](#).

For more information about AI services security, see [Authenticate requests to Azure AI services](#).

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

- To set the `AZURE_OPENAI_API_KEY` environment variable, replace `your-openai-key` with one of the keys for your resource.
- To set the `AZURE_OPENAI_ENDPOINT` environment variable, replace `your-openai-endpoint` with one of the regions for your resource.
- To set the `AZURE_OPENAI_CHAT_DEPLOYMENT` 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 AZURE_OPENAI_API_KEY your-openai-key  
setx AZURE_OPENAI_ENDPOINT your-openai-endpoint
```

```
setx AZURE_OPENAI_CHAT_DEPLOYMENT 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 "AZURE_OPENAI_API_KEY",
// "AZURE_OPENAI_ENDPOINT" and "AZURE_OPENAI_CHAT_DEPLOYMENT"
// Your endpoint should look like the following
https://YOUR_OPEN_AI_RESOURCE_NAME.openai.azure.com/
string openAIKey = Environment.GetEnvironmentVariable("AZURE_OPENAI_API_KEY")
??
    throw new ArgumentException("Missing
AZURE_OPENAI_API_KEY");
string openAIEndpoint =
Environment.GetEnvironmentVariable("AZURE_OPENAI_ENDPOINT") ??
    throw new ArgumentException("Missing
AZURE_OPENAI_ENDPOINT");

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

// 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 ChatWithAzureOpenAI();
}
catch (Exception ex)
{
    Console.WriteLine(ex);
}

// Prompts Azure OpenAI with a request and synthesizes the response.
async Task AskAzureOpenAI(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.WriteLine($"{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 ChatWithAzureOpenAI()
{
    // 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($"Recognized speech:
{speechRecognitionResult.Text}");
                    await AskAzureOpenAI(speechRecognitionResult.Text);
                }
                break;
            case ResultReason.NoMatch:
                Console.WriteLine($"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. For more information tokens and cost implications, see [Azure OpenAI tokens](#) and [Azure OpenAI pricing](#).

6. Run your new console application to start speech recognition from a microphone:

```
Console
```

```
dotnet run
```

Important

Make sure that you set the `AZURE_OPENAI_API_KEY`, `AZURE_OPENAI_ENDPOINT`, `AZURE_OPENAI_CHAT_DEPLOYMENT`, `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.

```
Console
```

```
PS C:\dev\openai\csharp> dotnet run  
Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.  
Recognized speech: Make a comma separated list of all continents.  
Azure OpenAI response: Africa, Antarctica, Asia, Australia, Europe, North America,  
South America  
Speech synthesized to speaker for text [Africa, Antarctica, Asia, Australia,  
Europe, North America, South America]  
Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.  
Recognized speech: Make a comma separated list of 1 Astronomical observatory for  
each continent. A list should include each continent name in parentheses.  
Azure OpenAI response: Mauna Kea Observatories (North America), La Silla  
Observatory (South America), Tenerife Observatory (Europe), Siding Spring  
Observatory (Australia), Beijing Xinglong Observatory (Asia), Naukluft Plateau  
Observatory (Africa), Rutherford Appleton Laboratory (Antarctica)  
Speech synthesized to speaker for text [Mauna Kea Observatories (North America),  
La Silla Observatory (South America), Tenerife Observatory (Europe), Siding Spring  
Observatory (Australia), Beijing Xinglong Observatory (Asia), Naukluft Plateau  
Observatory (Africa), Rutherford Appleton Laboratory (Antarctica)]  
Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.
```

```
Conversation ended.  
PS C:\dev\openai\csharp>
```

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`. For details about how to identify one of multiple languages that might be spoken, see [language identification](#).
- 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 reduce latency for text to speech output, use the text streaming feature, which enables real-time text processing for fast audio generation and minimizes latency, enhancing the fluidity and responsiveness of real-time audio outputs. Refer to [how to use text streaming](#).
- To enable [TTS Avatar](#) as a visual experience of speech output, refer to [real-time synthesis for text to speech avatar](#) and [sample code](#) for chat scenario with avatar.
- 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. For more information, see the [content filtering](#) article.

Clean up resources

You can use the [Azure portal](#) or [Azure Command Line Interface \(CLI\)](#) to remove the Speech resource you created.

Related content

- [Learn more about Speech](#)
- [Learn more about Azure OpenAI](#)

What is multi-device conversation?

08/07/2025

Multi-device conversation makes it easy to create a speech or text conversation between multiple clients and coordinate the messages sent between them.

(!) Note

Multi-device conversation access is a preview feature.

With multi-device conversation, you can:

- Connect multiple clients into the same conversation and manage the sending and receiving of messages between them.
- Easily transcribe audio from each client and send the transcription to the others, with optional translation.
- Easily send text messages between clients, with optional translation.

You can build a feature or solution that works across an array of devices. Each device can independently send messages (either transcriptions of audio or instant messages) to all other devices.

Multi-device Conversation is suited for scenarios with multiple devices, each with a single microphone.

(i) Important

Multi-device conversation does not support sending audio files between clients: only the transcription and/or translation.

Key features

- **Real-time transcription:** Everyone receives a transcript of the conversation, so they can follow along the text in real-time or save it for later.
- **Real-time translation:** With more than 70 [supported languages](#) for text translation, users can translate the conversation to their preferred languages.
- **Readable transcripts:** The transcription and translation are easy to follow, with punctuation and sentence breaks.
- **Voice or text input:** Each user can speak or type on their own device, depending on the language support capabilities enabled for the participant's chosen language. Refer to

Language support.

- **Message relay:** The multi-device conversation service distributes messages sent by one client to all the others, in the languages of their choice.
- **Message identification:** Every message that users receive in the conversation is tagged with the nickname of the user who sent it.

Use cases

Lightweight conversations

Creating and joining a conversation is easy. One user acts as the 'host' and creates a conversation, which generates a random five letter conversation code and a QR code. All other users can join the conversation by typing in the conversation code or scanning the QR code.

Since users join via the conversation code and aren't required to share contact information, it's easy to create quick, on-the-spot conversations.

Inclusive meetings

Real-time transcription and translation can help make conversations accessible for people who speak different languages and/or are deaf or hard of hearing. Each person can also actively participate in the conversation, by speaking their preferred language or sending instant messages.

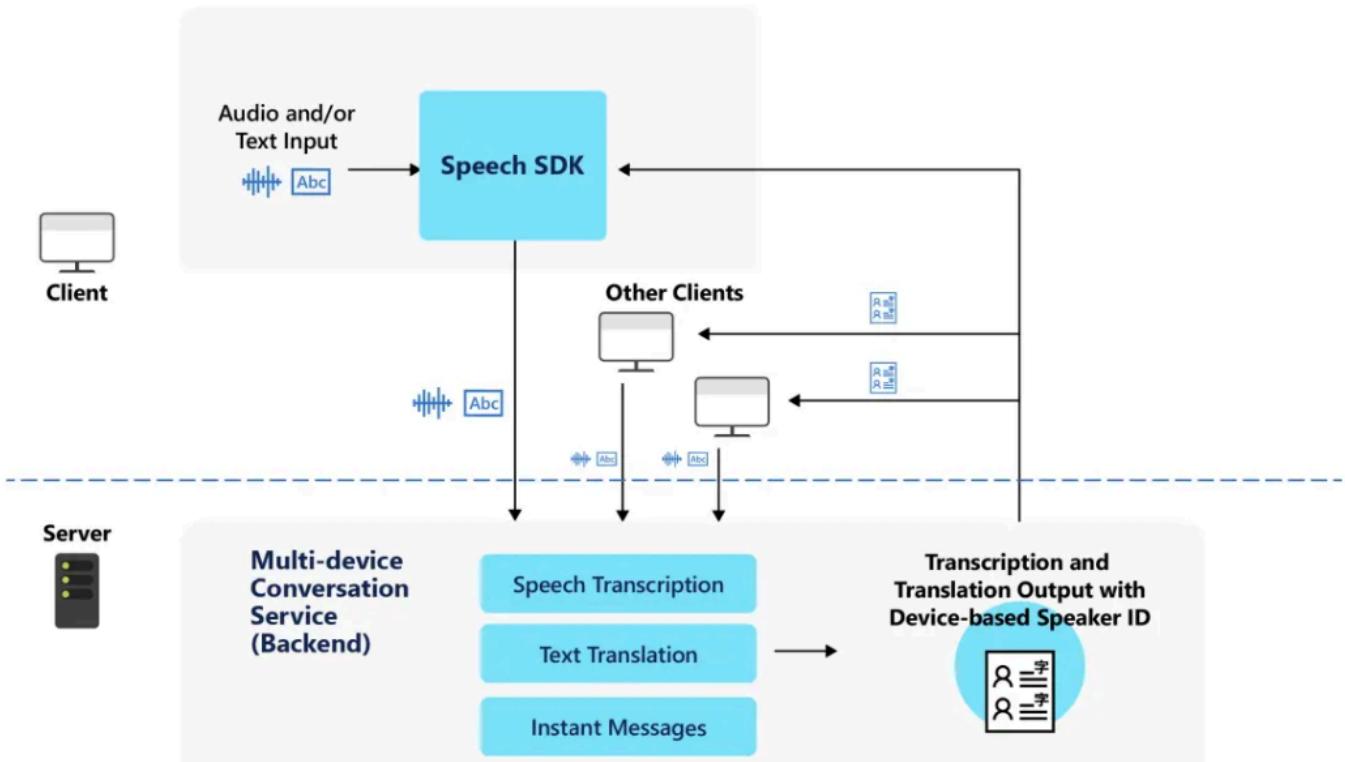
Presentations

You can also provide captions for presentations and lectures both on-screen and on the audience members' own devices. After the audience joins with the conversation code, they can see the transcript in their preferred language, on their own device.

How it works

All clients use the Speech SDK to create or join a conversation. The Speech SDK interacts with the multi-device conversation service, which manages the lifetime of a conversation. The conversation includes the list of participants, each client's chosen language, and the messages sent.

Each client can send audio or instant messages. The service uses speech recognition to convert audio into text, and send instant messages as-is. If clients choose different languages, then the service translates all messages to the specified language(s) of each client.



Overview of Conversation, Host, and Participant

A conversation is a session that one user starts for the other participating users to join. All clients connect to the conversation using the five-letter conversation code.

Each conversation creates metadata that includes:

- Timestamps of when the conversation started and ended
- List of all participants in the conversation, which includes each user's chosen nickname and primary language for speech or text input.

There are two types of users in a conversation: host and participant.

The host is the user who starts a conversation, and who acts as the administrator of that conversation.

- Each conversation can only have one host
- The host must be connected to the conversation during the conversation. If the host leaves the conversation, the conversation ends for all other participants.
- The host has a few extra controls to manage the conversation:
 - Lock the conversation - prevent more participants from joining
 - Mute all participants - prevent other participants from sending any messages to the conversation, whether transcribed from speech or instant messages
 - Mute individual participants
 - Unmute all participants

- Unmute individual participants

A participant is a user who joins a conversation.

- A participant can leave and rejoin the same conversation at any time, without ending the conversation for other participants.
- Participants can't lock the conversation or mute/unmute others

 **Note**

Each conversation can have up to 100 participants, of which 10 can be simultaneously speaking at any given time.

Language support

Each user must choose a primary language when they join a conversation. Their selection is the language they speak and send instant messages in, and also the language they see other users' messages.

There are two kinds of languages: speech to text and text-only:

- If the user chooses a speech to text language as their primary language, then they can use both speech and text input in the conversation.
- If the user chooses a text-only language, then they can only use text input and send instant messages in the conversation. Text-only languages are the languages that are supported for text translation, but not speech to text. You can see available languages on the [language support](#) page.

Apart from their primary language, each participant can also specify more languages for translating the conversation.

The following table is a summary of what the user can do in a multi-device conversation, based to their chosen primary language.

 [Expand table](#)

What the user can do in the conversation	Speech to text	Text-only
Use speech input	✓	✗
Send instant messages	✓	✓
Translate the conversation	✓	✓

 Note

For lists of available speech to text and text translation languages, see [supported languages](#).

Next steps

- [Translate conversations in real-time](#)

Quickstart: Multi-device Conversation

07/12/2025

In this quickstart, you'll learn how to use the [Speech SDK](#) to create a new multi-device conversation with translation support, as well as join an existing conversation.

ⓘ Note

The Speech SDK for Java, JavaScript, Objective-C, and Swift support multi-device conversation, but we haven't yet included a guide here.

You can view or download all [Speech SDK C# Samples](#) on GitHub.

Prerequisites

Before you get started, make sure to:

- ✓ [Create an AI Foundry resource for Speech](#)
- ✓ [Setup your development environment and create an empty project](#)

Add sample code

1. Open `Program.cs`, and replace all code in it with the following code:

C#

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using Microsoft.CognitiveServices.Speech.Transcription;
using System;
using System.Threading.Tasks;

namespace HelloWorld
{
    class Program
    {
        static async Task Main(string[] args)
        {
            await CreateConversationAsync();
        }

        static async Task CreateConversationAsync()
        {
            // Replace these values with the details of your Cognitive Speech
            subscription
```

```
        string subscriptionKey = "YourSpeechResourceKey";

        // Replace below with your region identifier from here:
https://aka.ms/speech/sdkregion
        string region = "YourServiceRegion";

        // Sets source and target languages.
        // Replace with the languages of your choice, from list found
        here: https://aka.ms/speech/sttt-languages
        string fromLanguage = "en-US";
        string toLanguage = "de";

        // Set this to the display name you want for the conversation
        host
        string displayName = "The host";

        // Create the task completion source that will be used to wait
        until the user presses Ctrl + C
        var completionSource = new TaskCompletionSource<bool>();

        // Register to listen for Ctrl + C
        Console.CancelKeyPress += (s, e) =>
        {
            completionSource.TrySetResult(true);
            e.Cancel = true; // don't terminate the current process
        };

        // Create an instance of the speech translation config
        var config =
SpeechTranslationConfig.FromSubscription(subscriptionKey, region);
        config.SpeechRecognitionLanguage = fromLanguage;
        config.AddTargetLanguage(toLanguage);

        // Create the conversation
        using (var conversation = await
Conversation.CreateConversationAsync(config).ConfigureAwait(false))
        {
            // Start the conversation so the host user and others can
            join
            await
conversation.StartConversationAsync().ConfigureAwait(false);

            // Get the conversation ID. It will be up to your scenario to
            determine how this is shared with other participants.
            string conversationId = conversation.ConversationId;
            Console.WriteLine($"Created '{conversationId}'"
conversation");

            // At this point, you can use the conversation object to
            manage the conversation.
            // For example, to mute everyone else in the room you can
            call this method:
            await
conversation.MuteAllParticipantsAsync().ConfigureAwait(false);
```

```

        // Configure which audio source you want to use. If you are
        // using a text only language, you
        // can use the other overload of the constructor that takes
        // no arguments
        var audioConfig = AudioConfig.FromDefaultMicrophoneInput();
        using (var conversationTranslator = new
ConversationTranslator(audioConfig))
        {
            // You should connect all the event handlers you need at
this point
            conversationTranslator.SessionStarted += (s, e) =>
            {
                Console.WriteLine($"Session started: {e.SessionId}");
            };
            conversationTranslator.SessionStopped += (s, e) =>
            {
                Console.WriteLine($"Session stopped: {e.SessionId}");
            };
            conversationTranslator.Canceled += (s, e) =>
            {
                switch (e.Reason)
                {
                    case CancellationReason.EndOfStream:
                        Console.WriteLine($"End of audio reached");
                        break;

                    case CancellationReason.Error:
                        Console.WriteLine($"Canceled due to error.
{e.ErrorCode}: {e.ErrorDetails}");
                        break;
                }
            };
            conversationTranslator.ConversationExpiration += (s, e)
=>
            {
                Console.WriteLine($"Conversation will expire in
{e.ExpirationTime.TotalMinutes} minutes");
            };
            conversationTranslator.ParticipantsChanged += (s, e) =>
            {
                Console.Write("The following participant(s) have ");
                switch (e.Reason)
                {
                    case ParticipantChangedReason.JoinedConversation:
                        Console.Write("joined");
                        break;

                    case ParticipantChangedReason.LeftConversation:
                        Console.Write("left");
                        break;

                    case ParticipantChangedReason.Updated:
                        Console.Write("been updated");
                        break;
                }
            };
        }
    }
}

```

```

        Console.WriteLine(":");

        foreach (var participant in e.Participants)
        {

Console.WriteLine($"\\t{participant.DisplayName}");
        }
    };

conversationTranslator.TextMessageReceived += (s, e) =>
{
    Console.WriteLine($"Received an instant message from
'{e.Result.ParticipantId}': '{e.Result.Text}'");
    foreach (var entry in e.Result.Translations)
    {
        Console.WriteLine($"\\tTranslated into
'{entry.Key}': '{entry.Value}'");
    }
};

conversationTranslator.Transcribed += (s, e) =>
{
    Console.WriteLine($"Received a transcription from
'{e.Result.ParticipantId}': '{e.Result.Text}'");
    foreach (var entry in e.Result.Translations)
    {
        Console.WriteLine($"\\tTranslated into
'{entry.Key}': '{entry.Value}'");
    }
};

conversationTranslator.Transcribing += (s, e) =>
{
    Console.WriteLine($"Received a partial transcription
from '{e.Result.ParticipantId}': '{e.Result.Text}'");
    foreach (var entry in e.Result.Translations)
    {
        Console.WriteLine($"\\tTranslated into
'{entry.Key}': '{entry.Value}'");
    }
};

// Enter the conversation to start receiving events
await
conversationTranslator.JoinConversationAsync(conversation,
displayName).ConfigureAwait(false);

// You can now send an instant message to all other
participants in the room
await conversationTranslator.SendTextMessageAsync("The
instant message to send").ConfigureAwait(false);

// If specified a speech to text language, you can start
capturing audio
await
conversationTranslator.StartTranscribingAsync().ConfigureAwait(false);
Console.WriteLine("Started transcribing. Press Ctrl + c

```

```
        // At this point, you should start receiving
transcriptions for what you are saying using the default microphone. Press
Ctrl+c to stop audio capture
        await completionSource.Task.ConfigureAwait(false);

        // Stop audio capture
        await
conversationTranslator.StopTranscribingAsync().ConfigureAwait(false);

        // Leave the conversation. After this you will no longer
receive events
        await
conversationTranslator.LeaveConversationAsync().ConfigureAwait(false);
    }

        // End the conversation
        await
conversation.EndConversationAsync().ConfigureAwait(false);

        // Delete the conversation. Any other participants that are
still in the conversation will be removed
        await
conversation.DeleteConversationAsync().ConfigureAwait(false);
    }
}
```

2. In the same file, replace the string `YourSpeechResourceKey` with your Cognitive Speech subscription key.
 3. Replace the string `YourServiceRegion` with the `region` associated with your Speech resource.
 4. From the menu bar, choose **File > Save All**.

Build and run the application to create a new conversation

1. From the menu bar, select **Build > Build Solution** to build the application. The code should compile without errors now.
 2. Choose **Debug > Start Debugging** (or press **F5**) to start the **helloworld** application.
 3. Once you see the `Started transcribing` message appear, you can start speaking. You'll see the transcriptions appear as you speak.

- If you share the conversation code with the others and they join the conversation, you'll see their transcriptions as well.

4. Once you're done speaking, press `Ctrl+C` to stop audio capture, and end the conversation.

Build and run the application to join an existing conversation

1. Copy and paste the following function into your `Program.cs`:

```
C#  
  
static async Task JoinConversationAsync(string conversationId)  
{  
    // Set this to the display name you want for the participant  
    string displayName = "participant";  
  
    // Set the speech to text, or text language you want to use  
    string language = "en-US";  
  
    // Create the task completion source that will be used to wait until the  
    user presses Ctrl + c  
    var completionSource = new TaskCompletionSource<bool>();  
  
    // Register to listen for Ctrl+C  
    Console.CancelKeyPress += (s, e) =>  
    {  
        completionSource.TrySetResult(true);  
        e.Cancel = true; // don't terminate the current process  
    };  
  
    // As a participant, you don't need to specify any subscription key, or  
    region. You can directly create  
    // the conversation translator object  
    var audioConfig = AudioConfig.FromDefaultMicrophoneInput();  
    using (var conversationTranslator = new  
ConversationTranslator(audioConfig))  
    {  
        // Register for any events you are interested here. For now let's  
        just register for  
        // transcription, and instant message events  
        conversationTranslator.TextMessageReceived += (s, e) =>  
        {  
            Console.WriteLine($"Received an instant message from  
'{e.Result.ParticipantId}': '{e.Result.Text}'");  
            foreach (var entry in e.Result.Translations)  
            {  
                Console.WriteLine($"{entry.Key}: '{entry.Value}'");  
            }  
        };  
    }  
}
```

```

        }
    };
    conversationTranslator.Transcribed += (s, e) =>
    {
        Console.WriteLine($"Received a transcription from
'{e.Result.ParticipantId}': '{e.Result.Text}'");
        foreach (var entry in e.Result.Translations)
        {
            Console.WriteLine($"{entry.Key}: {entry.Value}");
        }
    };
    conversationTranslator.Transcribing += (s, e) =>
    {
        Console.WriteLine($"Received a partial transcription from
'{e.Result.ParticipantId}': '{e.Result.Text}'");
        foreach (var entry in e.Result.Translations)
        {
            Console.WriteLine($"{entry.Key}: {entry.Value}");
        }
    };

    // To start receiving events, you will need to join the conversation
    await conversationTranslator.JoinConversationAsync(conversationId,
    displayName, language).ConfigureAwait(false);

    // You can now send an instant message
    await conversationTranslator.SendTextMessageAsync("Message from
participant").ConfigureAwait(false);

    // Start capturing audio if you specified a speech to text language
    await
conversationTranslator.StartTranscribingAsync().ConfigureAwait(false);
    Console.WriteLine("Started transcribing. Press Ctrl-C to stop");

    // At this point, you should start receiving transcriptions for what
you are saying using
    // the default microphone. Press Ctrl+C to stop audio capture
    await completionSource.Task.ConfigureAwait(false);

    // Stop audio capture
    await
conversationTranslator.StopTranscribingAsync().ConfigureAwait(false);

    // Leave the conversation. You will stop receiving events after this
    await
conversationTranslator.LeaveConversationAsync().ConfigureAwait(false);
}
}

```

2. Replace `CreateConversationAsync();` in your `public static async Task Main(string[] args)` function with:

C#

```
// Set this to the conversation you want to join
JoinConversationAsync("YourConversationId");
```

3. You'll need to create a conversation that you can join:

- a. Launch your browser and navigate to: <https://translator.microsoft.com>.
- b. Select "Start conversation".
- c. Sign in using any of the available options.
- d. Type in a name (e.g. The host).
- e. Select a language (e.g. English).
- f. Select the **Enter** button.
- g. Note the conversation code at the top of the page.

 **Tip**

If you use the copy button on the web page, make sure you remove the `translate.it/` at the start. You only need the 5 character conversation ID (example: ABCDE).

4. Go back to Visual Studio and replace the string `YourConversationId` with the conversation ID you created in the previous step.
5. From the menu bar, select **Build > Build Solution** to build the application. The code should compile without errors now.
6. Choose **Debug > Start Debugging** (or press **F5**) to start the **helloworld** application.
7. Once you see the `Started transcribing` message appear, you can start speaking. You'll see the transcriptions appear as you speak.
 - If you go back to your browser, you should see your transcriptions appear there as you speak as well.
8. Once you're done speaking, press `Ctrl+C` to stop audio capture, and end the conversation.
9. Go back to your browser and exit the conversation using the exit button in the upper right corner.

Next Steps

[Explore C# samples on GitHub](#)

Role-based access control for Speech resources

05/22/2025

You can manage access and permissions to your Speech resources with Azure role-based access control (Azure RBAC). Assigned roles can vary across Speech resources.

For example, you can assign a role to an AI Speech resource that should only be used to train a custom speech model. You can assign another role to an AI Speech resource that is used to transcribe audio files.

Depending on who can access each Speech resource, you can effectively set a different level of access per application or user. For more information on Azure RBAC, see the [Azure RBAC documentation](#).

! Note

This article describes how to assign access roles for an AI Speech resource. For information on how to assign access roles for Azure AI Foundry resources, see the [Azure AI Foundry documentation](#).

Roles for Speech resources

A role definition is a collection of permissions. An AI Speech resource can inherit or be assigned multiple roles. The final level of access to the resource is a combination of all role permissions. When you create an AI Speech resource, the built-in roles in the following table are available for assignment.

Expand table

Role	Can list resource keys	Access to data, models, and endpoints in custom projects	Access to speech transcription and synthesis APIs
Owner	Yes	None	No
Contributor	Yes	None	No
Cognitive Services Contributor	Yes	None	No

Role	Can list resource keys	Access to data, models, and endpoints in custom projects	Access to speech transcription and synthesis APIs
Cognitive Services User	Yes	View, create, edit, and delete	Yes
Cognitive Services Speech Contributor	No	View, create, edit, and delete	Yes
Cognitive Services Speech User	No	View only	Yes
Cognitive Services Data Reader (Preview)	No	View only	Yes

Keep the built-in roles if your Speech resource can have full read and write access to the projects.

For finer-grained resource access control, you can [add or remove roles](#) using the Azure portal. For example, you could create a custom role with permission to upload custom speech datasets, but without permission to deploy a custom speech model to an endpoint.

Special considerations for Speech resources

i Important

Speech service architecture differs from other Azure AI services in the way it uses [Azure control plane and data plane](#).

Speech service is extensively using data plane comparing to other Azure AI services, and this requires different set up for the roles. Because of this some general "Cognitive Services" roles have actual access right set that doesn't exactly match their name when used in Speech services scenario.

For instance *Cognitive Services User* provides in effect the Contributor rights, while *Cognitive Services Contributor* provides no access at all. The same is true for generic *Owner* and *Contributor* roles which have no data plane rights and consequently provide no access to Speech resource.

To keep consistency we recommend to use roles containing *Speech* in their names. These roles are *Cognitive Services Speech User* and *Cognitive Services Speech Contributor*. Their access right sets were designed specifically for the Speech service.

Authentication with keys and tokens

The [roles](#) define what permissions you have. Authentication is required to use the Speech resource.

To authenticate with Speech resource keys, all you need is the key and region. To authenticate with a Microsoft Entra token, the Speech resource must have a [custom subdomain](#).

Here's how to create a new Speech resource with a custom subdomain. You can also use an existing resource, but it must have a custom subdomain. For more information about creating a custom subdomain, see [Create a custom domain name](#).

Bash

```
resourceGroupName=my-speech-rg
location=eastus
AI Services resourceName=my-aiservices-$location

# create an AI Services resource for Speech and other AI services
az cognitiveservices account create --name $AI Services resourceName --resource-group $resourceGroupName --kind AI Services --sku S0 --location $location --custom-domain $AI Services resourceName

# get the resource id
speechResourceId=$(az cognitiveservices account show --name
$AI Services resourceName --resource-group $resourceGroupName --query id -o tsv)
# assign Cognitive Services User role to the app id
appId=$(az ad signed-in-user show --query id -o tsv)
az role assignment create --role "Cognitive Services User" --assignee $appId --
scope $speechResourceId
# assign Cognitive Services Speech User role to the app id
az role assignment create --role "Cognitive Services Speech User" --assignee
$appId --scope $speechResourceId

# get an access token
accessToken=$(az account get-access-token --scope
"https://cognitiveservices.azure.com/.default" --query accessToken -o tsv)
echo $accessToken
```

The returned `accessToken` is a Microsoft Entra token that you can use to authenticate without API keys. The token has a [limited lifetime](#).

Now you can use the `accessToken` to authenticate with the AI Foundry resource. For example, you can use the token via the [Fast transcription REST API](#):

Bash

```
uri="https://$AI Services resourceName.cognitiveservices.azure.com/speechtotext/transcriptions:transcribe?api-version=2024-11-15"
```

```
curl -v "$uri" \
--header 'Content-Type: multipart/form-data' \
--form 'definition={"locales": ["en-US"]}' \
--form 'audio=@Call1_separated_16k_health_insurance.wav' \
--header "Authorization: Bearer $accessToken"
```

Speech SDK authentication

For the SDK, you configure whether to authenticate with an API key or Microsoft Entra token. For details, see [Microsoft Entra authentication with the Speech SDK.](#) |

Next steps

- [Microsoft Entra authentication with the Speech SDK.](#)
- [Speech service encryption of data at rest.](#)

Microsoft Entra authentication with the Speech SDK

08/13/2025

When using the Speech SDK to access the Speech service, there are three authentication methods available: service keys, a key-based token, and Microsoft Entra ID. This article describes how to configure an AI Foundry resource and create a Speech SDK configuration object to use Microsoft Entra ID for authentication.

This article shows how to use Microsoft Entra authentication with the Speech SDK. You learn how to:

- ✓ Create an AI Foundry resource
- ✓ Configure the Speech resource for Microsoft Entra authentication
- ✓ Get a Microsoft Entra access token
- ✓ Create the appropriate SDK configuration object.

To learn more about Microsoft Entra access tokens, including token lifetime, visit [Access tokens in the Microsoft identity platform](#).

Create an AI Foundry resource

To create an AI Foundry resource in the [Azure portal](#), see [this quickstart](#).

Configure the Speech resource for Microsoft Entra authentication

To configure your Speech resource for Microsoft Entra authentication, create a custom domain name and assign roles.

Create a custom domain name

Follow these steps to create a [custom subdomain name for Azure AI services](#) for your Speech resource.

✖ Caution

When you turn on a custom domain name, the operation is [not reversible](#). The only way to go back to the [regional name](#) is to create a new Speech resource.

If your Speech resource has a lot of associated custom models and projects created via [Speech Studio](#), we strongly recommend trying the configuration with a test resource before you modify the resource used in production.

Azure portal

To create a custom domain name using the Azure portal, follow these steps:

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the required Speech resource.
3. In the **Resource Management** group on the left pane, select **Networking**.
4. On the **Firewalls and virtual networks** tab, select **Generate Custom Domain Name**. A new right panel appears with instructions to create a unique custom subdomain for your resource.
5. In the **Generate Custom Domain Name** panel, enter a custom domain name. Your full custom domain will look like: `https://{{your custom name}}.cognitiveservices.azure.com`.
Remember that after you create a custom domain name, it *cannot* be changed.
After you've entered your custom domain name, select **Save**.
6. After the operation finishes, in the **Resource management** group, select **Keys and Endpoint**. Confirm that the new endpoint name of your resource starts this way:
`https://{{your custom name}}.cognitiveservices.azure.com`.

Assign roles

For Microsoft Entra authentication with Speech resources, you need to assign either the *Cognitive Services Speech Contributor* or *Cognitive Services Speech User* role.

You can assign roles to the user or application using the [Azure portal](#) or [PowerShell](#).

Get a Microsoft Entra access token

To get a Microsoft Entra access token in C#, use the [Azure Identity Client Library](#).

Here's an example of using Azure Identity to get a Microsoft Entra access token from an interactive browser:

```
c#  
  
TokenRequestContext context = new Azure.Core.TokenRequestContext(new string[] {  
    "https://cognitiveservices.azure.com/.default" });  
InteractiveBrowserCredential browserCredential = new  
InteractiveBrowserCredential();  
var browserToken = browserCredential.GetToken(context);  
string aadToken = browserToken.Token;
```

! Note

The token context must be set to "https://cognitiveservices.azure.com/.default".

More samples

Find samples that get a Microsoft Entra access token in [Microsoft identity platform code samples](#).

For programming languages where a Microsoft identity platform client library isn't available, you can directly [request an access token](#).

Get the Speech resource ID

You need your Speech resource ID to make SDK calls using Microsoft Entra authentication in scenarios that don't yet support Entra ID directly.

Azure portal

To get the resource ID in the Azure portal:

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select an AI Foundry resource.
3. In the **Resource Management** group on the left pane, select **Properties**.
4. Copy the **Resource ID**

Create the Speech SDK configuration object

With a Microsoft Entra access token, you can now create a Speech SDK configuration object.

The method of providing the token, and the method to construct the corresponding Speech SDK `Config` object varies by the object you're using.

SpeechRecognizer, SourceLanguageRecognizer, ConversationTranscriber

For `SpeechRecognizer`, `SourceLanguageRecognizer`, `ConversationTranscriber` objects, use an appropriate instance of `TokenCredential` for authentication, along with the endpoint that includes your `custom domain`, to create a `SpeechConfig` object.

C#

```
TokenCredential browserCredential = new InteractiveBrowserCredential();

// Define the custom domain endpoint for your Speech resource.
var endpoint = "https://{{your custom name}}.cognitiveservices.azure.com/";

// Create the SpeechConfig object using the custom domain endpoint and
// TokenCredential.
var speechConfig = SpeechConfig.FromEndpoint(new Uri(endpoint),
browserCredential);
```

TranslationRecognizer

For `TranslationRecognizer` object, use an appropriate instance of `TokenCredential` for authentication, along with the endpoint that includes your `custom domain`, to create a `SpeechTranslationConfig` object.

C#

```
TokenCredential browserCredential = new InteractiveBrowserCredential();

// Define the custom domain endpoint for your Speech resource
var endpoint = "https://{{your custom name}}.cognitiveservices.azure.com/";

// Create the SpeechTranslationConfig object using the custom domain endpoint and
// TokenCredential.
var speechConfig = SpeechTranslationConfig.FromEndpoint(new Uri(endpoint),
browserCredential);
```

SpeechSynthesizer, IntentRecognizer

For `SpeechSynthesizer`, `IntentRecognizer` objects, build the authorization token from the resource ID and the Microsoft Entra access token and then use it to create a `SpeechConfig` object.

C#

```
string resourceId = "Your Resource ID";
string aadToken = "Your Microsoft Entra access token";
string region = "Your Speech Region";

// You need to include the "aad#" prefix and the "#" (hash) separator between
// resource ID and Microsoft Entra access token.
var authorizationToken = $"aad#{resourceId}#{aadToken}";
var speechConfig = SpeechConfig.FromAuthorizationToken(authorizationToken,
region);
```

VoiceProfileClient

To use the `VoiceProfileClient` with Microsoft Entra authentication, use the custom domain name created above.

C#

```
string customDomainName = "Your Custom Name";
string hostName = $"https://'{customDomainName}'.cognitiveservices.azure.com/";
string token = "Your Microsoft Entra access token";

var config = SpeechConfig.FromHost(new Uri(hostName));

// You need to include the "aad#" prefix and the "#" (hash) separator between
// resource ID and Microsoft Entra access token.
var authorizationToken = $"aad#{resourceId}#{aadToken}";
config.AuthorizationToken = authorizationToken;
```

ⓘ Note

The `ConversationTranslator` doesn't support Microsoft Entra authentication.

Audio concepts in Azure AI Speech

Article • 04/28/2025

The Speech service accepts and provides audio in multiple formats, and the area of audio is a complex topic but some background information can be helpful.

Audio concepts

Speech is inherently analog, which is approximated by converting it to a digital signal by sampling. The number of times it's sampled per second is the sampling rate, and how accurate each sample is defined by the bit-depth.

Sample rate

How many audio samples there are per second. A higher sampling rate will more accurately reproduce higher frequencies such as music. Humans can typically hear between 20 Hz and 20 kHz but most sensitive up to 5 kHz. The sample rate needs to be twice the highest frequency so for human speech a 16 kHz sampling rate is normally adequate, but a higher sampling rate can provide a higher quality although larger files. The default for both speech to text and text to speech is 16 kHz, however 48 kHz is recommended for audio books. Some source audio is in 8 kHz, especially when coming from legacy telecom systems, which will result in degraded results.

Bit-depth

Uncompressed audio samples are each represented by many bits that define its accuracy or resolution. For human speech 13 bits are needed, which is rounded up to a 16 bit sample. A higher bit-depth would be needed for professional audio or music. Legacy telephony systems often use 8 bits with compression, but it isn't ideal.

Channels

The Speech service typically expects and provides a mono stream. The behavior of stereo and multi-channel files is API specific, for example the speech to text REST API will split a stereo file and generate a result for each channel. Text to speech is mono only.

Audio formats and codecs

For the Speech service to be able to use the audio it needs to know how it's encoded. Also as audio files can be relatively large it's common to use compression to reduce their size. Audio

files and streams can be described by their container format and the audio codec. Common containers are WAV or MP4 and common audio formats are PCM or MP3. You normally can't presume that a container uses a specific audio format, for instance WAV files often contain PCM data but other audio formats are possible.

Uncompressed audio

The Speech service internally works on uncompressed audio, which is encoded with Pulse Code Modulation (or PCM). This means that every sample represents the amplitude of the signal. This is a simple representation for processing, but not space efficient so compression is often used for transporting audio.

Lossy compressed audio

Lossy algorithms might enable greater compression resulting in smaller files or lower bandwidth, which can be important on mobile connections or busy networks. A common audio format is MP3, which is an example of lossy compression. MP3 files are significantly smaller than the originals, and might sound nearly identical to the original, but you can't recreate the exact source file. Lossy compression works by removing parts of the audio or approximating it. When encoding with a lossy algorithm you trade off bandwidth for accuracy.

MP3 was designed for music rather than speech.

AMR and AMR-WB were designed to efficiently compress speech for mobile phones, and won't work as well representing music or noise.

A-Law and Mu-Law are older algorithms that compress each sample by itself, and converts a 16 bit sample to 8 bit using a logarithmic quantization technique. It should only be used to support legacy systems.

Lossless compressed audio

Lossless compression allows you to recreate the original uncompressed file. The compressed file is typically much smaller than the original, without any loss, but the actual compression depends on the input. It achieves compression by uses multiple methods to remove redundancy from the file.

The most common lossless compression is FLAC.

Next steps

Use the Speech SDK for audio processing

How to use the audio input stream

08/07/2025

The Speech SDK provides a way to stream audio into the recognizer as an alternative to microphone or file input.

This guide describes how to use audio input streams. It also describes some of the requirements and limitations of the audio input stream.

See more examples of speech to text recognition with audio input stream on [GitHub](#).

Identify the format of the audio stream

Identify the format of the audio stream.

Supported audio samples are:

- PCM format (int-16, signed)
- One channel
- 16 bits per sample, 8,000 or 16,000 samples per second (16,000 bytes or 32,000 bytes per second)
- Two-block aligned (16 bit including padding for a sample)

The corresponding code in the SDK to create the audio format looks like this example:

C#

```
byte channels = 1;
byte bitsPerSample = 16;
int samplesPerSecond = 16000; // or 8000
var audioFormat = AudioStreamFormat.GetWaveFormatPCM(samplesPerSecond,
bitsPerSample, channels);
```

Make sure that your code provides the RAW audio data according to these specifications. Also, make sure that 16-bit samples arrive in little-endian format. If your audio source data doesn't match the supported formats, the audio must be transcoded into the required format.

Create your own audio input stream class

You can create your own audio input stream class derived from `PullAudioInputStreamCallback`. Implement the `Read()` and `Close()` members. The exact function signature is language-dependent, but the code looks similar to this code sample:

C#

```
public class ContosoAudioStream : PullAudioInputStreamCallback
{
    public ContosoAudioStream() {}

    public override int Read(byte[] buffer, uint size)
    {
        // Returns audio data to the caller.
        // E.g., return read(config.YYY, buffer, size);
        return 0;
    }

    public override void Close()
    {
        // Close and clean up resources.
    }
}
```

Create an audio configuration based on your audio format and custom audio input stream. For example:

C#

```
var audioConfig = AudioConfig.FromStreamInput(new ContosoAudioStream(),
audioFormat);
```

Here's how the custom audio input stream is used in the context of a speech recognizer:

C#

```
using System;
using System.IO;
using System.Threading.Tasks;
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;

public class ContosoAudioStream : PullAudioInputStreamCallback
{
    public ContosoAudioStream() {}

    public override int Read(byte[] buffer, uint size)
    {
        // Returns audio data to the caller.
        // E.g., return read(config.YYY, buffer, size);
        return 0;
    }

    public override void Close()
    {
        // Close and clean up resources.
    }
}
```

```
    }

}

class Program
{
    static string speechKey = Environment.GetEnvironmentVariable("SPEECH_KEY");
    static string speechRegion =
Environment.GetEnvironmentVariable("SPEECH_REGION");

    async static Task Main(string[] args)
    {
        byte channels = 1;
        byte bitsPerSample = 16;
        uint samplesPerSecond = 16000; // or 8000
        var audioFormat = AudioStreamFormat.GetWaveFormatPCM(samplesPerSecond,
bitsPerSample, channels);
        var audioConfig = AudioConfig.FromStreamInput(new ContosoAudioStream(),
audioFormat);

        var speechConfig = SpeechConfig.FromSubscription(speechKey, speechRegion);
        speechConfig.SpeechRecognitionLanguage = "en-US";
        var speechRecognizer = new SpeechRecognizer(speechConfig, audioConfig);

        Console.WriteLine("Speak into your microphone.");
        var speechRecognitionResult = await speechRecognizer.RecognizeOnceAsync();
        Console.WriteLine($"RECOGNIZED: Text={speechRecognitionResult.Text}");
    }
}
```

Next steps

- [Speech to text quickstart](#)
- [How to recognize speech](#)

How to use compressed input audio

08/07/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

The Speech SDK and Speech CLI use GStreamer to support different kinds of input audio formats. GStreamer decompresses the audio before it's sent over the wire to the Speech service as raw PCM.

The default audio streaming format is WAV (16 kHz or 8 kHz, 16-bit, and mono PCM). Outside WAV and PCM, the following compressed input formats are also supported through GStreamer:

- MP3
- OPUS/OGG
- FLAC
- ALAW in WAV container
- MULAW in WAV container
- ANY for MP4 container or unknown media format

GStreamer configuration

The Speech SDK can use [GStreamer](#) ↗ to handle compressed audio. For licensing reasons, GStreamer binaries aren't compiled and linked with the Speech SDK. You need to install some dependencies and plug-ins.

GStreamer binaries must be in the system path so that they can be loaded by the Speech SDK at runtime. For example, on Windows, if the Speech SDK finds `libgstreamer-1.0-0.dll` or `gstreamer-1.0-0.dll` (for the latest GStreamer) during runtime, it means the GStreamer binaries are in the system path.

Choose a platform for installation instructions.

Windows

Make sure that packages of the same platform (x64 or x86) are installed. For example, if you installed the x64 package for Python, you need to install the x64 GStreamer package. The following instructions are for the x64 packages.

1. Create the folder `c:\gstreamer`.
2. Download the [installer](#) ↗ .

3. Copy the installer to c:\gstreamer.
4. Open PowerShell as an administrator.
5. Run the following command in PowerShell:

```
PowerShell  
  
cd c:\gstreamer  
msiexec /passive INSTALLLEVEL=1000 INSTALLDIR=C:\gstreamer /i gstreamer-  
1.0-msvc-x86_64-1.18.3.msi
```

6. Add the system variable `GST_PLUGIN_PATH` with "C:\gstreamer\1.0\msvc_x86_64\lib\gstreamer-1.0" as the variable value.
7. Add the system variable `GSTREAMER_ROOT_X86_64` with "C:\gstreamer\1.0\msvc_x86_64" as the variable value.
8. Edit the system `PATH` variable to add "C:\gstreamer\1.0\msvc_x86_64\bin" as a new entry.
9. Reboot the machine.

For more information about GStreamer, see [Windows installation instructions ↗](#).

Example

To configure the Speech SDK to accept compressed audio input, create `PullAudioInputStream` or `PushAudioInputStream`. Then, create an `AudioConfig` from an instance of your stream class that specifies the compression format of the stream. Find related sample code snippets in [About the Speech SDK audio input stream API](#).

Let's assume that you have an input stream class called `pullStream` and are using OPUS/OGG. Your code might look like this:

```
C#  
  
using Microsoft.CognitiveServices.Speech;  
using Microsoft.CognitiveServices.Speech.Audio;  
  
// ... omitted for brevity  
  
var speechConfig =  
    SpeechConfig.FromSubscription(  
        "YourSpeechResourceKey",
```

```
"YourServiceRegion");  
  
// Create an audio config specifying the compressed  
// audio format and the instance of your input stream class.  
var pullStream = AudioInputStream.CreatePullStream(  
    AudioStreamFormat.GetCompressedFormat(AudioStreamContainerFormat.OGG_OPUS));  
var audioConfig = AudioConfig.FromStreamInput(pullStream);  
  
using var recognizer = new SpeechRecognizer(speechConfig, audioConfig);  
var result = await recognizer.RecognizeOnceAsync();  
  
var text = result.Text;
```

Next steps

- [Try the speech to text quickstart](#)
- [Improve recognition accuracy with custom speech](#)

Select an audio input device with the Speech SDK

08/07/2025

This article describes how to obtain the IDs of the audio devices connected to a system. These IDs can then be used in the Speech SDK to select the audio input. You configure the audio device through the `AudioConfig` object:

C++

```
audioConfig = AudioConfig.FromMicrophoneInput("<device id>");
```

cs

```
audioConfig = AudioConfig.FromMicrophoneInput("<device id>");
```

Python

```
audio_config = AudioConfig(device_name="<device id>");
```

Objective-C

```
audioConfig = AudioConfiguration.FromMicrophoneInput("<device id>");
```

Java

```
audioConfig = AudioConfiguration.fromMicrophoneInput("<device id>");
```

JavaScript

```
audioConfig = AudioConfiguration.fromMicrophoneInput("<device id>");
```

ⓘ Note

Microphone use isn't available for JavaScript running in Node.js.

Audio device IDs on Windows for desktop applications

Audio device [endpoint ID strings](#) can be retrieved from the [IMMDevice](#) object in Windows for desktop applications.

The following code sample illustrates how to use it to enumerate audio devices in C++:

C++

```
#include <stdio>
#include <mmdeviceapi.h>

#include <FunctionDiscoveryKeys_devkey.h>

const CLSID CLSID_MMDeviceEnumerator = __uuidof(MMDeviceEnumerator);
const IID IID_IMMDeviceEnumerator = __uuidof(IMMDeviceEnumerator);

constexpr auto REFTIMES_PER_SEC = (10000000 * 25);
constexpr auto REFTIMES_PER_MILLISEC = 10000;

#define EXIT_ON_ERROR(hres) \
    if (FAILED(hres)) { goto Exit; }
#define SAFE_RELEASE(punk) \
    if ((punk) != NULL) \
        { (punk)->Release(); (punk) = NULL; }

void ListEndpoints();

int main()
{
    CoInitializeEx(NULL, COINIT_MULTITHREADED);
    ListEndpoints();
}

//-----
// This function enumerates all active (plugged in) audio
// rendering endpoint devices. It prints the friendly name
// and endpoint ID string of each endpoint device.
//-----
void ListEndpoints()
{
    HRESULT hr = S_OK;
    IMMDeviceEnumerator *pEnumerator = NULL;
    IMMDeviceCollection *pCollection = NULL;
    IMMDevice *pEndpoint = NULL;
    IPropertyStore *pProps = NULL;
    LPWSTR pwszID = NULL;

    hr = CoCreateInstance(CLSID_MMDeviceEnumerator, NULL, CLSCTX_ALL,
    IID_IMMDeviceEnumerator, (void**)&pEnumerator);
    EXIT_ON_ERROR(hr);
```

```

    hr = pEnumerator->EnumAudioEndpoints(eCapture, DEVICE_STATE_ACTIVE,
&pCollection);
    EXIT_ON_ERROR(hr);

    UINT count;
    hr = pCollection->GetCount(&count);
    EXIT_ON_ERROR(hr);

    if (count == 0)
    {
        printf("No endpoints found.\n");
    }

    // Each iteration prints the name of an endpoint device.
    PROPVARIANT varName;
    for (ULONG i = 0; i < count; i++)
    {
        // Get the pointer to endpoint number i.
        hr = pCollection->Item(i, &pEndpoint);
        EXIT_ON_ERROR(hr);

        // Get the endpoint ID string.
        hr = pEndpoint->GetId(&pwszID);
        EXIT_ON_ERROR(hr);

        hr = pEndpoint->OpenPropertyStore(
            STGM_READ, &pProps);
        EXIT_ON_ERROR(hr);

        // Initialize the container for property value.
        PropVariantInit(&varName);

        // Get the endpoint's friendly-name property.
        hr = pProps->GetValue(PKEY_Device_FriendlyName, &varName);
        EXIT_ON_ERROR(hr);

        // Print the endpoint friendly name and endpoint ID.
        printf("Endpoint %d: \"%S\" (%S)\n", i, varName.pwszVal, pwszID);

        CoTaskMemFree(pwszID);
        pwszID = NULL;
        PropVariantClear(&varName);
    }

Exit:
    CoTaskMemFree(pwszID);
    pwszID = NULL;
    PropVariantClear(&varName);
    SAFE_RELEASE(pEnumerator);
    SAFE_RELEASE(pCollection);
    SAFE_RELEASE(pEndpoint);
    SAFE_RELEASE(pProps);
}

```

In C#, you can use the [NAudio](#) library to access the CoreAudio API and enumerate devices as follows:

```
cs

using System;
using NAudio.CoreAudioApi;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var enumerator = new MMDeviceEnumerator();
            foreach (var endpoint in
                enumerator.EnumerateAudioEndPoints(DataFlow.Capture,
DeviceState.Active))
            {
                Console.WriteLine("{0} ({1})", endpoint.FriendlyName,
endpoint.ID);
            }
        }
    }
}
```

A sample device ID is `{0.0.1.00000000}.{5f23ab69-6181-4f4a-81a4-45414013aac8}`.

Audio device IDs on UWP

On the Universal Windows Platform (UWP), you can obtain audio input devices by using the `Id()` property of the corresponding [DeviceInformation](#) object.

The following code samples show how to do this step in C++ and C#:

```
C++

#include <winrt/Windows.Foundation.h>
#include <winrt/Windows.Devices.Enumeration.h>

using namespace winrt::Windows::Devices::Enumeration;

void enumerateDeviceIds()
{
    auto promise = DeviceInformation::FindAllAsync(DeviceClass::AudioCapture);

    promise.Completed(
        []
    (winrt::Windows::Foundation::IAsyncOperation<DeviceInformationCollection> const&
```

```

    sender,
        winrt::Windows::Foundation::AsyncStatus /* asyncStatus */) {
    auto info = sender.GetResults();
    auto num_devices = info.Size();

    for (const auto &device : info)
    {
        std::wstringstream ss{};
        ss << "looking at device (of " << num_devices << "): " <<
device.Id().c_str() << "\n";
        OutputDebugString(ss.str().c_str());
    }
});
}

```

cs

```

using Windows.Devices.Enumeration;
using System.Linq;

namespace helloworld {
    private async void EnumerateDevices()
    {
        var devices = await
DeviceInformation.FindAllAsync(DeviceClass.AudioCapture);

        foreach (var device in devices)
        {
            Console.WriteLine($"{device.Name}, {device.Id}\n");
        }
    }
}

```

A sample device ID is `\\\\?\\\SWD#MMDEVAPI#{0.0.1.00000000}.{5f23ab69-6181-4f4a-81a4-45414013aac8}#{2eef81be-33fa-4800-9670-1cd474972c3f}`.

Audio device IDs on Linux

The device IDs are selected by using standard ALSA device IDs.

The IDs of the inputs attached to the system are contained in the output of the command `arecord -L`. Alternatively, they can be obtained by using the [ALSA C library](#).

Sample IDs are `hw:1,0` and `hw:CARD=CC,DEV=0`.

Audio device IDs on macOS

The following function implemented in Objective-C creates a list of the names and IDs of the audio devices attached to a Mac.

The `deviceUID` string is used to identify a device in the Speech SDK for macOS.

Objective-C

```
#import <Foundation/Foundation.h>
#import <CoreAudio/CoreAudio.h>

CFArrayRef CreateInputDeviceArray()
{
    AudioObjectPropertyAddress propertyAddress = {
        kAudioHardwarePropertyDevices,
        kAudioObjectPropertyScopeGlobal,
        kAudioObjectPropertyElementMaster
    };

    UInt32 dataSize = 0;
    OSStatus status = AudioObjectGetPropertyDataSize(kAudioObjectSystemObject,
&propertyAddress, 0, NULL, &dataSize);
    if (kAudioHardwareNoError != status) {
        fprintf(stderr, "AudioObjectGetPropertyDataSize
(kAudioHardwarePropertyDevices) failed: %i\n", status);
        return NULL;
    }

    UInt32 deviceCount = (uint32)(dataSize / sizeof(AudioDeviceID));

    AudioDeviceID *audioDevices = (AudioDeviceID *)(malloc(dataSize));
    if (NULL == audioDevices) {
        fputs("Unable to allocate memory", stderr);
        return NULL;
    }

    status = AudioObjectGetPropertyData(kAudioObjectSystemObject,
&propertyAddress, 0, NULL, &dataSize, audioDevices);
    if (kAudioHardwareNoError != status) {
        fprintf(stderr, "AudioObjectGetPropertyData
(kAudioHardwarePropertyDevices) failed: %i\n", status);
        free(audioDevices);
        audioDevices = NULL;
        return NULL;
    }

    CFMutableArrayRef inputDeviceArray = CFArrayCreateMutable(kCFAlocatorDefault,
deviceCount, &kCFTypeArrayCallBacks);
    if (NULL == inputDeviceArray) {
        fputs("CFArrayCreateMutable failed", stderr);
        free(audioDevices);
        audioDevices = NULL;
        return NULL;
    }
```

```

// Iterate through all the devices and determine which are input-capable
propertyAddress.mScope = kAudioDevicePropertyScopeInput;
for (UInt32 i = 0; i < deviceCount; ++i) {
    // Query device UID
    CFStringRef deviceUID = NULL;
    dataSize = sizeof(deviceUID);
    propertyAddress.mSelector = kAudioDevicePropertyDeviceUID;
    status = AudioObjectGetPropertyData(audioDevices[i], &propertyAddress, 0,
NULL, &dataSize, &deviceUID);
    if (kAudioHardwareNoError != status) {
        fprintf(stderr, "AudioObjectGetPropertyData
(kAudioDevicePropertyDeviceUID) failed: %i\n", status);
        continue;
    }

    // Query device name
    CFStringRef deviceName = NULL;
    dataSize = sizeof(deviceName);
    propertyAddress.mSelector = kAudioDevicePropertyNameCFString;
    status = AudioObjectGetPropertyData(audioDevices[i], &propertyAddress, 0,
NULL, &dataSize, &deviceName);
    if (kAudioHardwareNoError != status) {
        fprintf(stderr, "AudioObjectGetPropertyData
(kAudioDevicePropertyNameCFString) failed: %i\n", status);
        continue;
    }

    // Determine if the device is an input device (it is an input device if it
has input channels)
    dataSize = 0;
    propertyAddress.mSelector = kAudioDevicePropertyStreamConfiguration;
    status = AudioObjectGetPropertyDataSize(audioDevices[i], &propertyAddress,
0, NULL, &dataSize);
    if (kAudioHardwareNoError != status) {
        fprintf(stderr, "AudioObjectGetPropertyDataSize
(kAudioDevicePropertyStreamConfiguration) failed: %i\n", status);
        continue;
    }

    AudioBufferList *bufferList = (AudioBufferList *)malloc(dataSize));
    if (NULL == bufferList) {
        fputs("Unable to allocate memory", stderr);
        break;
    }

    status = AudioObjectGetPropertyData(audioDevices[i], &propertyAddress, 0,
NULL, &dataSize, bufferList);
    if (kAudioHardwareNoError != status || 0 == bufferList->mNumberBuffers) {
        if (kAudioHardwareNoError != status)
            fprintf(stderr, "AudioObjectGetPropertyData
(kAudioDevicePropertyStreamConfiguration) failed: %i\n", status);
        free(bufferList);
        bufferList = NULL;
        continue;
    }
}

```

```

    }

    free(bufferList);
    bufferList = NULL;

    // Add a dictionary for this device to the array of input devices
    CFStringRef keys [] = { CFSTR("deviceUID"), CFSTR("deviceName") };
    CFStringRef values [] = { deviceUID, deviceName};

    CFDictionaryRef deviceDictionary = CFDictionaryCreate(kCFAlocatorDefault,
        (const void **)
    (keys),
        (const void **)
    (values),
        2,
        &kCFTypeDictionaryKeyCallBacks,
        &kCFTypeDictionaryValueCallBacks);

    CFArrayAppendValue(inputDeviceArray, deviceDictionary);

    CFRelease(deviceDictionary);
    deviceDictionary = NULL;
}

free(audioDevices);
audioDevices = NULL;

// Return a non-mutable copy of the array
CFArrayRef immutableInputDeviceArray = CFArrayCreateCopy(kCFAlocatorDefault,
inputDeviceArray);
CFRelease(inputDeviceArray);
inputDeviceArray = NULL;

return immutableInputDeviceArray;
}

```

For example, the UID for the built-in microphone is `BuiltInMicrophoneDevice`.

Audio device IDs on iOS

Audio device selection with the Speech SDK isn't supported on iOS. Apps that use the SDK can influence audio routing through the [AVAudioSession](#) Framework.

For example, the instruction

Objective-C

```
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryRecord
```

```
withOptions:AVAudioSessionCategoryOptionAllowBluetooth error:NULL];
```

Enables the use of a Bluetooth headset for a speech-enabled app.

Audio device IDs in JavaScript

In JavaScript, the [MediaDevices.enumerateDevices\(\)](#) method can be used to enumerate the media devices and find a device ID to pass to `fromMicrophone(...)`.

Next steps

- [Explore samples on GitHub](#)
- [Customize acoustic models](#)
- [Customize language models](#)

Speech containers overview

07/01/2025

By using containers, you can use a subset of the Speech service features in your own environment. With Speech containers, you can build a speech application architecture optimized for both robust cloud capabilities and edge locality. Containers are great for specific security and data governance requirements.

Available Speech containers

The following table lists the Speech containers available in the Microsoft Container Registry (MCR). The table also lists the features supported by each container and the latest version of the container.

 Expand table

Container	Features	Supported versions and locales
Speech to text	Transcribes continuous real-time speech or batch audio recordings with intermediate results.	Latest: 4.12.0 Latest preview: 5.0.1 For all supported versions and locales, see the Microsoft Container Registry (MCR) and JSON tags .
Custom speech to text	Using a custom model from the custom speech portal , transcribes continuous real-time speech or batch audio recordings into text with intermediate results.	Latest: 4.12.0 Latest preview: 5.0.1 For all supported versions and locales, see the Microsoft Container Registry (MCR) and JSON tags .
Speech language identification ^{1, 2}	Detects the language spoken in audio files.	Latest preview: 1.18.0 For all supported versions and locales, see the Microsoft Container Registry (MCR) and JSON tags .

Container	Features	Supported versions and locales
Neural text to speech	Converts text to natural-sounding speech by using deep neural network technology, which allows for more natural synthesized speech.	Latest: 3.11.0 For all supported versions and locales, see the Microsoft Container Registry (MCR) and JSON tags .

¹ The container is available in public preview. Containers in preview are still under development and don't meet Microsoft's stability and support requirements.

² Not available as a disconnected container.

Request approval to run containers disconnected from the internet

To use the Speech containers in environments that are disconnected from the internet, you must submit a [request form](#) and wait for approval. For more information about applying and purchasing a commitment plan to use containers in disconnected environments, see [Use containers in disconnected environments](#) in the Azure AI services documentation.

The form requests information about you, your company, and the user scenario for which you use the container.

- On the form, you must use an email address associated with an Azure subscription ID.
- The Azure resource you use to run the container must be created with the approved Azure subscription ID.
- Check your email for updates on the status of your application from Microsoft.

After you submit the form, the Azure AI services team reviews it and emails you with a decision within 10 business days.

Billing

The Speech containers send billing information to Azure by using an AI Foundry resource for Speech on your Azure account.

Note

Connected and disconnected container pricing and commitment tiers vary. For more information, see [Speech service pricing](#).

Speech containers aren't licensed to run without being connected to Azure for metering. You must configure your container to always communicate billing information with the metering service. For more information, see [billing arguments](#).

Container recipes and other container services

You can use container recipes to create containers that can be reused. Containers can be built with some or all configuration settings so that they aren't needed when the container is started. For container recipes see the following Azure AI services articles:

- [Create containers for reuse](#)
- [Deploy and run container on Azure Container Instance](#)
- [Deploy a language detection container to Azure Kubernetes Service](#)
- [Use Docker Compose to deploy multiple containers](#)

For information about other container services, see the following Azure AI services articles:

- [Tutorial: Create a container image for deployment to Azure Container Instances](#)
- [Quickstart: Create a private container registry using the Azure CLI](#)
- [Tutorial: Prepare an application for Azure Kubernetes Service \(AKS\)](#)

Next steps

- [Install and run Speech containers](#)

Install and run Speech containers with Docker

07/01/2025

By using containers, you can use a subset of the Speech service features in your own environment. In this article, you learn how to download, install, and run a Speech container.

! Note

Disconnected container pricing and commitment tiers vary from standard containers. For more information, see [Speech service pricing](#).

Prerequisites

You must meet the following prerequisites before you use Speech service containers. If you don't have an Azure subscription, create a [free account](#) before you begin. You need:

- [Docker](#) installed on a host computer. Docker must be configured to allow the containers to connect with and send billing data to Azure.
 - On Windows, Docker must also be configured to support Linux containers.
 - You should have a basic understanding of [Docker concepts](#).
- A [Speech service resource](#) with the free (F0) or standard (S) [pricing tier](#).

Billing arguments

Speech containers aren't licensed to run without being connected to Azure for metering. You must configure your container to communicate billing information with the metering service always.

Three primary parameters for all Azure AI containers are required. The Microsoft Software License Terms must be present with a value of `accept`. An Endpoint URI and API key are also needed.

Queries to the container are billed at the pricing tier of the Azure resource that's used for the `ApiKey` parameter.

The `docker run` command starts the container when all three of the following options are provided with valid values:

Option	Description
ApiKey	The API key of the Speech resource that's used to track billing information. The <code>ApiKey</code> value is used to start the container and is available on the Azure portal's Keys page of the corresponding Speech resource. Go to the Keys page, and select the Copy to clipboard  icon.
Billing	The endpoint of the Speech resource that's used to track billing information. The endpoint is available on the Azure portal Overview page of the corresponding Speech resource. Go to the Overview page, hover over the endpoint, and a Copy to clipboard  icon appears. Copy and use the endpoint where needed.
Eula	Indicates that you accepted the license for the container. The value of this option must be set to <code>accept</code> .

Important

These subscription keys are used to access your Azure AI services API. Don't share your keys. Store them securely. For example, use Azure Key Vault. We also recommend that you regenerate these keys regularly. Only one key is necessary to make an API call. When you regenerate the first key, you can use the second key for continued access to the service.

The container needs the billing argument values to run. These values allow the container to connect to the billing endpoint. The container reports usage about every 10 to 15 minutes. If the container doesn't connect to Azure within the allowed time window, the container continues to run but doesn't serve queries until the billing endpoint is restored. The connection is attempted 10 times at the same time interval of 10 to 15 minutes. If it can't connect to the billing endpoint within the 10 tries, the container stops serving requests. For an example of the information sent to Microsoft for billing, see the [Azure AI container FAQ](#) in the Azure AI services documentation.

For more information about these options, see [Configure containers](#).

Container requirements and recommendations

The following table describes the minimum and recommended allocation of resources for each Speech container:

Container	Minimum	Recommended	Speech Model
Speech to text	4 core, 4-GB memory	8 core, 8-GB memory	+4 to 8 GB memory
Custom speech to text	4 core, 4-GB memory	8 core, 8-GB memory	+4 to 8 GB memory
Speech language identification	1 core, 1-GB memory	1 core, 1-GB memory	n/a
Neural text to speech	6 core, 12-GB memory	8 core, 16-GB memory	n/a

Each core must be at least 2.6 gigahertz (GHz) or faster.

Core and memory correspond to the `--cpus` and `--memory` settings, which are used as part of the `docker run` command.

(!) Note

The minimum and recommended allocations are based on Docker limits, *not* the host machine resources. For example, speech to text containers memory map portions of a large language model. We recommend that the entire file should fit in memory. You need to add an additional 4 to 8 GB to load the speech models (see the previous table). Also, the first run of either container might take longer because models are being paged into memory.

Host computer requirements and recommendations

The host is an x64-based computer that runs the Docker container. It can be a computer on your premises or a Docker hosting service in Azure, such as:

- [Azure Kubernetes Service](#).
- [Azure Container Instances](#).
- A [Kubernetes](#) cluster deployed to [Azure Stack](#). For more information, see [Deploy Kubernetes to Azure Stack](#).

(!) Note

Containers support compressed audio input to the Speech SDK by using GStreamer. To install GStreamer in a container, follow Linux instructions for GStreamer in [Use codec compressed audio input with the Speech SDK](#).

Advanced Vector Extension support

The *host* is the computer that runs the Docker container. The host *must support Advanced Vector Extensions* (AVX2). You can check for AVX2 support on Linux hosts with the following command:

Console

```
grep -q avx2 /proc/cpuinfo && echo AVX2 supported || echo No AVX2 support detected
```

⚠ Warning

The host computer is *required* to support AVX2. The container *will not* function correctly without AVX2 support.

Run the container

Use the [docker run](#) command to run the container. Once running, the container continues to run until you [stop the container](#).

Take note the following best practices with the `docker run` command:

- **Line-continuation character:** The Docker commands in the following sections use the back slash, `\`, as a line continuation character. Replace or remove this character based on your host operating system's requirements.
- **Argument order:** Don't change the order of the arguments unless you're familiar with Docker containers.

You can use the [docker images](#) command to list your downloaded container images. The following command lists the ID, repository, and tag of each downloaded container image, formatted as a table:

Bash

```
docker images --format "table {{.ID}}\t{{.Repository}}\t{{.Tag}}"
```

Here's an example result:

IMAGE ID	REPOSITORY	TAG

<image-id> <repository-path/name> <tag-name>

Validate that a container is running

There are several ways to validate that the container is running. Locate the *External IP* address and exposed port of the container in question, and open your favorite web browser. Use the various request URLs that follow to validate the container is running.

The example request URLs listed here are `http://localhost:5000`, but your specific container might vary. Make sure to rely on your container's *External IP* address and exposed port.

 Expand table

Request URL	Purpose
<code>http://localhost:5000/</code>	The container provides a home page.
<code>http://localhost:5000/ready</code>	Requested with GET, this URL provides a verification that the container is ready to accept a query against the model. This request can be used for Kubernetes liveness and readiness probes .
<code>http://localhost:5000/status</code>	Also requested with GET, this URL verifies if the api-key used to start the container is valid without causing an endpoint query. This request can be used for Kubernetes liveness and readiness probes .
<code>http://localhost:5000/swagger</code>	The container provides a full set of documentation for the endpoints and a Try it out feature. With this feature, you can enter your settings into a web-based HTML form and make the query without having to write any code. After the query returns, an example CURL command is provided to demonstrate the HTTP headers and body format that's required.

Stop the container

To shut down the container, in the command-line environment where the container is running, select `Ctrl+C`.

Run multiple containers on the same host

If you intend to run multiple containers with exposed ports, make sure to run each container with a different exposed port. For example, run the first container on port 5000 and the second container on port 5001.

You can have this container and a different Azure AI container running on the HOST together. You also can have multiple containers of the same Azure AI container running.

Host URLs

 Note

Use a unique port number if you're running multiple containers.

 Expand table

Protocol	Host URL	Containers
WS	<code>ws://localhost:5000</code>	Speech to text
		Custom speech to text
HTTP	<code>http://localhost:5000</code>	Neural text to speech
		Speech language identification

For more information on using WSS and HTTPS protocols, see [Container security](#) in the Azure AI services documentation.

Troubleshooting

When you start or run the container, you might experience issues. Use an output `mount` and enable logging. Doing so allows the container to generate log files that are helpful when you troubleshoot issues.

 Tip

For more troubleshooting information and guidance, see [Azure AI containers frequently asked questions \(FAQ\)](#) in the Azure AI services documentation.

Logging settings

Speech containers come with ASP.NET Core logging support. Here's an example of the `neural-text-to-speech container` started with default logging to the console:

Bash

```
docker run --rm -it -p 5000:5000 --memory 12g --cpus 6 \
mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-speech \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY} \
Logging:Console:LogLevel:Default=Information
```

For more information about logging, see [Configure Speech containers](#) and [usage records](#) in the Azure AI services documentation.

Microsoft diagnostics container

If you're having trouble running an Azure AI container, you can try using the Microsoft diagnostics container. Use this container to diagnose common errors in your deployment environment that might prevent Azure AI containers from functioning as expected.

To get the container, use the following `docker pull` command:

Bash

```
docker pull mcr.microsoft.com/azure-cognitive-services/diagnostic
```

Then run the container. Replace `{ENDPOINT_URI}` with your endpoint, and replace `{API_KEY}` with your key to your resource:

Bash

```
docker run --rm mcr.microsoft.com/azure-cognitive-services/diagnostic \
eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

The container tests for network connectivity to the billing endpoint.

Run disconnected containers

To run disconnected containers (not connected to the internet), you must submit [this request form](#) and wait for approval. For more information about applying and purchasing a commitment plan to use containers in disconnected environments, see [Use containers in disconnected environments](#) in the Azure AI services documentation.

Next steps

- Review [configure containers](#) for configuration settings.
- Learn how to [use Speech service containers with Kubernetes and Helm](#).
- Deploy and run containers on [Azure Container Instance](#)
- Use more [Azure AI containers](#).

Speech to text containers with Docker

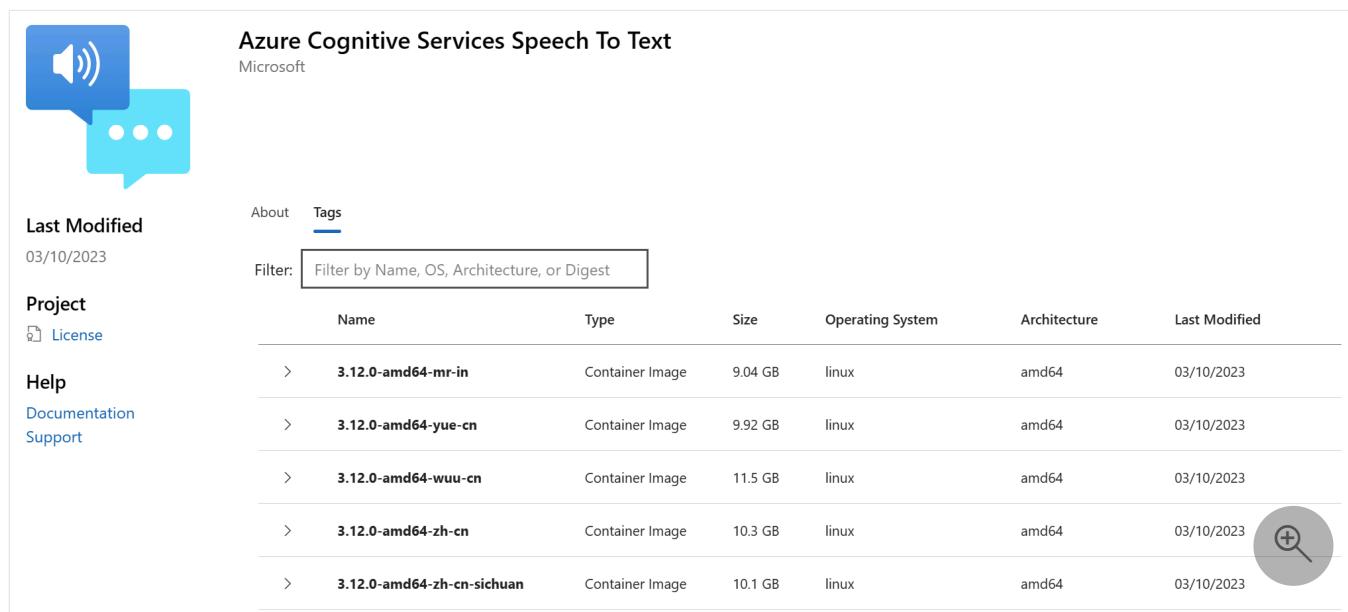
07/01/2025

The Speech to text container transcribes real-time speech or batch audio recordings with intermediate results. In this article, you learn how to download, install, and run a speech to text container.

For more information about prerequisites, validating that a container is running, running multiple containers on the same host, and running disconnected containers, see [Install and run Speech containers with Docker](#).

Container images

The Speech to text container image for all supported versions and locales can be found on the [Microsoft Container Registry \(MCR\)](#) syndicate. It resides within the `azure-cognitive-services/speechservices/` repository and is named `speech-to-text`.



A screenshot of the Microsoft Container Registry (MCR) interface. The page title is "Azure Cognitive Services Speech To Text" under the "Microsoft" organization. On the left, there are navigation links for "Last Modified", "Project", "Help", and "Documentation Support". The main content area has tabs for "About" and "Tags", with "Tags" being active. A search bar labeled "Filter by Name, OS, Architecture, or Digest" is present. Below the filter is a table listing five container images:

	Name	Type	Size	Operating System	Architecture	Last Modified
>	3.12.0-amd64-mr-in	Container Image	9.04 GB	linux	amd64	03/10/2023
>	3.12.0-amd64-yue-cn	Container Image	9.92 GB	linux	amd64	03/10/2023
>	3.12.0-amd64-wuu-cn	Container Image	11.5 GB	linux	amd64	03/10/2023
>	3.12.0-amd64-zh-cn	Container Image	10.3 GB	linux	amd64	03/10/2023
>	3.12.0-amd64-zh-cn-sichuan	Container Image	10.1 GB	linux	amd64	03/10/2023

The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text`. Either append a specific version or append `:latest` to get the most recent version.

[Expand table](#)

Version	Path
Latest	<code>mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text:latest</code>

The `latest` tag pulls the latest image for the `en-us` locale.

Version	Path
4.12.0	mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text:4.12.0-amd64-mr-in

All tags, except for `latest`, are in the following format and are case sensitive:

```
<major>.<minor>.<patch>-<platform>-<locale>-<prerelease>
```

The tags are also available [in JSON format](#) for your convenience. The body includes the container path and list of tags. The tags aren't sorted by version, but `"latest"` is always included at the end of the list as shown in this snippet:

JSON

```
{  
  "name": "azure-cognitive-services/speechservices/speech-to-text",  
  "tags": [  
    <--redacted for brevity-->  
    "4.12.0-amd64-sw-tz",  
    "4.12.0-amd64-ta-in",  
    "4.12.0-amd64-th-th",  
    "4.12.0-amd64-tr-tr",  
    "4.12.0-amd64-vi-vn",  
    "4.12.0-amd64-wuu-cn",  
    "4.12.0-amd64-yue-cn",  
    "4.12.0-amd64-zh-cn",  
    "4.12.0-amd64-zh-cn-sichuan",  
    "4.12.0-amd64-zh-hk",  
    "4.12.0-amd64-zh-tw",  
    "4.12.0-amd64-zu-za",  
    "latest"  
  ]  
}
```

Get the container image with docker pull

You need the [prerequisites](#) including required hardware. Also see the [recommended allocation of resources](#) for each Speech container.

Use the [docker pull](#) command to download a container image from Microsoft Container Registry:

Bash

```
docker pull mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text:latest
```

Important

The `latest` tag pulls the latest image for the `en-US` locale. For additional versions and locales, see [speech to text container images](#).

Run the container with docker run

Use the [docker run](#) command to run the container.

Speech to text

The following table represents the various `docker run` parameters and their corresponding descriptions:

 Expand table

Parameter	Description
<code>{ENDPOINT_URI}</code>	The endpoint is required for metering and billing. For more information, see billing arguments .
<code>{API_KEY}</code>	The API key is required. For more information, see billing arguments .

When you run the speech to text container, configure the port, memory, and CPU according to the speech to text container [requirements and recommendations](#).

Here's an example `docker run` command with placeholder values. You must specify the `ENDPOINT_URI` and `API_KEY` values:

Bash

```
docker run --rm -it -p 5000:5000 --memory 8g --cpus 4 \
mcr.microsoft.com/azure-cognitive-services/speechservices/speech-to-text \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

This command:

- Runs a `speech-to-text` container from the container image.
- Allocates 4 CPU cores and 8 GB of memory.
- Exposes TCP port 5000 and allocates a pseudo-TTY for the container.
- Automatically removes the container after it exits. The container image is still available on the host computer.

For more information about `docker run` with Speech containers, see [Install and run Speech containers with Docker](#).

Use the 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.

Important

When you use the Speech service with containers, be sure to use [host authentication](#). If you configure the key and region, requests will go to the public Speech service. Results from the Speech service might not be what you expect. Requests from disconnected containers will fail.

Instead of using this Azure-cloud initialization config:

```
C#  
  
var config = SpeechConfig.FromSubscription(...);
```

Use this config with the container [host](#):

```
C#  
  
var config = SpeechConfig.FromHost(  
    new Uri("ws://localhost:5000"));
```

Try the [speech to text quickstart](#) using host authentication instead of key and region.

Next steps

- See the [Speech containers overview](#)

- Review [configure containers](#) for configuration settings
- Use more [Azure AI containers](#)

Custom speech to text containers with Docker

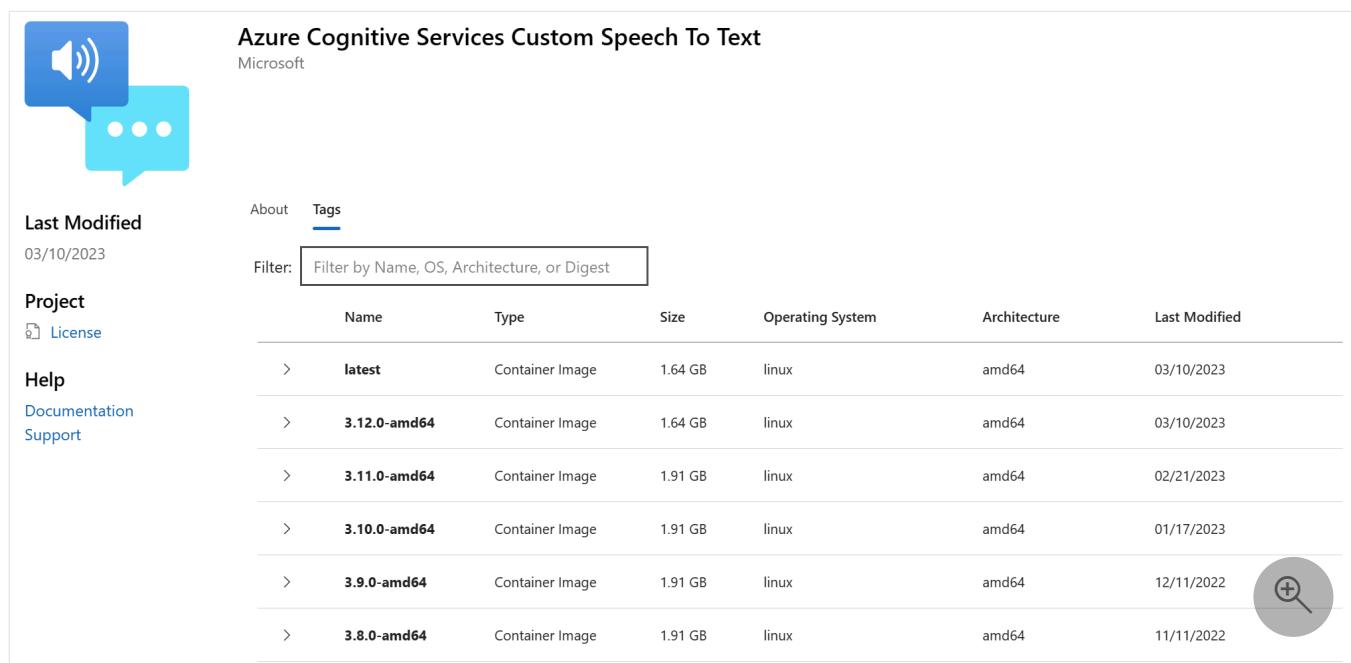
07/01/2025

The custom speech to text container transcribes real-time speech or batch audio recordings with intermediate results. You can use a custom model that you created in the [custom speech portal](#). In this article, you learn how to download, install, and run a custom speech to text container.

For more information about prerequisites, validating that a container is running, running multiple containers on the same host, and running disconnected containers, see [Install and run Speech containers with Docker](#).

Container images

The custom speech to text container image for all supported versions and locales can be found on the [Microsoft Container Registry \(MCR\)](#) syndicate. It resides within the `azure-cognitive-services/speechservices/` repository and is named `custom-speech-to-text`.



The screenshot shows the Microsoft Container Registry (MCR) interface. On the left, there's a sidebar with icons for Last Modified (03/10/2023), Project (with a License link), and Help (Documentation and Support). The main area has a title "Azure Cognitive Services Custom Speech To Text" and a Microsoft logo. Below the title, there are tabs for "About" and "Tags", with "Tags" being active. A search bar says "Filter by Name, OS, Architecture, or Digest". A table lists container images with columns: Name, Type, Size, Operating System, Architecture, and Last Modified. The table contains the following data:

	Name	Type	Size	Operating System	Architecture	Last Modified
>	latest	Container Image	1.64 GB	linux	amd64	03/10/2023
>	3.12.0-amd64	Container Image	1.64 GB	linux	amd64	03/10/2023
>	3.11.0-amd64	Container Image	1.91 GB	linux	amd64	02/21/2023
>	3.10.0-amd64	Container Image	1.91 GB	linux	amd64	01/17/2023
>	3.9.0-amd64	Container Image	1.91 GB	linux	amd64	12/11/2022
>	3.8.0-amd64	Container Image	1.91 GB	linux	amd64	11/11/2022

A magnifying glass icon is located in the bottom right corner of the table area.

The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text`. Either append a specific version or append `:latest` to get the most recent version.

[\[+\] Expand table](#)

Version	Path
Latest	mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text:latest
4.12.0	mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text:4.12.0-amd64

All tags, except for `latest`, are in the following format and are case sensitive:

```
<major>.<minor>.<patch>-<platform>-<prerelease>
```

ⓘ Note

The `locale` and `voice` for custom speech to text containers is determined by the custom model ingested by the container.

The tags are also available [in JSON format](#) for your convenience. The body includes the container path and list of tags. The tags aren't sorted by version, but `"latest"` is always included at the end of the list as shown in this snippet:

JSON

```
{
  "name": "azure-cognitive-services/speechservices/custom-speech-to-text",
  "tags": [
    <--redacted for brevity-->
    "4.0.0-amd64",
    "4.1.0-amd64",
    "4.10.0-amd64",
    "4.11.0-amd64",
    "4.12.0-amd64",
    "4.2.0-amd64",
    "4.3.0-amd64",
    "4.4.0-amd64",
    "4.5.0-amd64",
    "4.6.0-amd64",
    "4.7.0-amd64",
    "4.8.0-amd64",
    "4.9.0-amd64",
    "5.0.0-preview-amd64",
    "5.0.1-preview-amd64",
    "latest"
  ]
}
```

Get the container image with docker pull

You need the [prerequisites](#) including required hardware. Also see the [recommended allocation of resources](#) for each Speech container.

Use the [docker pull](#) command to download a container image from Microsoft Container Registry:

```
Bash
```

```
docker pull mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text:latest
```

! Note

The `locale` and `voice` for custom Speech containers is determined by the custom model ingested by the container.

Get the model ID

Before you can [run](#) the container, you need to know the model ID of your custom model or a base model ID. When you run the container, you specify one of the model IDs to download and use.

Custom model ID

The custom model must be [trained](#) by using the [Speech Studio](#). For information about how to get the model ID, see [custom speech model lifecycle](#).

The screenshot shows the Azure Cognitive Services Speech Studio interface. At the top, there's a navigation bar with 'Cognitive Services | Speech Studio'. Below it, a breadcrumb navigation shows 'Speech Studio > Custom Speech'. The main area has a title 'custom-speech-model English (United States)'. There are tabs for 'Data', 'Testing', 'Training' (which is underlined, indicating it's active), and 'Deployment'. A sub-instruction says 'Use your speech data to train a custom speech model. [Learn more](#)'. Below this, there's a 'Train model' button and some icons. A table lists the model details: Name (custom-speech-model), Description (Widgets custom speech ...), Baseline (20190701 (v4.17 Unified)), Created (10/14/2019 8:52 AM), and Status (Succeeded). The 'Status' column has an upward arrow icon.

Obtain the **Model ID** to use as the argument to the `ModelId` parameter of the `docker run` command.

The screenshot shows the 'Widgets custom speech model' page in the Speech Studio. At the top, there's a navigation bar with 'Cognitive Services | Speech Studio' and icons for notifications, settings, help, and JS. Below the navigation, the path is 'Speech Studio > Custom Speech > widgets-custom-speech > Training'. The main title is 'custom-speech-model'. Underneath it, it says 'Widgets custom speech model'. On the left, there's a 'Status' section with 'Succeeded' and a green checkmark. To its right is a 'Model ID' section with a red box around the text 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'. Further right is a 'Language' section with 'English (United States)'. Below this, there's a 'Training data' table with columns: 'Name ↑↓', 'Description ↑↓', 'Type ↑↓', and 'Created ↑↓'. A single row is shown: 'custom-speech-data' (Name), 'Custom speech voice and transcript' (Description), 'Audio + transcript' (Type), and '10/14/2019 8:50 AM' (Created).

Display model download

Before you [run](#) the container, you can optionally get the available display models information and choose to download those models into your speech to text container to get highly improved final display output. Display model download is available with custom-speech-to-text container version 3.1.0 and later.

! Note

Although you use the `docker run` command, the container isn't started for service.

You can query or download any or all of these display model types: Rescoring (`Rescore`), Punctuation (`Punct`), resegmentation (`Resegment`), and wfstitn (`Wfstitn`). Otherwise, you can use the `FullDisplay` option (with or without the other types) to query or download all types of display models.

Set the `BaseModelLocale` to query the latest available display model on the target locale. If you include multiple display model types, the command returns the latest available display models for each type. For example:

Bash

```
docker run --rm -it \
mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text \
Punct Rescore Resegment Wfstitn \ # Specify `FullDisplay` or a space-separated
subset of display models
BaseModelLocale={LOCALE} \
Eula=accept \
```

```
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

Set the `DisplayLocale` to download the latest available display model on the target locale.

When you set `DisplayLocale`, you must also specify `FullDisplay` or a space-separated subset of display models. The command downloads the latest available display model for each specified type. For example:

Bash

```
docker run --rm -it \
mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text \
Punct Rescore Resegment Wfsttin \ # Specify `FullDisplay` or a space-separated
subset of display models
DisplayLocale={LOCALE} \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

Set one model ID parameter to download a specific display model: Rescoring (`RescoreId`), Punctuation (`PuctId`), resegmentation (`ResegmentId`), or wfsttin (`WfsttinId`). This is similar to how you would download a base model via the `ModelId` parameter. For example, to download a rescoring display model, you can use the following command with the `RescoreId` parameter:

Bash

```
docker run --rm -it \
mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-text \
RescoreId={RESCORE_MODEL_ID} \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

! Note

If you set more than one query or download parameter, the command will prioritize in this order: `BaseModelLocale`, model ID, and then `DisplayLocale` (only applicable for display models).

Run the container with docker run

Use the [docker run](#) command to run the container for service.

The following table represents the various `docker run` parameters and their corresponding descriptions:

[Expand table](#)

Parameter	Description
{VOLUME_MOUNT}	The host computer volume mount , which Docker uses to persist the custom model. An example is <code>c:\CustomSpeech</code> where the <code>c:\</code> drive is located on the host machine.
{MODEL_ID}	The custom speech or base model ID. For more information, see Get the model ID .
{ENDPOINT_URI}	The endpoint is required for metering and billing. For more information, see billing arguments .
{API_KEY}	The API key is required. For more information, see billing arguments .

When you run the custom speech to text container, configure the port, memory, and CPU according to the custom speech to text container [requirements and recommendations](#).

Here's an example `docker run` command with placeholder values. You must specify the `VOLUME_MOUNT`, `MODEL_ID`, `ENDPOINT_URI`, and `API_KEY` values:

Bash

```
docker run --rm -it -p 5000:5000 --memory 8g --cpus 4 \
-v {VOLUME_MOUNT}:/usr/local/models \
mcr.microsoft.com/azure-cognitive-services/speechservices/custom-speech-to-
text \
ModelId={MODEL_ID} \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

This command:

- Runs a custom speech to text container from the container image.
- Allocates 4 CPU cores and 8 GB of memory.
- Loads the custom speech to text model from the volume input mount, for example, `C:\CustomSpeech`.
- Exposes TCP port 5000 and allocates a pseudo-TTY for the container.
- Downloads the model given the `ModelId` (if not found on the volume mount).
- If the custom model was previously downloaded, the `ModelId` is ignored.

- Automatically removes the container after it exits. The container image is still available on the host computer.

For more information about `docker run` with Speech containers, see [Install and run Speech containers with Docker](#).

Use the 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.

Important

When you use the Speech service with containers, be sure to use [host authentication](#). If you configure the key and region, requests will go to the public Speech service. Results from the Speech service might not be what you expect. Requests from disconnected containers will fail.

Instead of using this Azure-cloud initialization config:

```
C#  
  
var config = SpeechConfig.FromSubscription(...);
```

Use this config with the container [host](#):

```
C#  
  
var config = SpeechConfig.FromHost(  
    new Uri("ws://localhost:5000"));
```

[Try the speech to text quickstart](#) using host authentication instead of key and region.

Next steps

- See the [Speech containers overview](#)
- Review [configure containers](#) for configuration settings
- Use more [Azure AI containers](#)

Text to speech containers with Docker

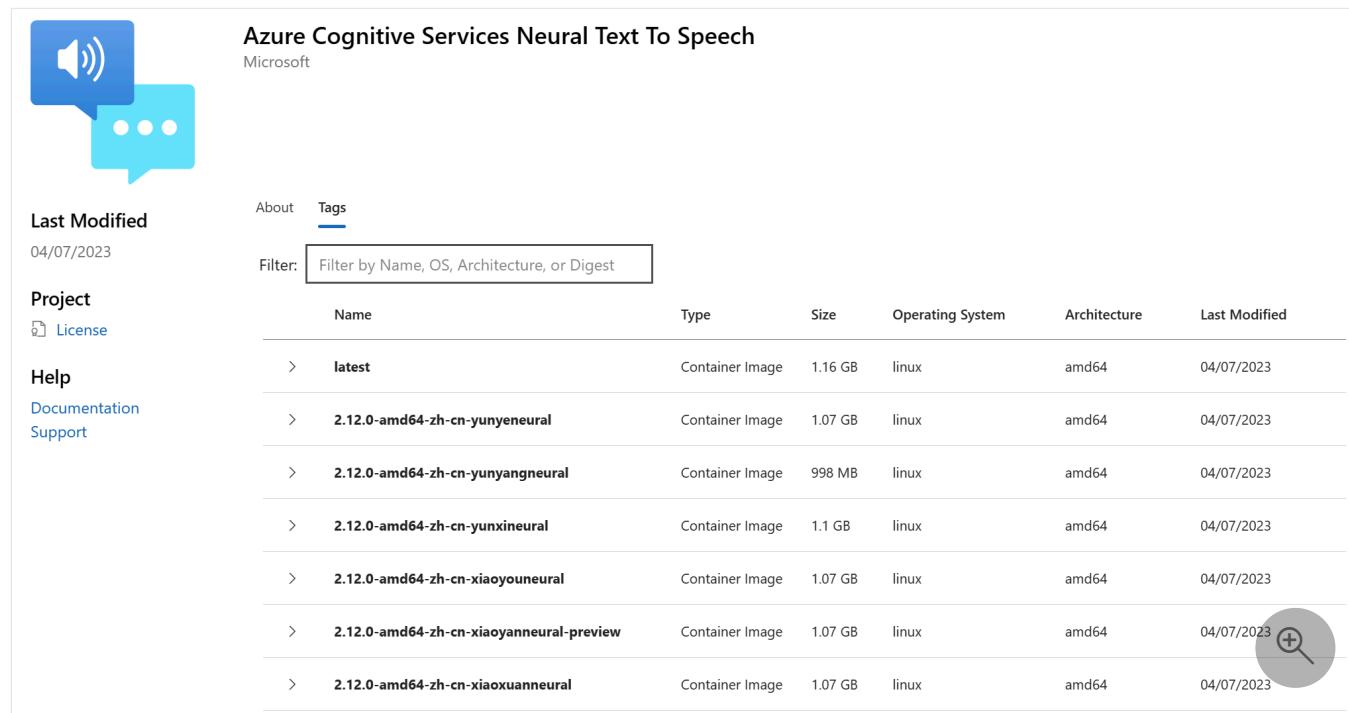
07/01/2025

The neural text to speech container converts text to natural-sounding speech by using deep neural network technology, which allows for more natural synthesized speech. In this article, you learn how to download, install, and run a Text to speech container.

For more information about prerequisites, validating that a container is running, running multiple containers on the same host, and running disconnected containers, see [Install and run Speech containers with Docker](#).

Container images

The neural text to speech container image for all supported versions and locales can be found on the [Microsoft Container Registry \(MCR\)](#) syndicate. It resides within the `azure-cognitive-services/speechservices/` repository and is named `neural-text-to-speech`.



Name	Type	Size	Operating System	Architecture	Last Modified
> <code>latest</code>	Container Image	1.16 GB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-yunyeneural</code>	Container Image	1.07 GB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-yunyangneural</code>	Container Image	998 MB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-yunxineural</code>	Container Image	1.1 GB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-xiaoyouneural</code>	Container Image	1.07 GB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-xiaoyanneural-preview</code>	Container Image	1.07 GB	linux	amd64	04/07/2023
> <code>2.12.0-amd64-zh-cn-xiaoxuanneural</code>	Container Image	1.07 GB	linux	amd64	04/07/2023

The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-speech`. Either append a specific version or append `:latest` to get the most recent version.

 [Expand table](#)

Version	Path
Latest	mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-speech:latest
	The <code>latest</code> tag pulls the <code>en-US</code> locale and <code>en-us-arianeural</code> voice.

All tags, except for `latest`, are in the following format and are case sensitive:

```
<major>.<minor>.<patch>-<platform>-<voice>-<preview>
```

The tags are also available [in JSON format](#) for your convenience. The body includes the container path and list of tags. The tags aren't sorted by version, but `"latest"` is always included at the end of the list as shown in this snippet:

JSON

```
{
  "name": "azure-cognitive-services/speechservices/neural-text-to-speech",
  "tags": [
    <--redacted for brevity-->
    "3.11.0-amd64-tr-tr-emelneural",
    "3.11.0-amd64-uk-ua-ostapneural",
    "3.11.0-amd64-zh-cn-xiaochenneural-preview",
    "3.11.0-amd64-zh-cn-xiaohanneural",
    "3.11.0-amd64-zh-cn-xiaomoneural",
    "3.11.0-amd64-zh-cn-xiaoqiuneural-preview",
    "3.11.0-amd64-zh-cn-xiaoruineural",
    "3.11.0-amd64-zh-cn-xiaoshuangneural-preview",
    "3.11.0-amd64-zh-cn-xiaoxiaoneural",
    "3.11.0-amd64-zh-cn-xiaoyanneural-preview",
    "3.11.0-amd64-zh-cn-xiaoyounneural",
    "3.11.0-amd64-zh-cn-yunxineural",
    "3.11.0-amd64-zh-cn-yunyangneural",
    "3.11.0-amd64-zh-cn-yunyeneural",
    "latest"
  ]
}
```

ⓘ Important

We retired the standard speech synthesis voices and standard [text to speech](#) container on August 31, 2021. You should use neural voices with the [neural-text-to-speech](#)

container version 3.0 and higher instead.

Starting from February 29, 2024, the text to speech and neural text to speech container versions 2.19 and earlier aren't supported.

Get the container image with docker pull

You need the [prerequisites](#) including required hardware. Also see the [recommended allocation of resources](#) for each Speech container.

Use the [docker pull](#) command to download a container image from Microsoft Container Registry:

Bash

```
docker pull mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-speech:latest
```

Important

The `latest` tag pulls the `en-us` locale and `en-us-arianeural` voice. For additional locales and voices, see [text to speech container images](#).

Run the container with docker run

Use the [docker run](#) command to run the container.

Neural text to speech

The following table represents the various `docker run` parameters and their corresponding descriptions:

 Expand table

Parameter	Description
<code>{ENDPOINT_URI}</code>	The endpoint is required for metering and billing. For more information, see billing arguments .
<code>{API_KEY}</code>	The API key is required. For more information, see billing arguments .

When you run the text to speech container, configure the port, memory, and CPU according to the text to speech container [requirements and recommendations](#).

Here's an example `docker run` command with placeholder values. You must specify the `ENDPOINT_URI` and `API_KEY` values:

Bash

```
docker run --rm -it -p 5000:5000 --memory 12g --cpus 6 \
mcr.microsoft.com/azure-cognitive-services/speechservices/neural-text-to-
speech \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

This command:

- Runs a neural text to speech container from the container image.
- Allocates 6 CPU cores and 12 GB of memory.
- Exposes TCP port 5000 and allocates a pseudo-TTY for the container.
- Automatically removes the container after it exits. The container image is still available on the host computer.

For more information about `docker run` with Speech containers, see [Install and run Speech containers with Docker](#).

Use the 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.

Important

When you use the Speech service with containers, be sure to use [host authentication](#). If you configure the key and region, requests will go to the public Speech service. Results from the Speech service might not be what you expect. Requests from disconnected containers will fail.

Instead of using this Azure-cloud initialization config:

```
C#
```

```
var config = SpeechConfig.FromSubscription(...);
```

Use this config with the container [host](#):

C#

```
var config = SpeechConfig.FromHost(  
    new Uri("http://localhost:5000"));
```

Try the [text to speech quickstart](#) using host authentication instead of key and region.

SSML voice element

When you construct a neural text to speech HTTP POST, the [SSML](#) message requires a `voice` element with a `name` attribute. The [locale of the voice](#) must correspond to the locale of the container model. The [SSML tag support](#) is consistent for [each text to speech voice](#) both in the Azure cloud and the container environment.

For example, a model that was downloaded via the `latest` tag (defaults to "en-US") would have a voice name of `en-US-AriaNeural`.

XML

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">  
    <voice name="en-US-AriaNeural">  
        This is the text that is spoken.  
    </voice>  
</speak>
```

Next steps

- See the [Speech containers overview](#)
- Review [configure containers](#) for configuration settings
- Use more [Azure AI containers](#)

Language identification containers with Docker

07/01/2025

The Speech language identification container detects the language spoken in audio files. You can get real-time speech or batch audio recordings with intermediate results. In this article, you learn how to download, install, and run a language identification container.

Note

The Speech language identification container is available in public preview. Containers in preview are still under development and don't meet Microsoft's stability and support requirements.

For more information about prerequisites, validating that a container is running, running multiple containers on the same host, and running disconnected containers, see [Install and run Speech containers with Docker](#).

Tip

To get the most useful results, use the Speech language identification container with the [speech to text](#) or [custom speech to text](#) containers.

Container images

The Speech language identification container image for all supported versions and locales can be found on the [Microsoft Container Registry \(MCR\)](#) syndicate. It resides within the `azure-cognitive-services/speechservices/` repository and is named `language-detection`.



Azure Cognitive Services Speech Language Detection

Microsoft

Last Modified: 03/15/2023

Project

- License

Help

- Documentation
- Support

About Tags Filter by Name, OS, Architecture, or Digest

	Name	Type	Size	Operating System	Architecture	Last Modified
>	1.11.0-amd64-preview	Container Image	642 MB	linux	amd64	03/15/2023
>	latest	Container Image	642 MB	linux	amd64	03/15/2023
>	1.8.0-amd64-preview	Container Image	638 MB	linux	amd64	05/09/2022
>	1.7.0-amd64-preview	Container Image	579 MB	linux	amd64	05/04/2022
>	1.6.1-amd64-preview	Container Image	527 MB	linux	amd64	03/21/2022
>	1.5.0-amd64-preview	Container Image	520 MB	linux	amd64	11/17/2021

The fully qualified container image name is, `mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection`. Either append a specific version or append `:latest` to get the most recent version.

[] [Expand table](#)

Version	Path
Latest	<code>mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection:latest</code>
1.18.0	<code>mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection:1.18.0-amd64-preview</code>

All tags, except for `latest`, are in the following format and are case sensitive:

```
<major>.<minor>.<patch>-<platform>-<prerelease>
```

The tags are also available [in JSON format](#) for your convenience. The body includes the container path and list of tags. The tags aren't sorted by version, but `"latest"` is always included at the end of the list as shown in this snippet:

JSON

```
{
  "name": "azure-cognitive-services/speechservices/language-detection",
  "tags": [
    "1.1.0-amd64-preview",
    "1.11.0-amd64-preview",
    "latest"
  ]
}
```

```
"1.12.0-amd64-preview",
"1.13.0-amd64-preview",
"1.14.0-amd64-preview",
"1.15.0-amd64-preview",
"1.16.0-amd64-preview",
"1.17.0-amd64-preview",
"1.18.0-amd64-preview",
"1.3.0-amd64-preview",
"1.5.0-amd64-preview",
"1.6.1-amd64-preview",
"1.7.0-amd64-preview",
"1.8.0-amd64-preview",
"latest"
]
}
```

Get the container image with docker pull

You need the [prerequisites](#) including required hardware. Also see the [recommended allocation of resources](#) for each Speech container.

Use the [docker pull](#) command to download a container image from Microsoft Container Registry:

Bash

```
docker pull mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection:latest
```

Run the container with docker run

Use the [docker run](#) command to run the container.

The following table represents the various `docker run` parameters and their corresponding descriptions:

[] [Expand table](#)

Parameter	Description
{ENDPOINT_URI}	The endpoint is required for metering and billing. For more information, see billing arguments .
{API_KEY}	The API key is required. For more information, see billing arguments .

When you run the Speech language identification container, configure the port, memory, and CPU according to the language identification container [requirements and recommendations](#).

Here's an example `docker run` command with placeholder values. You must specify the `ENDPOINT_URI` and `API_KEY` values:

Bash

```
docker run --rm -it -p 5000:5003 --memory 1g --cpus 1 \
mcr.microsoft.com/azure-cognitive-services/speechservices/language-detection \
Eula=accept \
Billing={ENDPOINT_URI} \
ApiKey={API_KEY}
```

This command:

- Runs a Speech language identification container from the container image.
- Allocates 1 CPU core and 1 GB of memory.
- Exposes TCP port 5000 and allocates a pseudo-TTY for the container.
- Automatically removes the container after it exits. The container image is still available on the host computer.

For more information about `docker run` with Speech containers, see [Install and run Speech containers with Docker](#).

Run with the speech to text container

If you want to run the language identification container with the `speech to text` container, you can use this [docker image](#). After both containers are started, use this `docker run` command to execute `speech-to-text-with-languagedetection-client`:

Bash

```
docker run --rm -v ${HOME}:/root -ti antsu/on-prem-client:latest ./speech-to-text-
with-languagedetection-client ./audio/LanguageDetection_en-us.wav --host localhost
--lport 5003 --sport 5000
```

Increasing the number of concurrent calls can affect reliability and latency. For language identification, we recommend a maximum of four concurrent calls using 1 CPU with 1 GB of memory. For hosts with 2 CPUs and 2 GB of memory, we recommend a maximum of six concurrent calls.

Use the 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.

Important

When you use the Speech service with containers, be sure to use [host authentication](#). If you configure the key and region, requests will go to the public Speech service. Results from the Speech service might not be what you expect. Requests from disconnected containers will fail.

Instead of using this Azure-cloud initialization config:

```
C#  
  
var config = SpeechConfig.FromSubscription(...);
```

Use this config with the container [host](#):

```
C#  
  
var config = SpeechConfig.FromHost(  
    new Uri("http://localhost:5000"));
```

Try [language identification](#) using host authentication instead of key and region. When you run language ID in a container, use the `SourceLanguageRecognizer` object instead of `SpeechRecognizer` or `TranslationRecognizer`.

Next steps

- See the [Speech containers overview](#)
- Review [configure containers](#) for configuration settings
- Use more [Azure AI containers](#)

Configure Speech service containers

07/01/2025

Speech containers enable customers to build one speech application architecture that is optimized to take advantage of both robust cloud capabilities and edge locality.

The Speech container runtime environment is configured using the `docker run` command arguments. This container has some required and optional settings. The container-specific settings are the billing settings.

Configuration settings

The container has the following configuration settings:

 Expand table

Required	Setting	Purpose
Yes	ApiKey	Tracks billing information.
No	ApplicationInsights	Enables adding Azure Application Insights telemetry support to your container.
Yes	Billing	Specifies the endpoint URI of the service resource on Azure.
Yes	Eula	Indicates that you've accepted the license for the container.
No	Fluentd	Writes log and, optionally, metric data to a Fluentd server.
No	HTTP Proxy	Configures an HTTP proxy for making outbound requests.
No	Logging	Provides ASP.NET Core logging support for your container.
No	Mounts	Reads and writes data from the host computer to the container and from the container back to the host computer.

Important

The [ApiKey](#), [Billing](#), and [Eula](#) settings are used together, and you must provide valid values for all three of them; otherwise your container won't start. For more information about using these configuration settings to instantiate a container, see [Billing](#).

ApiKey configuration setting

The `ApiKey` setting specifies the Azure resource key used to track billing information for the container. You must specify a value for the `ApiKey` and the value must be a valid key for the *Speech* resource specified for the [Billing](#) configuration setting.

This setting can be found in the following place:

- Azure portal: [Speech Resource Management](#), under **Keys**

ApplicationInsights setting

The `ApplicationInsights` setting allows you to add [Azure Application Insights](#) telemetry support to your container. Application Insights provides in-depth monitoring of your container. You can easily monitor your container for availability, performance, and usage. You can also quickly identify and diagnose errors in your container.

The following table describes the configuration settings supported under the `ApplicationInsights` section.

 [Expand table](#)

Required	Name	Data type	Description
No	<code>InstrumentationKey</code>	String	<p>The instrumentation key of the Application Insights instance to which telemetry data for the container is sent. For more information, see Application Insights for ASP.NET Core.</p> <p>Example: <code>InstrumentationKey=123456789</code></p>

Billing configuration setting

The `Billing` setting specifies the endpoint URI of the *Speech* resource on Azure used to meter billing information for the container. You must specify a value for this configuration setting, and the value must be a valid endpoint URI for a *Speech* resource on Azure. The container reports usage about every 10 to 15 minutes.

This setting can be found in the following place:

- Azure portal: Labeled `Endpoint` on the [Speech](#) overview page

 [Expand table](#)

Required	Name	Data type	Description
Yes	<code>Billing</code>	String	Billing endpoint URL. For more information on obtaining the billing URI, see billing . For more information and a complete list of regional endpoints, see Custom subdomain names for Azure AI services .

Eula setting

The `Eula` setting indicates that you've accepted the license for the container. You must specify a value for this configuration setting, and the value must be set to `accept`.

[+] [Expand table](#)

Required	Name	Data type	Description
Yes	<code>Eula</code>	String	<p>License acceptance</p> <p>Example: <code>Eula=accept</code></p>

Azure AI services containers are licensed under [your agreement](#) governing your use of Azure. If you do not have an existing agreement governing your use of Azure, you agree that your agreement governing use of Azure is the [Microsoft Online Subscription Agreement](#), which incorporates the [Online Services Terms](#). For previews, you also agree to the [Supplemental Terms of Use for Microsoft Azure Previews](#). By using the container you agree to these terms.

Fluentd settings

Fluentd is an open-source data collector for unified logging. The `Fluentd` settings manage the container's connection to a [Fluentd](#) server. The container includes a Fluentd logging provider, which allows your container to write logs and, optionally, metric data to a Fluentd server.

The following table describes the configuration settings supported under the `Fluentd` section.

[+] [Expand table](#)

Name	Data type	Description
<code>Host</code>	String	The IP address or DNS host name of the Fluentd server.

Name	Data type	Description
Port	Integer	The port of the Fluentd server. The default value is 24224.
HeartbeatMs	Integer	The heartbeat interval, in milliseconds. If no event traffic has been sent before this interval expires, a heartbeat is sent to the Fluentd server. The default value is 60000 milliseconds (1 minute).
SendBufferSize	Integer	The network buffer space, in bytes, allocated for send operations. The default value is 32768 bytes (32 kilobytes).
TlsConnectionEstablishmentTimeoutMs	Integer	The timeout, in milliseconds, to establish a SSL/TLS connection with the Fluentd server. The default value is 10000 milliseconds (10 seconds). If <code>UseTLS</code> is set to false, this value is ignored.
UseTLS	Boolean	Indicates whether the container should use SSL/TLS for communicating with the Fluentd server. The default value is false.

HTTP proxy credentials settings

If you need to configure an HTTP proxy for making outbound requests, use these two arguments:

[Expand table](#)

Name	Data type	Description
HTTP_PROXY	string	The proxy to use, for example, <code>http://proxy:8888 <proxy-url></code>
HTTP_PROXY_CREDS	string	Any credentials needed to authenticate against the proxy, for example, <code>username:password</code> . This value must be in lower-case .
<code><proxy-user></code>	string	The user for the proxy.
<code><proxy-password></code>	string	The password associated with <code><proxy-user></code> for the proxy.

Bash

```
docker run --rm -it -p 5000:5000 \
--memory 2g --cpus 1 \
--mount type=bind,src=/home/azureuser/output,target=/output \
<registry-location>/<image-name> \
Eula=accept \
Billing=<endpoint> \
ApiKey=<api-key> \
HTTP_PROXY=<proxy-url> \
HTTP_PROXY_CREDS=<proxy-user>:<proxy-password> \
```

Logging settings

The `Logging` settings manage ASP.NET Core logging support for your container. You can use the same configuration settings and values for your container that you use for an ASP.NET Core application.

The following logging providers are supported by the container:

[] [Expand table](#)

Provider	Purpose
Console	The ASP.NET Core <code>Console</code> logging provider. All of the ASP.NET Core configuration settings and default values for this logging provider are supported.
Debug	The ASP.NET Core <code>Debug</code> logging provider. All of the ASP.NET Core configuration settings and default values for this logging provider are supported.
Disk	The JSON logging provider. This logging provider writes log data to the output mount.

This container command stores logging information in the JSON format to the output mount:

Bash

```
docker run --rm -it -p 5000:5000 \
--memory 2g --cpus 1 \
--mount type=bind,src=/home/azureuser/output,target=/output \
<registry-location>/<image-name> \
Eula=accept \
Billing=<endpoint> \
ApiKey=<api-key> \
Logging:Disk:Format=json \
Mounts:Output=/output
```

This container command shows debugging information, prefixed with `dbug`, while the container is running:

Bash

```
docker run --rm -it -p 5000:5000 \
--memory 2g --cpus 1 \
<registry-location>/<image-name> \
Eula=accept \
Billing=<endpoint> \
ApiKey=<api-key> \
Logging:Console:LogLevel:Default=Debug
```

Disk logging

The `Disk` logging provider supports the following configuration settings:

[Expand table](#)

Name	Data type	Description
<code>Format</code>	String	<p>The output format for log files.</p> <p>Note: This value must be set to <code>json</code> to enable the logging provider. If this value is specified without also specifying an output mount while instantiating a container, an error occurs.</p>
<code>MaxFileSize</code>	Integer	<p>The maximum size, in megabytes (MB), of a log file. When the size of the current log file meets or exceeds this value, a new log file is started by the logging provider. If <code>-1</code> is specified, the size of the log file is limited only by the maximum file size, if any, for the output mount. The default value is <code>1</code>.</p>

For more information about configuring ASP.NET Core logging support, see [Settings file configuration](#).

Mount settings

Use bind mounts to read and write data to and from the container. You can specify an input mount or output mount by specifying the `--mount` option in the [docker run](#) command.

The Standard Speech containers don't use input or output mounts to store training or service data. However, custom speech containers rely on volume mounts.

The exact syntax of the host mount location varies depending on the host operating system. Additionally, the [host computer](#)'s mount location might not be accessible due to a conflict between permissions used by the docker service account and the host mount location permissions.

Optional	Name	Data type	Description
Not allowed	Input	String	Standard Speech containers don't use this. Custom speech containers use volume mounts .
Optional	Output	String	The target of the output mount. The default value is <code>/output</code> . This is the location of the logs. This includes container logs.

Example:
`--mount type=bind,src=c:\output,target=/output`

Volume mount settings

The custom speech containers use [volume mounts](#) to persist custom models. You can specify a volume mount by adding the `-v` (or `--volume`) option to the [docker run](#) command.

 Note

The volume mount settings are only applicable for [custom speech to text](#) containers.

Custom models are downloaded the first time that a new model is ingested as part of the custom speech container `docker run` command. Sequential runs of the same `ModelId` for a custom speech container uses the previously downloaded model. If the volume mount isn't provided, custom models can't be persisted.

The volume mount setting consists of three color : separated fields:

1. The first field is the name of the volume on the host machine, for example `C:\input`.
2. The second field is the directory in the container, for example `/usr/local/models`.
3. The third field (optional) is a comma-separated list of options, for more information, see [use volumes](#).

Here's a volume mount example that mounts the host machine `C:\input` directory to the containers `/usr/local/models` directory.

Bash

```
-v C:\input:/usr/local/models
```

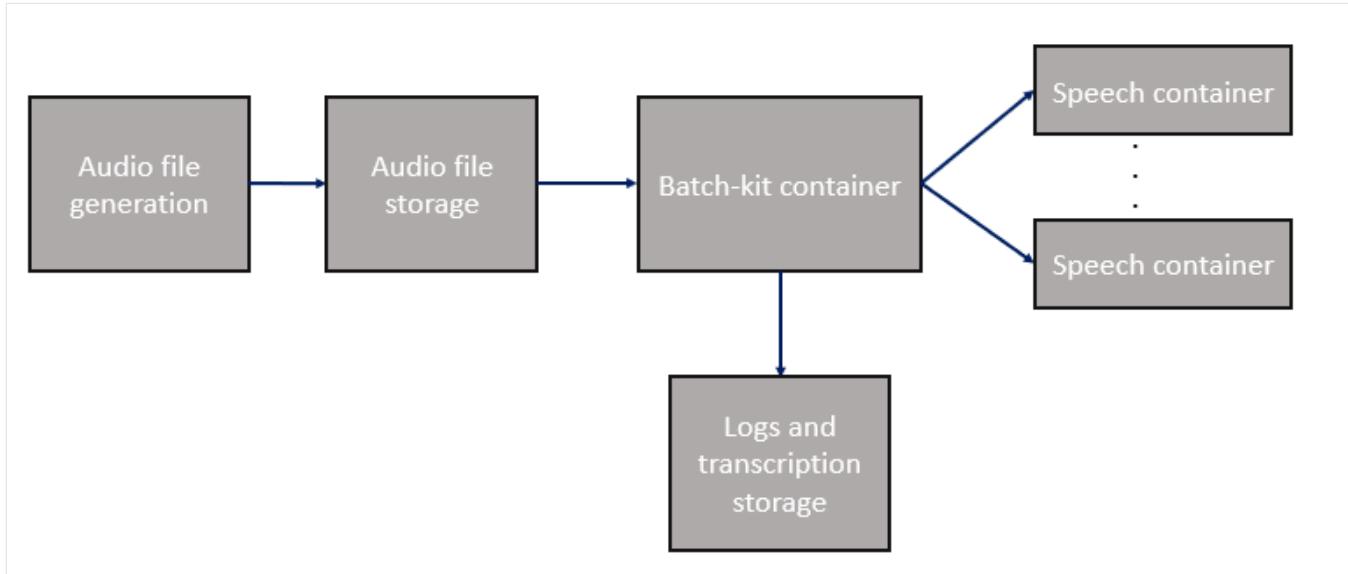
Next steps

- Review [How to install and run containers](#)

Batch processing kit for Speech containers

07/01/2025

Use the batch processing kit to complement and scale out workloads on Speech containers. Available as a container, this open-source utility helps facilitate batch transcription for large numbers of audio files, across any number of on-premises and cloud-based speech container endpoints.



The batch kit container is available for free on [GitHub](#) and [Docker hub](#). You're only billed for the Speech containers you use.

[] Expand table

Feature	Description
Batch audio file distribution	Automatically dispatch large numbers of files to on-premises or cloud-based Speech container endpoints. Files can be on any POSIX-compliant volume, including network filesystems.
Speech SDK integration	Pass common flags to the Speech SDK, including: n-best hypotheses, diarization, language, profanity masking.
Run modes	Run the batch client once, continuously in the background, or create HTTP endpoints for audio files.
Fault tolerance	Automatically retry and continue transcription without losing progress, and differentiate between which errors can, and can't be retried on.
Endpoint availability detection	If an endpoint becomes unavailable, the batch client continues transcribing, using other container endpoints. When the client is available, it automatically begins using the endpoint.

Feature	Description
Endpoint hot-swapping	Add, remove, or modify Speech container endpoints during runtime without interrupting the batch progress. Updates are immediate.
Real-time logging	Real-time logging of attempted requests, timestamps, and failure reasons, with Speech SDK log files for each audio file.

Get the container image with `docker pull`

Use the [docker pull](#) command to download the latest batch kit container.

! Note

The following example pulls a public container image from Docker Hub. We recommend that you authenticate with your Docker Hub account (`docker login`) first instead of making an anonymous pull request. To improve reliability when using public content, import and manage the image in a private Azure container registry. [Learn more about working with public images](#).

Bash

```
docker pull docker.io/batchkit/speech-batch-kit:latest
```

Endpoint configuration

The batch client takes a yaml configuration file that specifies the on-premises container endpoints. The following example can be written to `/mnt/my_nfs/config.yaml`, which is used in the following examples.

YAML

```
MyContainer1:
  concurrency: 5
  host: 192.168.0.100
  port: 5000
  rtf: 3
MyContainer2:
  concurrency: 5
  host: BatchVM0.corp.redmond.microsoft.com
  port: 5000
  rtf: 2
MyContainer3:
  concurrency: 10
```

```
host: localhost  
port: 6001  
rtf: 4
```

This yaml example specifies three speech containers on three hosts. The first host is specified by a IPv4 address, the second is running on the same VM as the batch-client, and the third container is specified by the DNS hostname of another VM. The `concurrency` value specifies the maximum concurrent file transcriptions that can run on the same container. The `rtf` (Real-Time Factor) value is optional and can be used to tune performance.

The batch client can dynamically detect if an endpoint becomes unavailable (for example, due to a container restart or networking issue), and when it becomes available again. Transcription requests aren't sent to containers that are unavailable, and the client continues using other available containers. You can add, remove, or edit endpoints at any time without interrupting the progress of your batch.

Run the batch processing container

ⓘ Note

- This example uses the same directory (`/my_nfs`) for the configuration file and the inputs, outputs, and logs directories. You can use hosted or NFS-mounted directories for these folders.
- Running the client with the `-h` flag lists the available command-line parameters, and their default values.
- The batch processing container is only supported on Linux.

Use the Docker `run` command to start the container. This command starts an interactive shell inside the container.

```
Bash
```

```
docker run --network host --rm -ti -v /mnt/my_nfs:/my_nfs --entrypoint /bin/bash  
/mnt/my_nfs:/my_nfs docker.io/batchkit/speech-batch-kit:latest
```

To run the batch client:

```
Bash
```

```
run-batch-client -config /my_nfs/config.yaml -input_folder /my_nfs/audio_files -  
output_folder /my_nfs/transcriptions -log_folder /my_nfs/logs -file_log_level  
DEBUG -nbest 1 -m ONESHOT -diarization None -language en-US -strict_config
```

To run the batch client and container in a single command:

Bash

```
docker run --network host --rm -ti -v /mnt/my_nfs:/my_nfs  
docker.io/batchkit/speech-batch-kit:latest -config /my_nfs/config.yaml -  
input_folder /my_nfs/audio_files -output_folder /my_nfs/transcriptions -log_folder  
/my_nfs/logs
```

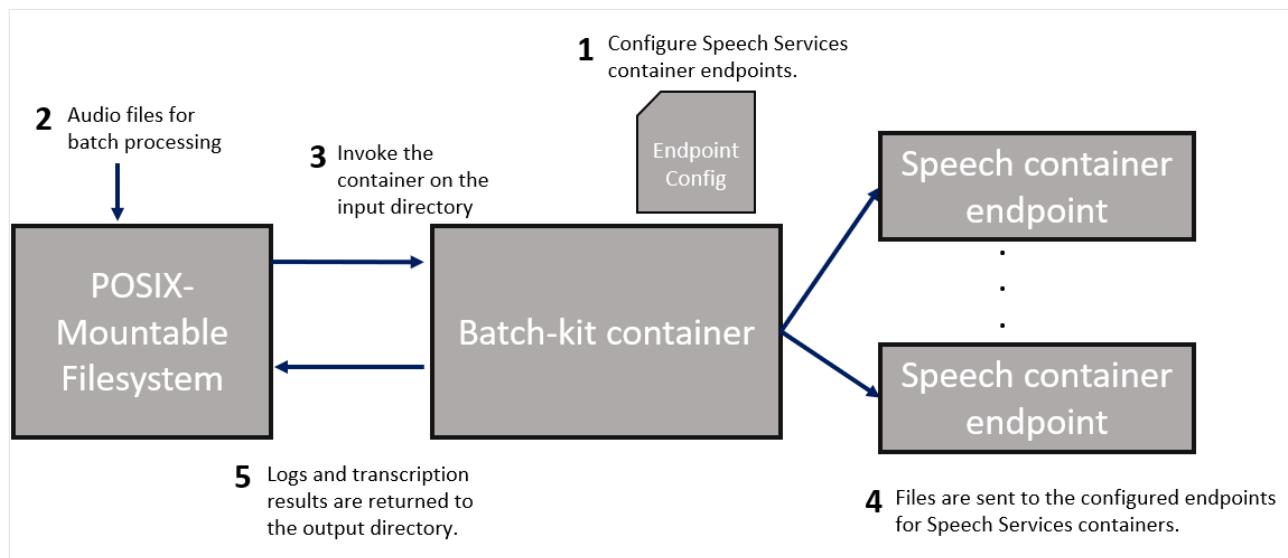
The client starts running. If an audio file was transcribed in a previous run, the client automatically skips the file. Files are sent with an automatic retry if transient errors occur, and you can differentiate between which errors you want the client to retry on. On a transcription error, the client continues transcription, and can retry without losing progress.

Run modes

The batch processing kit offers three modes, using the `--run-mode` parameter.

One-shot

`ONESHOT` mode transcribes a single batch of audio files (from an input directory and optional file list) to an output folder.



1. Define the Speech container endpoints that the batch client uses in the `config.yaml` file.
2. Place audio files for transcription in an input directory.

3. Invoke the container on the directory to begin processing the files. If the audio file is already transcribed in a previous run with the same output directory (same file name and checksum), the client skips the file.
4. The files are dispatched to the container endpoints from step 1.
5. Logs and the Speech container output are returned to the specified output directory.

Logging

 Note

The batch client may overwrite the *run.log* file periodically if it gets too large.

The client creates a *run.log* file in the directory specified by the `-log_folder` argument in the docker `run` command. Logs are captured at the DEBUG level by default. The same logs are sent to the `stdout/stderr`, and filtered depending on the `-file_log_level` or `console_log_level` arguments. This log is only necessary for debugging, or if you need to send a trace for support. The logging folder also contains the Speech SDK logs for each audio file.

The output directory specified by `-output_folder` contains a *run_summary.json* file, which is periodically rewritten every 30 seconds or whenever new transcriptions are finished. You can use this file to check on progress as the batch proceeds. It also contains the final run statistics and final status of every file when the batch is completed. The batch is completed when the process has a clean exit.

Next steps

- [How to install and run containers](#)

Use Speech service containers with Kubernetes and Helm

07/01/2025

One option to manage your Speech containers on-premises is to use Kubernetes and Helm. Using Kubernetes and Helm to define the speech to text and text to speech container images, we create a Kubernetes package. This package is deployed to a Kubernetes cluster on-premises. Finally, we explore how to test the deployed services and various configuration options. For more information about running Docker containers without Kubernetes orchestration, see [install and run Speech service containers](#).

Prerequisites

The following prerequisites before using Speech containers on-premises:

[] [Expand table](#)

Required	Purpose
Azure Account	If you don't have an Azure subscription, create a free account before you begin.
Container Registry access	In order for Kubernetes to pull the docker images into the cluster, it needs access to the container registry.
Kubernetes CLI	The Kubernetes CLI is required for managing the shared credentials from the container registry. Kubernetes is also needed before Helm, which is the Kubernetes package manager.
Helm CLI	Install the Helm CLI , which is used to install a helm chart (container package definition).
Speech resource	<p>In order to use these containers, you must have:</p> <p>A <i>Speech</i> Azure resource to get the associated billing key and billing endpoint URI. Both values are available on the Azure portal's <i>Speech</i> Overview and Keys pages and are required to start the container.</p> <p>{API_KEY}: resource key</p> <p>{ENDPOINT_URI}: endpoint URI example is: https://eastus.api.cognitive.microsoft.com/sts/v1.0</p>

The recommended host computer configuration

Refer to the [Speech service container host computer](#) details as a reference. This *helm chart* automatically calculates CPU and memory requirements based on how many decodes (concurrent requests) that the user specifies. Additionally, it adjusts based on whether optimizations for audio/text input are configured as `enabled`. The helm chart defaults to, two concurrent requests and disabling optimization.

[] [Expand table](#)

Service	CPU / Container	Memory / Container
speech to text	one decoder requires a minimum of 1,150 millicores. If the <code>optimizedForAudioFile</code> is enabled, then 1,950 millicores are required. (default: two decoders)	Required: 2 GB Limited: 4 GB
text to speech	one concurrent request requires a minimum of 500 millicores. If the <code>optimizeForTurboMode</code> is enabled, then 1,000 millicores are required. (default: two concurrent requests)	Required: 1 GB Limited: 2 GB

Connect to the Kubernetes cluster

The host computer is expected to have an available Kubernetes cluster. See this tutorial on [deploying a Kubernetes cluster](#) for a conceptual understanding of how to deploy a Kubernetes cluster to a host computer.

Configure Helm chart values for deployment

Visit the [Microsoft Helm Hub](#) for all the publicly available helm charts offered by Microsoft. From the Microsoft Helm Hub, you find the **Azure AI Speech On-Premises Chart**. The **Azure AI Speech On-Premises** is the chart we install, but we must first create a `config-values.yaml` file with explicit configurations. Let's start by adding the Microsoft repository to our Helm instance.

Console

```
helm repo add microsoft https://microsoft.github.io/charts/repo
```

Next, we configure our Helm chart values. Copy and paste the following YAML into a file named `config-values.yaml`. For more information on customizing the **Azure AI Speech On-Premises Helm Chart**, see [customize helm charts](#). Replace the `# {ENDPOINT_URI}` and `# {API_KEY}` comments with your own values.

YAML

```
# These settings are deployment specific and users can provide customizations
# speech to text configurations
speechToText:
  enabled: true
  numberOfConcurrentRequest: 3
  optimizeForAudioFile: true
  image:
    registry: mcr.microsoft.com
    repository: azure-cognitive-services/speechservices/speech-to-text
    tag: latest
    pullSecrets:
      - mcr # Or an existing secret
  args:
    eula: accept
    billing: # {ENDPOINT_URI}
    apikey: # {API_KEY}

# text to speech configurations
textToSpeech:
```

```
enabled: true
numberOfConcurrentRequest: 3
optimizeForTurboMode: true
image:
  registry: mcr.microsoft.com
  repository: azure-cognitive-services/speechservices/neural-text-to-speech
  tag: latest
  pullSecrets:
    - mcr # Or an existing secret
args:
  eula: accept
  billing: # {ENDPOINT_URI}
  apikey: # {API_KEY}
```

ⓘ Important

If the `billing` and `apikey` values are not provided, the services will expire after 15 min. Likewise, verification will fail as the services will not be available.

The Kubernetes package (Helm chart)

The *Helm chart* contains the configuration of which docker image(s) to pull from the `mcr.microsoft.com` container registry.

A [Helm chart](#) is a collection of files that describe a related set of Kubernetes resources. A single chart might be used to deploy something simple, like a memcached pod, or something complex, like a full web app stack with HTTP servers, databases, caches, and so on.

The provided *Helm charts* pull the docker images of the Speech service, both text to speech and the speech to text services from the `mcr.microsoft.com` container registry.

Install the Helm chart on the Kubernetes cluster

Run the [helm install](#) command to install the helm chart, replacing the `<config-values.yaml>` with the appropriate path and file name argument. The `microsoft/cognitive-services-speech-onpremise` Helm chart is available on the [Microsoft Helm Hub](#).

Console

```
helm install onprem-speech microsoft/cognitive-services-speech-onpremise \
--version 0.1.1 \
--values <config-values.yaml>
```

Here's an example output you might expect to see from a successful install execution:

Console

```
NAME: onprem-speech
LAST DEPLOYED: Tue Jul 2 12:51:42 2019
```

```
NAMESPACE: default
```

```
STATUS: DEPLOYED
```

```
RESOURCES:
```

```
--> v1/Pod(related)
```

NAME	READY	STATUS	RESTARTS	AGE
speech-to-text-7664f5f465-87w2d	0/1	Pending	0	0s
speech-to-text-7664f5f465-klbr8	0/1	ContainerCreating	0	0s
neural-text-to-speech-56f8fb685b-4jtzh	0/1	ContainerCreating	0	0s
neural-text-to-speech-56f8fb685b-frwxf	0/1	Pending	0	0s

```
--> v1/Service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
speech-to-text	LoadBalancer	10.0.252.106	<pending>	80:31811/TCP	1s
neural-text-to-speech	LoadBalancer	10.0.125.187	<pending>	80:31247/TCP	0s

```
--> v1beta1/PodDisruptionBudget
```

NAME	MIN	AVAILABLE	MAX	UNAVAILABLE	ALLOWED DISRUPTIONS	AGE
speech-to-text-poddisruptionbudget	N/A		20%		0	1s
neural-text-to-speech-poddisruptionbudget	N/A		20%		0	1s

```
--> v1beta2/Deployment
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
speech-to-text	0/2	2	0	0s
neural-text-to-speech	0/2	2	0	0s

```
--> v2beta2/HorizontalPodAutoscaler
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS
REPLICAS AGE speech-to-text-autoscaler 0s	Deployment/speech-to-text	<unknown>/50%	2	10 0
neural-text-to-speech-autoscaler 0s	Deployment/neural-text-to-speech	<unknown>/50%	2	10

NOTES:

cognitive-services-speech-onpremise has been installed!

Release is named onprem-speech

The Kubernetes deployment can take over several minutes to complete. To confirm that both pods and services are properly deployed and available, execute the following command:

```
Console
```

```
kubectl get all
```

You should expect to see something similar to the following output:

```
Console
```

NAME	READY	STATUS	RESTARTS	AGE
pod/speech-to-text-7664f5f465-87w2d	1/1	Running	0	34m
pod/speech-to-text-7664f5f465-klbr8	1/1	Running	0	34m
pod/neural-text-to-speech-56f8fb685b-4jtzh	1/1	Running	0	34m
pod/neural-text-to-speech-56f8fb685b-frwxf	1/1	Running	0	34m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	3h

service/speech-to-text	LoadBalancer	10.0.252.106	52.162.123.151	80:31811/TCP	34m
service/neural-text-to-speech	LoadBalancer	10.0.125.187	65.52.233.162	80:31247/TCP	
34m					
NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/speech-to-text	2	2	2	2	34m
deployment.apps/neural-text-to-speech	2	2	2	2	34m
NAME	DESIRED	CURRENT	READY	AGE	
replicaset.apps/speech-to-text-7664f5f465	2	2	2	2	34m
replicaset.apps/neural-text-to-speech-56f8fb685b	2	2	2	2	34m
NAME	REFERENCE				
TARGETS	MINPODS	MAXPODS	REPLICAS	AGE	
horizontalpodautoscaler.autoscaling/speech-to-text-autoscaler	Deployment/speech-to-text				
1%/50%	2	10	2	34m	
horizontalpodautoscaler.autoscaling/neural-text-to-speech-autoscaler	Deployment/neural-text-to-speech				
0%/50%	2	10	2	34m	

Verify Helm deployment with Helm tests

The installed Helm charts define *Helm tests*, which serve as a convenience for verification. These tests validate service readiness. To verify both speech to text and text to speech features, we execute the [Helm test](#) command.

Console

```
helm test onprem-speech
```

Important

These tests will fail if the POD status is not `Running` or if the deployment is not listed under the `AVAILABLE` column. Be patient as this can take over ten minutes to complete.

These tests output various status results:

Console

```
RUNNING: speech to text-readiness-test
PASSED: speech to text-readiness-test
RUNNING: text to speech-readiness-test
PASSED: text to speech-readiness-test
```

As an alternative to executing the *helm tests*, you could collect the *External IP* addresses and corresponding ports from the `kubectl get all` command. Using the IP and port, open a web browser and navigate to `http://<external-ip>:<port>/swagger/index.html` to view the API swagger page(s).

Customize Helm charts

Helm charts are hierarchical. Being hierarchical allows for chart inheritance, it also caters to the concept of specificity, where settings that are more specific override inherited rules.

Speech (umbrella chart)

Values in the top-level "umbrella" chart override the corresponding sub-chart values. Therefore, all on-premises customized values should be added here.

[\[\]](#) Expand table

Parameter	Description	Default
<code>speechToText.enabled</code>	Whether the speech to text service is enabled.	<code>true</code>
<code>speechToText.verification.enabled</code>	Whether the <code>helm test</code> capability for speech to text service is enabled.	<code>true</code>
<code>speechToText.verification.image.registry</code>	The docker image repository that <code>helm test</code> uses to test speech to text service. Helm creates separate pod inside the cluster for testing and pulls the <i>test-use</i> image from this registry.	<code>docker.io</code>
<code>speechToText.verification.image.repository</code>	The docker image repository that <code>helm test</code> uses to test speech to text service. Helm test pod uses this repository to pull <i>test-use</i> image.	<code>antsu/on-prem-client</code>
<code>speechToText.verification.image.tag</code>	The docker image tag used with <code>helm test</code> for speech to text service. Helm test pod uses this tag to pull <i>test-use</i> image.	<code>latest</code>
<code>speechToText.verification.image.pullByHash</code>	Whether the <i>test-use</i> docker image is pulled by hash. If <code>true</code> , <code>speechToText.verification.image.hash</code> should be added, with valid image hash value.	<code>false</code>
<code>speechToText.verification.image.arguments</code>	The arguments used to execute the <i>test-use</i> docker image. Helm test pod passes these arguments to the container when running <code>helm test</code> .	<code>./speech-to-text-client</code> <code>./audio/whatstheweatherlike.wav</code> <code>--expect=What's the weather like</code> <code>--host=\$(SPEECH_TO_TEXT_HOST)</code> <code>--port=\$(SPEECH_TO_TEXT_PORT)</code>
<code>textToSpeech.enabled</code>	Whether the text to speech service is enabled.	<code>true</code>
<code>textToSpeech.verification.enabled</code>	Whether the <code>helm test</code> capability for speech to text service is enabled.	<code>true</code>
<code>textToSpeech.verification.image.registry</code>	The docker image repository that <code>helm test</code> uses to test speech to text	<code>docker.io</code>

Parameter	Description	Default
	service. Helm creates separate pod inside the cluster for testing and pulls the <i>test-use</i> image from this registry.	
<code>textToSpeech.verification.image.repository</code>	The docker image repository that <code>helm test</code> uses to test speech to text service. Helm test pod uses this repository to pull <i>test-use</i> image.	<code>antsu/on-prem-client</code>
<code>textToSpeech.verification.image.tag</code>	The docker image tag used with <code>helm test</code> for speech to text service. Helm test pod uses this tag to pull <i>test-use</i> image.	<code>latest</code>
<code>textToSpeech.verification.image.pullByHash</code>	Whether the <i>test-use</i> docker image is pulled by hash. If <code>true</code> , <code>textToSpeech.verification.image.hash</code> should be added, with valid image hash value.	<code>false</code>
<code>textToSpeech.verification.image.arguments</code>	The arguments to execute with the <i>test-use</i> docker image. The helm test pod passes these arguments to container when running <code>helm test</code> .	<code>./text-to-speech-client --input='What's the weather like' --host=\$(TEXT_TO_SPEECH_HOST) --port=\$(TEXT_TO_SPEECH_PORT)</code>

Speech to text (sub-chart: charts/speechToText)

To override the "umbrella" chart, add the prefix `speechToText.` on any parameter to make it more specific.

For example, it will override the corresponding parameter for example,

`speechToText.numberOfConcurrentRequest` overrides `numberOfConcurrentRequest`.

[Expand table](#)

Parameter	Description	Default
<code>enabled</code>	Whether the speech to text service is enabled.	<code>false</code>
<code>numberOfConcurrentRequest</code>	The number of concurrent requests for the speech to text service. This chart automatically calculates CPU and memory resources, based on this value.	<code>2</code>
<code>optimizeForAudioFile</code>	Whether the service needs to optimize for audio input via audio files. If <code>true</code> , this chart will allocate more CPU resource to service.	<code>false</code>
<code>image.registry</code>	The speech to text docker image registry.	<code>containerpreview.azurecr.io</code>
<code>image.repository</code>	The speech to text docker image repository.	<code>microsoft/cognitive-services-speech-to-text</code>
<code>image.tag</code>	The speech to text docker image tag.	<code>latest</code>

Parameter	Description	Default
<code>image.pullSecrets</code>	The image secrets for pulling the speech to text docker image.	
<code>image.pullByHash</code>	Whether the docker image is pulled by hash. If <code>true</code> , <code>image.hash</code> is required.	<code>false</code>
<code>image.hash</code>	The speech to text docker image hash. Only used when <code>image.pullByHash: true</code> .	
<code>image.args.eula</code> (required)	Indicates you've accepted the license. The only valid value is <code>accept</code>	
<code>image.args.billing</code> (required)	The billing endpoint URI value is available on the Azure portal's Speech Overview page.	
<code>image.args.apikey</code> (required)	Used to track billing information.	
<code>service.type</code>	The Kubernetes service type of the speech to text service. See the Kubernetes service types instructions for more details and verify cloud provider support.	<code>LoadBalancer</code>
<code>service.port</code>	The port of the speech to text service.	<code>80</code>
<code>service.annotations</code>	The speech to text annotations for the service metadata. Annotations are key value pairs. <code>annotations:</code> <code>some/annotation1: value1</code> <code>some/annotation2: value2</code>	
<code>service.autoScaler.enabled</code>	Whether the Horizontal Pod Autoscaler is enabled. If <code>true</code> , the <code>speech-to-text-autoscaler</code> will be deployed in the Kubernetes cluster.	<code>true</code>
<code>service.podDisruption.enabled</code>	Whether the Pod Disruption Budget is enabled. If <code>true</code> , the <code>speech-to-text-poddisruptionbudget</code> will be deployed in the Kubernetes cluster.	<code>true</code>

Sentiment analysis (sub-chart: charts/speechToText)

Starting with v2.2.0 of the speech to text container and v0.2.0 of the Helm chart, the following parameters are used for sentiment analysis using the Language service API.

[!\[\]\(9f7854b68ee2721bbb438a838dde1d36_img.jpg\) Expand table](#)

Parameter	Description	Values	Default
<code>textanalytics.enabled</code>	Whether the text-analytics service is enabled	true/false	<code>false</code>
<code>textanalytics.image.registry</code>	The text-analytics docker image registry	valid docker image registry	

Parameter	Description	Values	Default
<code>textanalytics.image.repository</code>	The text-analytics docker image repository	valid docker image repository	
<code>textanalytics.image.tag</code>	The text-analytics docker image tag	valid docker image tag	
<code>textanalytics.image.pullSecrets</code>	The image secrets for pulling text-analytics docker image	valid secrets name	
<code>textanalytics.image.pullByHash</code>	Specifies if you are pulling docker image by hash. If <code>yes</code> , <code>image.hash</code> is required to have as well. If <code>no</code> , set it as 'false'. Default is <code>false</code> .	true/false	<code>false</code>
<code>textanalytics.image.hash</code>	The text-analytics docker image hash. Only use it with <code>image.pullByHash:true</code> .	valid docker image hash	
<code>textanalytics.image.args.eula</code>	One of the required arguments by text-analytics container, which indicates you've accepted the license. The value of this option must be: <code>accept</code> .	<code>accept</code> , if you want to use the container	
<code>textanalytics.image.args.billing</code>	One of the required arguments by text-analytics container, which specifies the billing endpoint URI. The billing endpoint URI value is available on the Azure portal's Speech Overview page.	valid billing endpoint URI	
<code>textanalytics.image.args.apikey</code>	One of the required arguments by text-analytics container, which is used to track billing information.	valid apikey	
<code>textanalytics.cpuRequest</code>	The requested CPU for text-analytics container	int	<code>3000m</code>
<code>textanalytics.cpuLimit</code>	The limited CPU for text-analytics container		<code>8000m</code>
<code>textanalytics.memoryRequest</code>	The requested memory for text-analytics container		<code>3Gi</code>
<code>textanalytics.memoryLimit</code>	The limited memory for text-analytics container		<code>8Gi</code>
<code>textanalytics.service.sentimentURISuffix</code>	The sentiment analysis URI suffix, the whole URI is in format <code>"http://<service>:<port>/<sentimentURISuffix>"</code> .		<code>text/analytics/v3.0-preview/sentiment</code>

Parameter	Description	Values	Default
<code>textanalytics.service.type</code>	The type of text-analytics service in Kubernetes. See Kubernetes service types	valid Kubernetes service type	<code>LoadBalancer</code>
<code>textanalytics.service.port</code>	The port of the text-analytics service	int	<code>50085</code>
<code>textanalytics.service.annotations</code>	The annotations users can add to text-analytics service metadata. For instance: annotations: <code>some/annotation1: value1</code> <code>some/annotation2: value2</code>	annotations, one per each line	
<code>textanalytics.servicce.autoScaler.enabled</code>	Whether Horizontal Pod Autoscaler is enabled. If enabled, <code>text-analytics-autoscaler</code> will be deployed in the Kubernetes cluster	true/false	<code>true</code>
<code>textanalytics.service.podDisruption.enabled</code>	Whether Pod Disruption Budget is enabled. If enabled, <code>text-analytics-poddisruptionbudget</code> will be deployed in the Kubernetes cluster	true/false	<code>true</code>

Text to speech (sub-chart: charts/textToSpeech)

To override the "umbrella" chart, add the prefix `textToSpeech.` on any parameter to make it more specific. For example, it will override the corresponding parameter for example, `textToSpeech.numberOfConcurrentRequest` overrides `numberOfConcurrentRequest`.

[\[\]](#) Expand table

Parameter	Description	Default
<code>enabled</code>	Whether the text to speech service is enabled.	<code>false</code>
<code>numberOfConcurrentRequest</code>	The number of concurrent requests for the text to speech service. This chart automatically calculates CPU and memory resources, based on this value.	<code>2</code>
<code>optimizeForTurboMode</code>	Whether the service needs to optimize for text input via text files. If <code>true</code> , this chart will allocate more CPU resource to service.	<code>false</code>
<code>image.registry</code>	The text to speech docker image registry.	<code>containerpreview.azurecr.io</code>
<code>image.repository</code>	The text to speech docker image repository.	<code>microsoft/cognitive-services-text-to-speech</code>

Parameter	Description	Default
<code>image.tag</code>	The text to speech docker image tag.	<code>latest</code>
<code>image.pullSecrets</code>	The image secrets for pulling the text to speech docker image.	
<code>image.pullByHash</code>	Whether the docker image is pulled by hash. If <code>true</code> , <code>image.hash</code> is required.	<code>false</code>
<code>image.hash</code>	The text to speech docker image hash. Only used when <code>image.pullByHash: true</code> .	
<code>image.args.eula</code> (required)	Indicates you've accepted the license. The only valid value is <code>accept</code>	
<code>image.args.billing</code> (required)	The billing endpoint URI value is available on the Azure portal's Speech Overview page.	
<code>image.args.apikey</code> (required)	Used to track billing information.	
<code>service.type</code>	The Kubernetes service type of the text to speech service. See the Kubernetes service types instructions for more details and verify cloud provider support.	<code>LoadBalancer</code>
<code>service.port</code>	The port of the text to speech service.	<code>80</code>
<code>service.annotations</code>	The text to speech annotations for the service metadata. Annotations are key value pairs. <code>annotations:</code> <code>some/annotation1: value1</code> <code>some/annotation2: value2</code>	
<code>service.autoScaler.enabled</code>	Whether the Horizontal Pod Autoscaler is enabled. If <code>true</code> , the <code>text-to-speech-autoscaler</code> will be deployed in the Kubernetes cluster.	<code>true</code>
<code>service.podDisruption.enabled</code>	Whether the Pod Disruption Budget is enabled. If <code>true</code> , the <code>text-to-speech-poddisruptionbudget</code> will be deployed in the Kubernetes cluster.	<code>true</code>

Next steps

For more details on installing applications with Helm in Azure Kubernetes Service (AKS), [visit here](#).

[Azure AI containers](#)

Deploy and run containers on Azure Container Instance

05/19/2025

With the following steps, scale Azure AI services applications in the cloud easily with Azure [Container Instances](#). Containerization helps you focus on building your applications instead of managing the infrastructure. For more information on using containers, see [features and benefits](#).

Prerequisites

The recipe works with any Azure AI services container. The Azure AI Foundry resource must be created before using the recipe. Each Azure AI service that supports containers has a "How to install" article for installing and configuring the service for a container. Some services require a file or set of files as input for the container, it is important that you understand and have used the container successfully before using this solution.

- An Azure resource for the Azure AI service that you're using.
- Azure resource **endpoint URL** - review your specific service's "How to install" for the container, to find where the endpoint URL is from within the Azure portal, and what a correct example of the URL looks like. The exact format can change from service to service.
- Azure resource **key** - the keys are on the [Keys](#) page for the Azure resource. You only need one of the two keys. The key is a string of 84 alpha-numeric characters.
- A single Azure AI services container on your local host (your computer). Make sure you can:
 - Pull down the image with a `docker pull` command.
 - Run the local container successfully with all required configuration settings with a `docker run` command.
 - Call the container's endpoint, getting a response of HTTP 2xx and a JSON response back.

All variables in angle brackets, `<>`, need to be replaced with your own values. This replacement includes the angle brackets.

Important

The LUIS container requires a `.gz` model file that is pulled in at runtime. The container must be able to access this model file via a volume mount from the container instance. To upload a model file, follow these steps:

1. [Create an Azure file share](#). Take note of the Azure Storage account name, key, and file share name as you'll need them later.
2. [export your LUIS model \(packaged app\) from the LUIS portal](#).
3. In the Azure portal, navigate to the **Overview** page of your storage account resource, and select **File shares**.
4. Select the file share name that you recently created, then select **Upload**. Then upload your packaged app.

Azure portal

Create an Azure Container Instance resource using the Azure portal

1. Go to the [Create](#) page for Container Instances.
2. On the **Basics** tab, enter the following details:

[Expand table](#)

Setting	Value
Subscription	Select your subscription.
Resource group	Select the available resource group or create a new one such as <code>cognitive-services</code> .
Container name	Enter a name such as <code>cognitive-container-instance</code> . The name must be in lower caps.
Location	Select a region for deployment.
Image type	If your container image is stored in a container registry that doesn't require credentials, choose <code>Public</code> . If accessing your container image requires credentials, choose <code>Private</code> . Refer to container repositories and images for details on whether or not the container image is <code>Public</code> or <code>Private</code> ("Public Preview").

Setting	Value
Image name	<p>Enter the Azure AI services container location. The location is what's used as an argument to the <code>docker pull</code> command. Refer to the container repositories and images for the available image names and their corresponding repository.</p> <p>The image name must be fully qualified specifying three parts. First, the container registry, then the repository, finally the image name: <container-registry>/<repository>/<image-name>.</p> <p>Here is an example, <code>mcr.microsoft.com/azure-cognitive-services/keyphrase</code> would represent the Key Phrase Extraction image in the Microsoft Container Registry under the Azure AI services repository. Another example is, <code>containerpreview.azurecr.io/microsoft/cognitive-services-speech-to-text</code> which would represent the Speech to text image in the Microsoft repository of the Container Preview container registry.</p>
OS type	Linux
Size	<p>Change size to the suggested recommendations for your specific Azure AI container:</p> <ul style="list-style-type: none"> 2 CPU cores 4 GB

3. On the **Networking** tab, enter the following details:

[+] [Expand table](#)

Setting	Value
Ports	Set the TCP port to <code>5000</code> . Exposes the container on port 5000.

4. On the **Advanced** tab, enter the required **Environment Variables** for the container billing settings of the Azure Container Instance resource:

[+] [Expand table](#)

Key	Value
ApiKey	Copied from the Keys and endpoint page of the resource. It is a 84 alphanumeric-character string with no spaces or dashes, <code>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</code> .
Billing	Your endpoint URL copied from the Keys and endpoint page of the resource.
Eula	<code>accept</code>

5. Select **Review and Create**

6. After validation passes, click **Create** to finish the creation process

7. When the resource is successfully deployed, it's ready

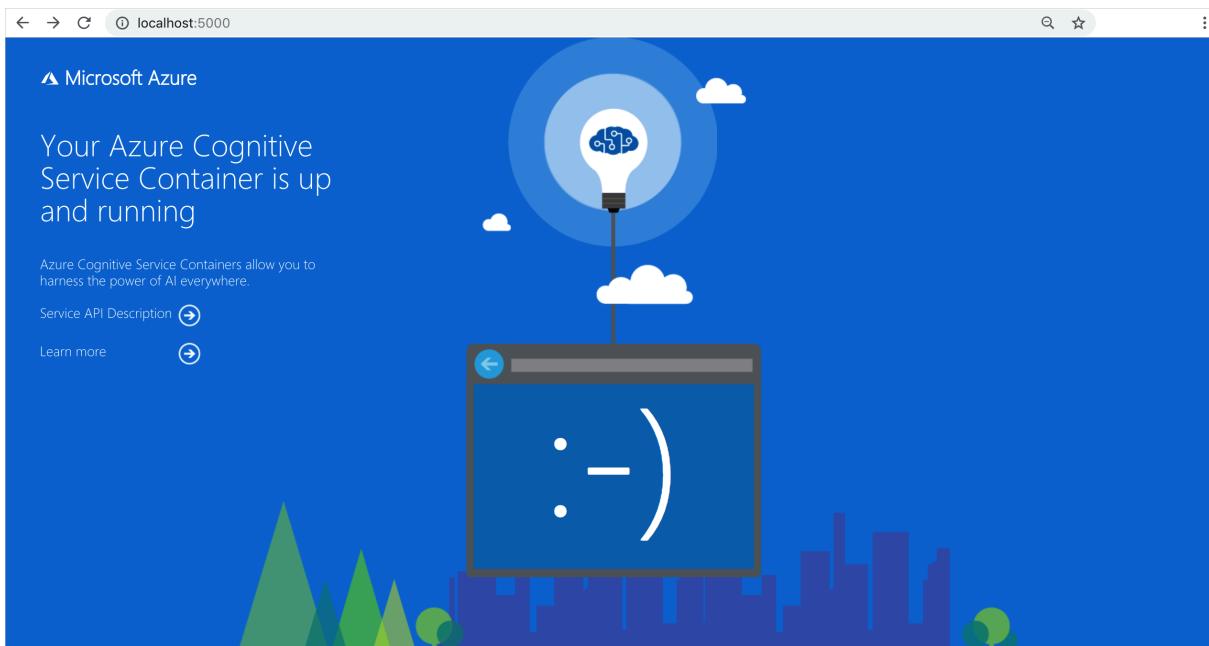
Use the Container Instance

Azure portal

1. Select the **Overview** and copy the IP address. It will be a numeric IP address such as

55.55.55.55.

2. Open a new browser tab and use the IP address, for example, `http://<IP-address>:5000` (`http://55.55.55.55:5000`). You will see the container's home page, letting you know the container is running.



3. Select **Service API Description** to view the swagger page for the container.

4. Select any of the **POST** APIs and select **Try it out**. The parameters are displayed including the input. Fill in the parameters.

5. Select **Execute** to send the request to your Container Instance.

You have successfully created and used Azure AI containers in Azure Container Instance.

Speech service containers frequently asked questions (FAQ)

When using the Speech service with containers, you can refer to these frequently asked questions before contacting support. This article captures questions of varying degree, from general to technical.

General questions

How do Speech containers work and how do I set them up?

For an overview, see [Install and run Speech containers](#). When setting up the production cluster, there are several things to consider. First, setting up single language, multiple containers, on the same machine, shouldn't be a large issue. If you're experiencing problems, it might be a hardware-related issue - so we would first look at resource, that is; CPU and memory specifications.

Consider for a moment, the `ja-JP` container and latest model. The acoustic model is the most demanding piece CPU-wise, while the language model demands the most memory. When we benchmarked the use, it takes about 0.6 CPU cores to process a single speech to text request when audio is flowing in at real-time, for example, from a microphone. If you're feeding audio faster than real-time, for example, from a file, that usage can double (1.2x cores). Meanwhile, the memory in this context is operating memory for decoding speech. It does *not* take into account the actual full size of the language model, which resides in file cache. It's an extra 2 GB for `ja-JP`; for `en-US`, it might be more (6-7 GB).

If you have a machine where memory is scarce, and you're trying to deploy multiple languages on it, it's possible that file cache is full, and the OS is forced to page models in and out. For a running transcription that could be disastrous and might lead to slowdowns and other performance implications.

Furthermore, we prepackage executables for machines with the [advanced vector extension \(AVX2\)](#) instruction set. A machine with the AVX512 instruction set requires code generation for that target, and starting 10 containers for 10 languages might temporarily exhaust CPU. A message like this one appears in the docker logs:

Console

```
2020-01-16 16:46:54.981118943  
[W:onnxruntime:Default, tvm_utils.cc:276 LoadTVMFuncFromCache]
```

```
Can't find Scan4_llvm__mcpu_skylake_avx512 in cache, using JIT...
```

You can set the number of decoders you want inside a *single* container using `DECODER_MAX_COUNT` variable. Start with your CPU and memory SKU and then refer to the recommended host machine resource specifications.

Could you help with capacity planning and cost estimation of on-premises Speech to text containers?

For container capacity in batch processing mode, each decoder could handle 2-3x in real-time, with two CPU cores, for a single recognition. We don't recommend keeping more than two concurrent recognitions per container instance, but recommend running more instances of containers for reliability and availability reasons, behind a load balancer.

Though we could have each container instance running with more decoders. For example, we may be able to set up seven decoders per container instance on an eight-core machine at more than 2x each, yielding 15x throughput. There's a param `DECODER_MAX_COUNT` to be aware of. For the extreme case, reliability and latency issues arise, with throughput increased significantly. For a microphone, it is at 1x real-time. The overall usage should be at about one core for a single recognition.

For scenario of processing 1-K hours per day in batch processing mode, in an extreme case, 3 VMs could handle it within 24 hours but not guaranteed. To handle spike days, failover, update, and to provide minimum backup/BCP, we recommend 4-5 machines instead of 3 per cluster, and with 2+ clusters.

For hardware, we use standard Azure VM `DS13_v2` as a reference (each core must be 2.6 GHz or better, with AVX2 instruction set enabled).

[+] Expand table

Instance	vCPU(s)	RAM	Temp storage	Standard pricing with AHB	1-year reserve with AHB (% Savings)	3-year reserved with AHB (% Savings)
DS13 v2	8	56 GiB	112 GiB	\$0.598/hour	\$0.3528/hour (~41%)	\$0.2333/hour (~61%)

Based on the design reference (two clusters of 5 VMs to handle 1 K hours/day audio batch processing), one-year hardware cost will be:

2 (clusters) * 5 (VMs per cluster) * \$0.3528/hour * 365 (days) * 24 (hours) = \$31 K / year

When mapping to a physical machine, a general estimation is 1 vCPU = 1 Physical CPU Core. In reality, 1vCPU is more powerful than a single core.

For on-premises, all of these extra factors come into play:

- On what type the physical CPU is and how many cores on it
- How many CPUs running together on the same box/machine
- How VMs are set up
- How hyper-threading / multi-threading is used
- How memory is shared
- The OS, etc.

Normally it isn't as well tuned as Azure the environment. Considering other overhead, one estimation is 10 physical CPU cores = 8 Azure vCPU. Though popular CPUs only have eight cores. With on-premises deployment, the cost is higher than using Azure VMs. Also, consider the depreciation rate.

Service cost is the same as the online service: \$1/hour for speech to text. The Speech service cost is:

\$1 * 1000 * 365 = \$365 K

Maintenance cost paid to Microsoft depends on the service level and content of the service. People cost isn't included. Other infrastructure costs such as storage, networks, and load balancers aren't included.

Can I use a custom acoustic model and language model with Speech container?

One model ID can be used, either a custom language model or custom acoustic model. Passing acoustic and language models concurrently isn't supported.

Could you explain these errors from the custom speech to text container?

Error 1:

Windows Command Prompt

```
Failed to fetch manifest: Status: 400 Bad Request Body:  
{
```

```
        "code": "InvalidModelError",
        "message": "The specified model isn't supported for endpoint manifests."
    }
```

If you're training with the latest custom model, we currently don't support that. If you train with an older version, it should be possible to use. We are still working on supporting the latest versions.

Essentially, the custom containers do not support Halide or ONNX-based acoustic models (which is the default in the custom training portal). This is due to custom models not being encrypted and we don't want to expose ONNX models, however; language models are fine. The model size may be larger by 100 MB.

Error 2:

Windows Command Prompt

```
HTTPAPI result code = HTTPAPI_OK.
HTTP status code = 400.
Reason: Synthesis failed.
StatusCode: InvalidArgument,
Details: Voice doesn't match.
```

You need to provide the correct voice name in the request, which is case-sensitive. Refer to the full service name mapping.

Error 3:

JSON

```
{
    "code": "InvalidProductId",
    "message": "The subscription SKU \"CognitiveServices.S0\" isn't supported in
this service instance."
}
```

You need to create an AI Foundry resource for Speech, not an AI Foundry resource.

What API protocols are supported, REST or WS?

For speech to text and custom speech to text containers, we currently only support the websocket based protocol. The SDK only supports calling in WS but not REST. For more information, see [host URLs](#).

How can we benchmark a rough measure of transactions/second/core?

Here are some of the rough numbers to expect from the model prior to general availability (GA):

- For files, the throttling is in the Speech SDK, at 2x. The first five seconds of audio aren't throttled. The decoder is capable of doing about 3x real-time. For this, the overall CPU usage is close to two cores for a single recognition.
- For mic, it is at 1x real-time. The overall usage should be at about one core for a single recognition.

This can all be verified from the docker logs. We actually dump the line with session and phrase/utterance statistics, and that includes the RTF numbers.

How do I make multiple containers run on the same host?

By setting each container to listen on a different port using the `-p` argument. Use `-p <outside_unique_port>:5000`. For example: `-p 5001:5000`.

Technical questions

How can I get real-time APIs to handle audio > 30 seconds long?

The `RecognizeOnce()` SDK operation in interactive mode processes up to 30 seconds of audio where utterances are expected to be short. You use `StartContinuousRecognition()` for dictation or conversation longer than 30 seconds.

What are the recommended CPU and RAM resources for Speech containers?

Speech to text

The following table describes the minimum and recommended allocation of resources for Speech to text.

 Expand table

Container	Minimum	Recommended	Speech Model	Concurrency Limit
Speech to text	4 core, 4-GB memory	8 core, 8-GB memory	+4 to 8 GB memory	2 sessions per core

⚠ Note

The minimum and recommended allocations are based on Docker limits, *not* the host machine resources. For example, speech to text containers memory map portions of a large language model. We recommend that the entire file should fit in memory. You need to add an extra 4 to 8 GB to load the speech models (see above table). Also, the first run of either container might take longer because models are being paged into memory.

- For files, the throttling will be in the Speech SDK, at 2x. The first 5 seconds of audio aren't throttled.
- For speech to text, the decoder is capable of doing about 2-3x real-time. That's why we don't recommend keeping more than two active connections per core. The extreme side would be to put about 10 decoders at 2x real-time in an eight-core machine like `DS13_V2`. For the container version 1.3 and later, there's a param you could try setting `-e DECODER_MAX_COUNT=20`.
- For microphone, speech to text happens at 1x real-time. The overall usage should be at about one core for a single recognition.
- Beyond 16 cores the system becomes nonuniform memory access (NUMA) node sensitive.
- Each core must be at least 2.6 GHz or faster.

Consider the total number of hours of audio you have, and your expected time to complete the task. If the number is large, to improve reliability and availability, we suggest running more instances of containers, either on a single box or on multiple boxes, behind a load balancer. Orchestration could be done using Kubernetes (K8S) and Helm, or with Docker compose.

As an example, we have performed speech to text on 1000 hours of audio within 24 hours with 4-5 VMs and 10 instances/decoders per VM.

Does the Speech container support punctuation?

We have capitalization (ITN) available in the on-premises container. Punctuation is language-dependent, and not supported for some languages, including Chinese and Japanese.

We *do* have implicit and basic punctuation support for the existing containers, but it is `off` by default. What that means is that you can get the `.` character in your example, but not the `,` character. To enable this implicit logic, here's an example of how to do so in Python using our Speech SDK. It would be similar in other languages:

Python

```
speech_config.set_service_property(  
    name='punctuation',  
    value='implicit',  
    channel=speechsdk.ServicePropertyChannel.UriQueryParameter  
)
```

Why am I getting 404 errors when attempting to POST data to speech to text container?

Speech to text containers don't support REST API. The Speech SDK uses WebSockets. For more information, see [host URLs](#).

Why is the container running as a nonroot user? What issues might occur because of this?

The default user inside the container is a nonroot user. This provides protection against processes escaping the container and obtaining escalated permissions on the host node. By default, some platforms like the OpenShift Container Platform already do this by running containers using an arbitrarily assigned user ID. For these platforms, the nonroot user must have permissions to write to any externally mapped volume that requires writes. For example a logging folder, or a custom model download folder.

When I use the speech to text service, why am I getting this error?

Windows Command Prompt

```
Error in STT call for file 9136835610040002161_413008000252496:  
{  
    "reason": "ResultReason.Canceled",  
    "error_details": "Due to service inactivity the client buffer size exceeded."
```

```
Resetting the buffer. SessionId: xxxxx..."  
}
```

Cancellation can occur when the audio feed exceeds the Speech container buffer size. You need to control the concurrency and the Real-Time Factor (RTF) at which you send the audio. The RTF can fluctuate depending on concurrent requests, CPU, and memory allocated.

Next steps

[Azure AI containers](#)

What are Speech devices?

08/07/2025

The [Speech service](#) works with a wide variety of devices and audio sources. You can use the default audio processing available on a device. Otherwise, the Speech SDK has an option for you to use our advanced audio processing algorithms that are designed to work well with the [Speech service](#). It provides accurate far-field [speech recognition](#) via noise suppression, echo cancellation, beamforming, and dereverberation.

Audio processing

Audio processing is enhancements applied to a stream of audio to improve the audio quality. Examples of common enhancements include automatic gain control (AGC), noise suppression, and acoustic echo cancellation (AEC). The Speech SDK integrates [Microsoft Audio Stack \(MAS\)](#), allowing any application or product to use its audio processing capabilities on input audio.

Microphone array recommendations

The Speech SDK works best with a microphone array designed according to our recommended guidelines. For details, see [Microphone array recommendations](#).

Device development kits

The Speech SDK is designed to work with purpose-built development kits, and varying microphone array configurations. For example, you can use one of these Azure development kits.

- [Azure Percept DK](#) contains a preconfigured audio processor and a four-microphone linear array. You can use voice commands, keyword spotting, and far field speech with the help of Azure AI services.
- [Azure Kinect DK](#) is a spatial computing developer kit with advanced AI sensors that provide sophisticated Azure AI Vision and speech models. As an all-in-one small device with multiple modes, it contains a depth sensor, spatial microphone array with a video camera, and orientation sensor.

Next steps

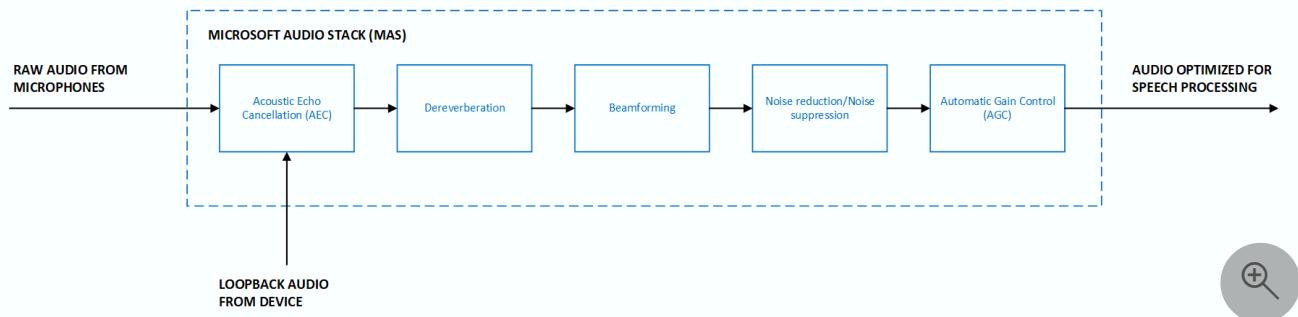
- [Audio processing concepts](#)

Audio processing with the Microsoft Audio Stack

08/06/2025

The Microsoft Audio Stack is a set of enhancements optimized for speech processing scenarios. This includes examples like keyword recognition and speech recognition. It consists of various enhancements/components that operate on the input audio signal:

- **Noise suppression** - Reduce the level of background noise.
- **Beamforming** - Localize the origin of sound and optimize the audio signal using multiple microphones.
- **Dereverberation** - Reduce the reflections of sound from surfaces in the environment.
- **Acoustic echo cancellation** - Suppress audio being played out of the device while microphone input is active.
- **Automatic gain control** - Dynamically adjust the person's voice level to account for soft speakers, long distances, or noncalibrated microphones.



Different scenarios and use-cases can require different optimizations that influence the behavior of the audio processing stack. For example, in telecommunications scenarios such as telephone calls, it's acceptable to have minor distortions in the audio signal after processing has been applied. This is because humans can continue to understand the speech with high accuracy. However, it's unacceptable and disruptive for a person to hear their own voice in an echo. This contrasts with speech processing scenarios, where distorted audio can adversely affect a machine-learned speech recognition model's accuracy, but it's acceptable to have minor levels of echo residual.

Processing is performed fully locally where the Speech SDK is being used. No audio data is streamed to Microsoft's cloud services for processing by the Microsoft Audio Stack. The only exception to this is for the Conversation Transcription Service, where raw audio is sent to Microsoft's cloud services for processing.

The Microsoft Audio Stack also powers a wide range of Microsoft products:

- **Windows** - Microsoft Audio Stack is the default speech processing pipeline when using the Speech audio category.
- **Microsoft Teams Displays and Microsoft Teams Rooms devices** - Microsoft Teams Displays and Teams Rooms devices use the Microsoft Audio Stack to enable high quality hands-free, voice-based experiences with Cortana.

Speech SDK integration

The Speech SDK integrates Microsoft Audio Stack (MAS), allowing any application or product to use its audio processing capabilities on input audio. Some of the key Microsoft Audio Stack features available via the Speech SDK include:

- **Real-time microphone input & file input** - Microsoft Audio Stack processing can be applied to real-time microphone input, streams, and file-based input.
- **Selection of enhancements** - To allow for full control of your scenario, the SDK allows you to disable individual enhancements like dereverberation, noise suppression, automatic gain control, and acoustic echo cancellation. For example, if your scenario doesn't include rendering output audio that needs to be suppressed from the input audio, you have the option to disable acoustic echo cancellation.
- **Custom microphone geometries** - The SDK allows you to provide your own custom microphone geometry information, in addition to supporting preset geometries like linear two-mic, linear four-mic, and circular 7-mic arrays (see more information on supported preset geometries at [Microphone array recommendations](#)).
- **Beamforming angles** - Specific beamforming angles can be provided to optimize audio input originating from a predetermined location, relative to the microphones.

Minimum requirements to use Microsoft Audio Stack

Microsoft Audio Stack can be used by any product or application that can meet the following requirements:

- **Raw audio** - Microsoft Audio Stack requires raw (unprocessed) audio as input to yield the best results. Providing audio that is already processed limits the audio stack's ability to perform enhancements at high quality.
- **Microphone geometries** - Geometry information about each microphone on the device is required to correctly perform all enhancements offered by the Microsoft Audio Stack. Information includes the number of microphones, their physical arrangement, and coordinates. Up to 16 input microphone channels are supported.

- **Loopback or reference audio** - An audio channel that represents the audio being played out of the device is required to perform acoustic echo cancellation.
- **Input format** - Microsoft Audio Stack supports down sampling for sample rates that are integral multiples of 16 kHz. A minimum sampling rate of 16 kHz is required. Additionally, the following formats are supported: 32-bit IEEE little endian float, 32-bit little endian signed int, 24-bit little endian signed int, 16-bit little endian signed int, and 8-bit signed int.

Related content

- [Use the Speech SDK for audio processing](#)

Use the Microsoft Audio Stack (MAS)

08/07/2025

The Speech SDK integrates Microsoft Audio Stack (MAS), allowing any application or product to use its audio processing capabilities on input audio. See the [Audio processing](#) documentation for an overview.

In this article, you learn how to use the Microsoft Audio Stack (MAS) with the Speech SDK.

Important

On Speech SDK for C++ and C# v1.33.0 and newer, the `Microsoft.CognitiveServices.Speech.Extension.MAS` package must be installed to use the Microsoft Audio Stack on Windows, and on Linux if you install the Speech SDK using NuGet.

Default options

This sample shows how to use MAS with all default enhancement options on input from the device's default microphone.

C#

```
C#  
  
var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),  
    "YourSpeechKey");  
  
var audioProcessingOptions =  
    AudioProcessingOptions.Create(AudioProcessingConstants.AUDIO_INPUT_PROCESSING_  
        ENABLE_DEFAULT);  
var audioInput =  
    AudioConfig.FromDefaultMicrophoneInput(audioProcessingOptions);  
  
var recognizer = new SpeechRecognizer(speechConfig, audioInput);
```

Preset microphone geometry

This sample shows how to use MAS with a predefined microphone geometry on a specified audio input device. In this example:

- **Enhancement options** - The default enhancements are applied on the input audio stream.
- **Preset geometry** - The preset geometry represents a linear 2-microphone array.
- **Audio input device** - The audio input device ID is `hw:0,1`. For more information on how to select an audio input device, see [How to: Select an audio input device with the Speech SDK](#).

C#

```
var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
    "YourSpeechKey");

var audioProcessingOptions =
    AudioProcessingOptions.Create(AudioProcessingConstants.AUDIO_INPUT_PROCESSING_
        ENABLE_DEFAULT, PresetMicrophoneArrayGeometry.Linear2);
var audioInput = AudioConfig.FromMicrophoneInput("hw:0,1",
    audioProcessingOptions);

var recognizer = new SpeechRecognizer(speechConfig, audioInput);
```

Custom microphone geometry

This sample shows how to use MAS with a custom microphone geometry on a specified audio input device. In this example:

- **Enhancement options** - The default enhancements are applied on the input audio stream.
- **Custom geometry** - A custom microphone geometry for a 7-microphone array is provided via the microphone coordinates. The units for coordinates are millimeters.
- **Audio input** - The audio input is from a file, where the audio within the file is expected from an audio input device corresponding to the custom geometry specified.

C#

```
var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
    "YourSpeechKey");

MicrophoneCoordinates[] microphoneCoordinates = new MicrophoneCoordinates[7]
{
    new MicrophoneCoordinates(0, 0, 0),
```

```

    new MicrophoneCoordinates(40, 0, 0),
    new MicrophoneCoordinates(20, -35, 0),
    new MicrophoneCoordinates(-20, -35, 0),
    new MicrophoneCoordinates(-40, 0, 0),
    new MicrophoneCoordinates(-20, 35, 0),
    new MicrophoneCoordinates(20, 35, 0)
};

var microphoneArrayGeometry = new
MicrophoneArrayGeometry(MicrophoneArrayType.Planar, microphoneCoordinates);
var audioProcessingOptions =
AudioProcessingOptions.Create(AudioProcessingConstants.AUDIO_INPUT_PROCESSING_
ENABLE_DEFAULT, microphoneArrayGeometry, SpeakerReferenceChannel.LastChannel);
var audioInput = AudioConfig.FromWavFileInput("katiesteve.wav",
audioProcessingOptions);

var recognizer = new SpeechRecognizer(speechConfig, audioInput);

```

Select enhancements

This sample shows how to use MAS with a custom set of enhancements on the input audio. By default, all enhancements are enabled but there are options to disable dereverberation, noise suppression, automatic gain control, and echo cancellation individually by using

`AudioProcessingOptions`.

In this example:

- **Enhancement options** - Echo cancellation and noise suppression are disabled, while all other enhancements remain enabled.
- **Audio input device** - The audio input device is the default microphone of the device.

C#

```

C#
```

```

var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
"YourSpeechKey");

var audioProcessingOptions =
AudioProcessingOptions.Create(AudioProcessingConstants.AUDIO_INPUT_PROCESSING_
DISABLE_ECHO_CANCELLATION |
AudioProcessingConstants.AUDIO_INPUT_PROCESSING_DISABLE_NOISE_SUPPRESSION |
AudioProcessingConstants.AUDIO_INPUT_PROCESSING_ENABLE_DEFAULT);
var audioInput =
AudioConfig.FromDefaultMicrophoneInput(audioProcessingOptions);

var recognizer = new SpeechRecognizer(speechConfig, audioInput);

```

Specify beamforming angles

This sample shows how to use MAS with a custom microphone geometry and beamforming angles on a specified audio input device. In this example:

- **Enhancement options** - The default enhancements are applied on the input audio stream.
- **Custom geometry** - A custom microphone geometry for a 4-microphone array is provided by specifying the microphone coordinates. The units for coordinates are millimeters.
- **Beamforming angles** - Beamforming angles are specified to optimize for audio originating in that range. The units for angles are degrees.
- **Audio input** - The audio input is from a push stream, where the audio within the stream is expected from an audio input device corresponding to the custom geometry specified.

In the following code example, the start angle is set to 70 degrees and the end angle is set to 110 degrees.

C#

C#

```
var speechConfig = SpeechConfig.FromEndpoint(new Uri("YourSpeechEndpoint"),
"YourSpeechKey");

MicrophoneCoordinates[] microphoneCoordinates = new MicrophoneCoordinates[4]
{
    new MicrophoneCoordinates(-60, 0, 0),
    new MicrophoneCoordinates(-20, 0, 0),
    new MicrophoneCoordinates(20, 0, 0),
    new MicrophoneCoordinates(60, 0, 0)
};

var microphoneArrayGeometry = new
MicrophoneArrayGeometry(MicrophoneArrayType.Linear, 70, 110,
microphoneCoordinates);
var audioProcessingOptions =
AudioProcessingOptions.Create(AudioProcessingConstants.AUDIO_INPUT_PROCESSING_
ENABLE_DEFAULT, microphoneArrayGeometry, SpeakerReferenceChannel.LastChannel);
var pushStream = AudioInputStream.CreatePushStream();
var audioInput = AudioConfig.FromStreamInput(pushStream,
audioProcessingOptions);

var recognizer = new SpeechRecognizer(speechConfig, audioInput);
```

Reference channel for echo cancellation

Microsoft Audio Stack requires the reference channel (also known as loopback channel) to perform echo cancellation. The source of the reference channel varies by platform:

- **Windows** - The reference channel is automatically gathered by the Speech SDK if the `SpeakerReferenceChannel::LastChannel` option is provided when creating `AudioProcessingOptions`.
- **Linux** - ALSA (Advanced Linux Sound Architecture) must be configured to provide the reference audio stream as the last channel for the audio input device used. ALSA is configured in addition to providing the `SpeakerReferenceChannel::LastChannel` option when creating `AudioProcessingOptions`.

Language and platform support

[] [Expand table](#)

Language	Platform	Reference docs
C++	Windows, Linux	C++ docs
C#	Windows, Linux	C# docs
Java	Windows, Linux	Java docs

Related content

- [Set up development environment](#)

Microphone array recommendations

08/07/2025

In this article, you learn how to design a microphone array customized for use with the Speech SDK. This is most pertinent if you're selecting, specifying, or building hardware for speech solutions.

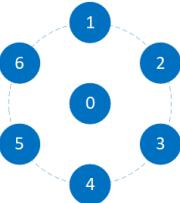
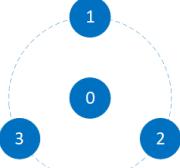
The Speech SDK works best with a microphone array designed according to these guidelines, including the microphone geometry, component selection, and architecture.

Microphone geometry

The following array geometries are recommended for use with the Microsoft Audio Stack.

Location of sound sources and rejection of ambient noise is improved with greater number of microphones with dependencies on specific applications, user scenarios, and the device form factor.

 Expand table

Array	Microphones	Geometry
Circular - 7 Microphones		6 Outer, 1 Center, Radius = 42.5 mm, Evenly Spaced
Circular - 4 Microphones		3 Outer, 1 Center, Radius = 42.5 mm, Evenly Spaced
Linear - 4 Microphones		Length = 120 mm, Spacing = 40 mm

Array	Microphones	Geometry
Linear - 2 Microphones		Spacing = 40 mm



Microphone channels should be ordered ascending from 0, according to the numbering previously described for each array. The Microsoft Audio Stack requires another reference stream of audio playback to perform echo cancellation.

Component selection

Microphone components should be selected to accurately reproduce a signal free of noise and distortion.

The recommended properties when selecting microphones are:

[\[+\] Expand table](#)

Parameter	Recommended
SNR	>= 65 dB (1 kHz signal 94 dB SPL, A-weighted noise)
Amplitude Matching	± 1 dB @ 1 kHz
Phase Matching	± 2° @ 1 kHz
Acoustic Overload Point (AOP)	>= 120 dB SPL (THD = 10%)
Bit Rate	Minimum 24-bit
Sampling Rate	Minimum 16 kHz*
Frequency Response	± 3 dB, 200-8000 Hz Floating Mask*
Reliability	Storage Temperature Range -40°C to 70°C Operating Temperature Range -20°C to 55°C

*Higher sampling rates or "wider" frequency ranges might be necessary for high-quality communications (VoIP) applications

Good component selection must be paired with good electroacoustic integration in order to avoid impairing the performance of the components used. Unique use cases might also

necessitate more requirements (such as operating temperature ranges).

Microphone array integration

The performance of the microphone array when integrated into a device differs from the component specification. It's important to ensure that the microphones are well matched after integration. Therefore the device performance measured after any fixed gain or EQ should meet the following recommendations:

[+] Expand table

Parameter	Recommended
SNR	>= 64 dB (1 kHz signal 94 dBSPL, A-weighted noise)
Output Sensitivity	-26 dBFS/Pa @ 1 kHz (recommended)
Amplitude Matching	± 2 dB, 200-8000 Hz
THD%*	≤ 1%, 200-8000 Hz, 94 dBSPL
Frequency Response	± 6 dB, 200-12000 Hz Floating Mask**

**A low distortion speaker is required to measure THD (for example, Neumann KH120)

**"Wider" frequency ranges might be necessary for high-quality communications (VoIP) applications

Speaker integration recommendations

As echo cancellation is necessary for speech recognition devices that contain speakers, more recommendations are provided for speaker selection and integration.

[+] Expand table

Parameter	Recommended
Linearity Considerations	No nonlinear processing after speaker reference, otherwise a hardware-based loopback reference stream is required
Speaker Loopback	Provided via WASAPI, private APIs, custom ALSA plug-in (Linux), or provided via firmware channel
THD%	Third Octave Bands minimum fifth Order, 70 dBA Playback @ 0.8 m ≤ 6.3%, 315-500 Hz ≤ 5%, 630-5000 Hz

Parameter	Recommended
Echo Coupling to Microphones	> -10 dB TCLw using ITU-T G.122 Annex B.4 method, normalized to mic level TCLw = TCLwmeasured + (Measured Level - Target Output Sensitivity) TCLw = TCLwmeasured + (Measured Level - (-26))

Integration design architecture

The following guidelines for architecture are necessary when integrating microphones into a device:

[] [Expand table](#)

Parameter	Recommendation
Mic Port Similarity	All microphone ports are same length in array
Mic Port Dimensions	Port size Ø0.8-1.0 mm. Port Length / Port Diameter < 2
Mic Sealing	Sealing gaskets uniformly implemented in stack-up. Recommend > 70% compression ratio for foam gaskets
Mic Reliability	Mesh should be used to prevent dust and ingress (between PCB for bottom ported microphones and sealing gasket/top cover)
Mic Isolation	Rubber gaskets and vibration decoupling through structure, particularly for isolating any vibration paths due to integrated speakers
Sampling Clock	Device audio must be free of jitter and drop-outs with low drift
Record Capability	The device must be able to record individual channel raw streams simultaneously
USB	All USB audio input devices must set descriptors according to the USB Audio Devices Rev3 Spec ↗
Microphone Geometry	Drivers must implement Microphone Array Geometry Descriptors correctly
Discoverability	Devices must not have any undiscoverable or uncontrollable hardware, firmware, or third party software-based nonlinear audio processing algorithms to/from the device
Capture Format	Capture formats must use a minimum sampling rate of 16 kHz and recommended 24-bit depth

Electrical architecture considerations

Where applicable, arrays can be connected to a USB host (such as a SoC that runs the [Microsoft Audio Stack \(MAS\)](#)) and interfaces to Speech services or other applications.

Hardware components such as PDM-to-TDM conversion should ensure that the dynamic range and SNR of the microphones is preserved within re-samplers.

High-speed USB Audio Class 2.0 should be supported within any audio MCUs in order to provide the necessary bandwidth for up to seven channels at higher sample rates and bit depths.

Next steps

[Learn more about audio processing](#)

What is embedded speech?

08/07/2025

Embedded Speech is designed for on-device [speech to text](#) and [text to speech](#) scenarios where cloud connectivity is intermittent or unavailable. For example, you can use embedded speech in industrial equipment, a voice enabled air conditioning unit, or a car that might travel out of range. You can also develop hybrid cloud and offline solutions. For scenarios where your devices must be in a secure environment like a bank or government entity, you should first consider [disconnected containers](#).

Important

Microsoft limits access to embedded speech. You can apply for access through the Azure AI Speech [embedded speech limited access review](#). For more information, see [Limited access for embedded speech](#).

Platform requirements

Embedded speech is included with the Speech SDK (version 1.24.1 and higher) for C#, C++, and Java. Refer to the general [Speech SDK installation requirements](#) for programming language and target platform specific details.

Choose your target environment

Android

Requires Android 8.0 (API level 26) or higher on Arm64 (`arm64-v8a`) or Arm32 (`armeabi-v7a`) hardware.

Limitations

Embedded speech is only available with C#, C++, and Java SDKs. The other Speech SDKs, Speech CLI, and REST APIs don't support embedded speech.

Embedded speech recognition only supports mono 16 bit, 8-kHz or 16-kHz PCM-encoded WAV audio formats.

Embedded neural voices support 24 kHz RIFF/RAW, with a RAM requirement of 100 MB.

Embedded speech SDK packages

For C# embedded applications, install following Speech SDK for C# packages:

[] Expand table

Package	Description
Microsoft.CognitiveServices.Speech	Required to use the Speech SDK
Microsoft.CognitiveServices.Speech.Extension.Embedded.SR	Required for embedded speech recognition
Microsoft.CognitiveServices.Speech.Extension.Embedded.TTS	Required for embedded speech synthesis
Microsoft.CognitiveServices.Speech.Extension.ONNX.Runtime	Required for embedded speech recognition and synthesis
Microsoft.CognitiveServices.Speech.Extension.Telemetry	Required for embedded speech recognition and synthesis

Models and voices

For embedded speech, you need to download the speech recognition models for [speech to text](#) and voices for [text to speech](#). Instructions are provided upon successful completion of the [limited access review](#) process.

The following [speech to text](#) models are available: da-DK, de-DE, en-AU, en-CA, en-GB, en-IE, en-IN, en-NZ, en-US, es-ES, es-MX, fr-CA, fr-FR, it-IT, ja-JP, ko-KR, pt-BR, pt-PT, zh-CN, zh-HK, and zh-TW.

All text to speech locales [here](#) (except fa-IR, Persian (Iran)) are available out of box with either 1 selected female and/or 1 selected male voices. We welcome your input to help us gauge demand for more languages and voices.

Embedded speech configuration

For cloud connected applications, as shown in most Speech SDK samples, you use the `SpeechConfig` object with an API key and endpoint. For embedded speech, you don't use an AI Foundry resource for Speech. Instead of a cloud resource, you use the [models and voices](#) that you download to your local device.

Use the `EmbeddedSpeechConfig` object to set the location of the models or voices. If your application is used for both speech to text and text to speech, you can use the same `EmbeddedSpeechConfig` object to set the location of the models and voices.

C#

```
// Provide the location of the models and voices.  
List<string> paths = new List<string>();  
paths.Add("C:\\dev\\embedded-speech\\stt-models");  
paths.Add("C:\\dev\\embedded-speech\\tts-voices");  
var embeddedSpeechConfig = EmbeddedSpeechConfig.FromPaths(paths.ToArray());  
  
// For speech to text  
embeddedSpeechConfig.SetSpeechRecognitionModel(  
    "Microsoft Speech Recognizer en-US FP Model V8",  
    Environment.GetEnvironmentVariable("EMBEDDED_SPEECH_MODEL_LICENSE"));  
  
// For text to speech  
embeddedSpeechConfig.SetSpeechSynthesisVoice(  
    "Microsoft Server Speech Text to Speech Voice (en-US, JennyNeural)",  
    Environment.GetEnvironmentVariable("EMBEDDED_SPEECH_MODEL_LICENSE"));  
embeddedSpeechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Ri  
ff24Khz16BitMonoPcm);
```

Embedded speech code samples

You can find ready to use embedded speech samples at [GitHub](#). For remarks on projects from scratch, see samples specific documentation:

- [C# \(.NET 8.0\)](#)
- [C# \(.NET MAUI\)](#)
- [C# for Unity](#)

Hybrid speech

Hybrid speech with the `HybridSpeechConfig` object uses the cloud speech service by default and embedded speech as a fallback in case cloud connectivity is limited or slow.

With hybrid speech configuration for [speech to text](#) (recognition models), embedded speech is used when connection to the cloud service fails after repeated attempts. Recognition might continue using the cloud service again if the connection is later resumed.

With hybrid speech configuration for [text to speech](#) (voices), embedded and cloud synthesis are run in parallel and the final result is selected based on response speed. The best result is evaluated again on each new synthesis request.

Cloud speech

For cloud speech, you use the `SpeechConfig` object, as shown in the [speech to text quickstart](#) and [text to speech quickstart](#). To run the quickstarts for embedded speech, you can replace `SpeechConfig` with `EmbeddedSpeechConfig` or `HybridSpeechConfig`. Most of the other speech recognition and synthesis code are the same, whether using cloud, embedded, or hybrid configuration.

Embedded voices capabilities

For embedded voices, it's essential to note that certain [Speech synthesis markup language \(SSML\)](#) tags might not be currently supported due to differences in the model structure. For detailed information regarding the unsupported SSML tags, refer to the following table.

[Expand table](#)

Level 1	Level 2	Sub values	Support in embedded NTTS
audio	src		No
bookmark			Yes
break	strength		Yes
	time		Yes
silence	type	Leading, Tailing, Comma-exact, etc.	No
	value		No
emphasis	level		No
lang			No
lexicon	uri		Yes
math			No
msttsaudioduration	value		No
msttsbackgroundaudio	src		No
	volume		No
	fadein		No
	fadeout		No

Level 1	Level 2	Sub values	Support in embedded NTTS
msttsexpress-as	style		Yes ¹
	styledegree		No
	role		No
msttssilence			No
msttviseme	type	redlips_front, FacialExpression	No
p			Yes
phoneme	alphabet	ipa, sapi, ups, etc.	Yes
	ph		Yes
prosody	contour	Sentences level support, word level only en-US and zh-CN	Yes
	pitch		Yes
	range		Yes
	rate		Yes
s	volume		Yes
			Yes
say-as	interpret-as	characters, spell-out, number_digit, date, etc.	Yes
	format		Yes
	detail		Yes
sub	alias		Yes
speak			Yes
voice			No

¹ The [msttsexpress-as](#) style is supported only for the `en-US-JennyNeural` voice.

Related content

- [Read about text to speech on devices for disconnected and hybrid scenarios ↗](#)
- [Limited Access to embedded Speech](#)

Evaluating performance of Embedded Speech

08/07/2025

Embedded speech models run fully on your target devices. Understanding the performance characteristics of these models on your devices' hardware can be critical to delivering low latency experiences within your products and applications. This guide provides information to help answer the question, "Is my device suitable to run embedded speech to text and speech translation models?"

Metrics & terminology

Real-time factor (RTF) – The real-time factor (RTF) of a device measures how fast the embedded speech model can process audio input. It's the ratio of the processing time to the audio length. For example, if a device processes a 1-minute audio file in 30 seconds, the RTF is 0.5. This metric evaluates the computational power of the device for running embedded speech models. It can help identify devices that are too slow to support the models. Measurement of this metric should only be done using file-based input rather than real-time microphone input.

To support real-time & interactive speech experiences, the device should have an RTF of 1 or lower. An RTF value higher than 1 means that the device can't keep up with the audio input, which can result in a poor user experience.

When measuring the RTF of a device, it's important to measure multiple samples and analyze the distribution across percentiles. This allows you to capture the effect of variations in the device's behavior like different CPU clock speeds due to thermal throttling. The predefined measurement tests outlined in [Measuring the real-time factor on your device](#) automatically measure the RTF for each speech recognition result, yielding a sufficiently large sample size.

User-perceived latency (UPL) – The user-perceived latency (UPL) of speech to text is the time between a word being spoken and the word being shown in the recognition results.

Factors that affect performance

Device specifications – The specifications of your device play a key role in whether embedded speech models can run without performance issues. CPU clock speed, architecture (for example, x64, ARM processor, etcetera), and memory can all affect model inference speed.

CPU load – In most cases, your device is running other applications in parallel to the application where embedded speech models are integrated. The amount of CPU load your device experiences when idle and at peak can also affect performance.

For example, if the device is under moderate to high CPU load from all other applications running on the device, it's possible to encounter performance issues for running embedded speech in addition to the other applications, even with a powerful processor.

Memory load – An embedded speech to text model consumes between 200-300 MB of memory at runtime. If your device has less memory available for the embedded speech process to use, frequent fallbacks to virtual memory and paging can introduce more latencies. This can affect both the real-time factor and user-perceived latency.

Built-in performance optimizations

All embedded speech to text models come with a Voice Activity Detector (VAD) component that aims to filter out silence and non-speech content from the audio input. The goal is to reduce the CPU load and the processing time for other speech to text model components.

The VAD component is always on and doesn't need any configuration from you as the developer. It works best when the audio input has non-negligible amounts of silence or non-speech content, which is common in scenarios like captioning, commanding, and dictation.

Measuring the real-time factor on your device

For all embedded speech supported platforms, a code sample is available on GitHub that includes a performance measurement mode. In this mode, the goal is to measure the real-time factor (RTF) of your device by controlling as many variables as possible:

- **Model** – The English (United States) model is used for measurement. Models for all other supported locales follow similar performance characteristics, so measuring through the English (United States) model is sufficient.
- **Audio input** – A prebuilt audio file designed for RTF measurements is made available as a supplemental download to the sample code.
- **Measurement mechanism** – The start and stop markers of time measurement are preconfigured in the sample to ensure accuracy and ease of comparing results across different devices & test iterations.

This measurement should be done with the sample running directly on your target device(s), with no code changes other than specifying your model paths & encryption key. The device

should be in a state that represents a real end-user state when embedded speech would be used (for example, other active applications, CPU & memory load, etc.).

Running the sample yields performance metrics outputted to the console. The full suite of metrics includes the real-time factor along with other properties like CPU usage and memory consumption. Each metric is defined and explained below.

Instruction set metrics

 Expand table

Metric	Description	Notes
AVX512Supported	True if CPU supports AVX512 instruction set.	This flag is for X64 platforms. ONNX runtime has optimizations for the various instruction sets, and having this information can help diagnose inconsistencies.
AVXSupported	True if CPU supports AVX instruction set.	This flag is for X64 platforms. ONNX runtime has optimizations for the various instruction sets, and having this information can help diagnose inconsistencies.
AVX2Supported	True if CPU supports AVX2 instruction set.	This flag is for X64 platforms. ONNX runtime has optimizations for the various instruction sets, and having this information can help diagnose inconsistencies.
SSE3Available	True if CPU supports SSE3 instruction set.	This flag is for X64 platforms. ONNX runtime has optimizations for the various instruction sets, and having this information can help diagnose inconsistencies.
NEONAvailable	True if CPU supports NEON instruction set.	This flag is for ARM processor platforms. ONNX runtime has optimizations for the various instruction sets, and having this information can help diagnose inconsistencies.
NPU	Name of Neural Processing Unit, or N/A if none found.	This flag is for hardware acceleration.

Memory metrics

 Expand table

Metric	Description	Notes
PagefileUsage	Amount of page file used by process. Implemented for Linux and Windows.	Values are relative to the machine configuration.

Metric	Description	Notes
WorkingSetSize	Amount of memory used for the process.	
ProcessCPUUsage	Aggregate of CPU usage for the process.	Includes all threads in the process, including Speech SDK and UI threads. Aggregated across all cores.
ThreadCPUUsage	Aggregate of CPU usage for the speech recognition or speech translation thread.	

Performance metrics

[Expand table](#)

Metric	Description	Notes
RealTimeFactor	Measures how much faster than real-time the embedded speech engine is processing audio. Includes audio loading time.	Values greater than 1 indicate that the engine is processing audio slower than real-time. Values less than 1 indicate the engine is processing audio faster than real-time. This value should only be analyzed in file-based input mode. It shouldn't be analyzed in streaming input mode.
StreamingRealTimeFactor	Measures how much faster than real-time the engine is processing audio. Excludes audio loading time.	Values greater than 1 indicate that the engine is processing audio slower than real-time. Values less than 1 indicate the engine is processing audio faster than real-time.

Related content

- [Read about text to speech on devices for disconnected and hybrid scenarios ↗](#)
- [Limited Access to embedded Speech](#)

Use Speech service through a private endpoint

09/12/2025

Azure Private Link lets you connect to services in Azure by using a [private endpoint](#). A private endpoint is a private IP address that's accessible only within a specific [virtual network](#) and subnet.

This article explains how to set up and use Private Link and private endpoints with the Speech service. This article then describes how to remove private endpoints later, but still use the Speech resource.

(!) Note

Before you proceed, review [how to use virtual networks with Azure AI services](#).

Setting up an AI Foundry resource for Speech for the private endpoint scenarios requires performing the following tasks:

1. [Create a custom domain name](#)
2. [Turn on private endpoints](#)
3. [Adjust existing applications and solutions](#)

Private endpoints and Virtual Network service endpoints

Azure provides private endpoints and Virtual Network service endpoints for traffic that tunnels via the [private Azure backbone network](#). The purpose and underlying technologies of these endpoint types are similar. But there are differences between the two technologies. We recommend that you learn about the pros and cons of both before you design your network.

There are a few things to consider when you decide which technology to use:

- Both technologies ensure that traffic between the virtual network and the Speech resource doesn't travel over the public internet.
- A private endpoint provides a dedicated private IP address for your Speech resource. This IP address is accessible only within a specific virtual network and subnet. You have full control of the access to this IP address within your network infrastructure.
- Virtual Network service endpoints don't provide a dedicated private IP address for the Speech resource. Instead, they encapsulate all packets sent to the Speech resource and

deliver them directly over the Azure backbone network.

- Both technologies support on-premises scenarios. By default, when they use Virtual Network service endpoints, Azure service resources secured to virtual networks can't be reached from on-premises networks. But you can [change that behavior](#).
- Virtual Network service endpoints are often used to restrict the access for an AI Foundry resource for Speech based on the virtual networks from which the traffic originates.
- For Azure AI services, enabling the Virtual Network service endpoint forces the traffic for all Azure AI Foundry resources to go through the private backbone network. That requires explicit network access configuration. (For more information, see [Configure virtual networks and the Speech resource networking settings](#).) Private endpoints don't have this limitation and provide more flexibility for your network configuration. You can access one resource through the private backbone and another through the public internet by using the same subnet of the same virtual network.
- Private endpoints incur [extra costs](#). Virtual Network service endpoints are free.
- Private endpoints require [extra DNS configuration](#).
- One Speech resource can work simultaneously with both private endpoints and Virtual Network service endpoints.

We recommend that you try both endpoint types before you make a decision about your production design.

For more information, see these resources:

- [Azure Private Link and private endpoint documentation](#)
- [Virtual Network service endpoints documentation](#)

This article describes the usage of the private endpoints with Speech service. Usage of the VNet service endpoints is described [here](#).

Create a custom domain name

Caution

An AI Foundry resource for Speech with a custom domain name enabled uses a different way to interact with Speech service. You might have to adjust your application code for both of these scenarios: [with private endpoint](#) and [without private endpoint](#).

Follow these steps to create a [custom subdomain name for Azure AI services](#) for your Speech resource.

Caution

When you turn on a custom domain name, the operation is **not reversible**. The only way to go back to the **regional name** is to create a new Speech resource.

If your Speech resource has a lot of associated custom models and projects created via [Speech Studio](#), we strongly recommend trying the configuration with a test resource before you modify the resource used in production.

Azure portal

To create a custom domain name using the Azure portal, follow these steps:

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the required Speech resource.
3. In the **Resource Management** group on the left pane, select **Networking**.
4. On the **Firewalls and virtual networks** tab, select **Generate Custom Domain Name**. A new right panel appears with instructions to create a unique custom subdomain for your resource.
5. In the **Generate Custom Domain Name** panel, enter a custom domain name. Your full custom domain will look like: `https://{{your custom name}}.cognitiveservices.azure.com`.

Remember that after you create a custom domain name, it *cannot* be changed.

After you've entered your custom domain name, select **Save**.

6. After the operation finishes, in the **Resource management** group, select **Keys and Endpoint**. Confirm that the new endpoint name of your resource starts this way:
`https://{{your custom name}}.cognitiveservices.azure.com`.

Turn on private endpoints

We recommend using the [private DNS zone](#) attached to the virtual network with the necessary updates for the private endpoints. You can create a private DNS zone during the provisioning process. If you're using your own DNS server, you might also need to change your DNS configuration.

Decide on a DNS strategy before you provision private endpoints for a production Speech resource. And test your DNS changes, especially if you use your own DNS server.

Use one of the following articles to create private endpoints. These articles use a web app as a sample resource to make available through private endpoints.

- [Create a private endpoint by using the Azure portal](#)
- [Create a private endpoint by using Azure PowerShell](#)
- [Create a private endpoint by using Azure CLI](#)

Use these parameters instead of the parameters in the article that you chose:

[] [Expand table](#)

Setting	Value
Resource type	Microsoft.CognitiveServices/accounts
Resource	<your-speech-resource-name>
Target sub-resource	account

DNS for private endpoints: Review the general principles of [DNS for private endpoints in Azure AI Foundry resources](#). Then confirm that your DNS configuration is working correctly by performing the checks described in the following sections.

Resolve DNS from the virtual network

This check is *required*.

Follow these steps to test the custom DNS entry from your virtual network:

1. Sign in to a virtual machine located in the virtual network to which you attached your private endpoint.
2. Open a Windows command prompt or a Bash shell, run `nslookup`, and confirm that it successfully resolves your resource's custom domain name.

```
dos
C:\>nslookup my-private-link-speech.cognitiveservices.azure.com
Server: UnKnown
Address: 168.63.129.16

Non-authoritative answer:
Name: my-private-link-speech.privatelink.cognitiveservices.azure.com
Address: 172.28.0.10
Aliases: my-private-link-speech.cognitiveservices.azure.com
```

3. Confirm that the IP address matches the IP address of your private endpoint.

Resolve DNS from other networks

Perform this check only if you've turned on either the **All networks** option or the **Selected Networks and Private Endpoints** access option in the **Networking** section of your resource.

If you plan to access the resource by using only a private endpoint, you can skip this section.

1. Sign in to a computer attached to a network allowed to access the resource.
2. Open a Windows command prompt or Bash shell, run `nslookup`, and confirm that it successfully resolves your resource's custom domain name.

```
dos
```

```
C:\>nslookup my-private-link-speech.cognitiveservices.azure.com
Server: UnKnown
Address: fe80::1

Non-authoritative answer:
Name: vnetproxyv1-weu-prod.westeurope.cloudapp.azure.com
Address: 13.69.67.71
Aliases: my-private-link-speech.cognitiveservices.azure.com
my-private-link-speech.privateliink.cognitiveservices.azure.com
westeurope.prod.vnet.cog.trafficmanager.net
```

➊ Note

The resolved IP address points to a virtual network proxy endpoint, which dispatches the network traffic to the private endpoint for the Speech resource. The behavior will be different for a resource with a custom domain name but *without* private endpoints. See [this section](#) for details.

Adjust an application to use an AI Foundry resource for Speech with a private endpoint

An AI Foundry resource for Speech with a custom domain interacts with the Speech service in a different way. This is true for a custom-domain-enabled Speech resource both with and without private endpoints. Information in this section applies to both scenarios.

Follow instructions in this section to adjust existing applications and solutions to use an AI Foundry resource for Speech with a custom domain name and a private endpoint turned on.

An AI Foundry resource for Speech with a custom domain name and a private endpoint turned on uses a different way to interact with the Speech service. This section explains how to use such a resource with the Speech service REST APIs and the [Speech SDK](#).

 **Note**

An AI Foundry resource for Speech without private endpoints that uses a custom domain name also has a special way of interacting with the Speech service. This way differs from the scenario of an AI Foundry resource for Speech that uses a private endpoint. This is important to consider because you may decide to remove private endpoints later. See [Adjust an application to use an AI Foundry resource for Speech without private endpoints](#) later in this article.

Speech resource with a custom domain name and a private endpoint: Usage with the REST APIs

We use `my-private-link-speech.cognitiveservices.azure.com` as a sample Speech resource DNS name (custom domain) for this section.

Speech service has REST APIs for [Speech to text](#) and [Text to speech](#). Consider the following information for the private-endpoint-enabled scenario.

Speech to text has two REST APIs. Each API serves a different purpose, uses different endpoints, and requires a different approach when you're using it in the private-endpoint-enabled scenario.

The Speech to text REST APIs are:

- [Speech to text REST API](#), which is used for [Batch transcription](#) and [custom speech](#).
- [Speech to text REST API for short audio](#), which is used for real-time speech to text.

Usage of the Speech to text REST API for short audio and the Text to speech REST API in the private endpoint scenario is the same. It's equivalent to the [Speech SDK case](#) described later in this article.

Speech to text REST API uses a different set of endpoints, so it requires a different approach for the private-endpoint-enabled scenario.

The next subsections describe both cases.

Speech to text REST API

Usually, Speech resources use [Azure AI services regional endpoints](#) for communicating with the [Speech to text REST API](#). These resources have the following naming format:

```
{region}.api.cognitive.microsoft.com.
```

This is a sample request URL:

HTTP

```
https://westeurope.api.cognitive.microsoft.com/speechtotext/v3.1/transcriptions
```

 **Note**

See [this article](#) for Azure Government and Microsoft Azure operated by 21Vianet endpoints.

After you turn on a custom domain for an AI Foundry resource for Speech (which is necessary for private endpoints), that resource will use the following DNS name pattern for the basic REST API endpoint:

```
{your custom name}.cognitiveservices.azure.com
```

That means that in our example, the REST API endpoint name is:

```
my-private-link-speech.cognitiveservices.azure.com
```

And the sample request URL needs to be converted to:

HTTP

```
https://my-private-link-
speech.cognitiveservices.azure.com/speechtotext/v3.1/transcriptions
```

This URL should be reachable from the virtual network with the private endpoint attached (provided the [correct DNS resolution](#)).

After you turn on a custom domain name for an AI Foundry resource for Speech, you typically replace the host name in all request URLs with the new custom domain host name. All other parts of the request (like the path `/speechtotext/v3.1/transcriptions` in the earlier example) remain the same.

 **Tip**

Some customers develop applications that use the region part of the regional endpoint's DNS name (for example, to send the request to the Speech resource deployed in the particular Azure region).

A custom domain for an AI Foundry resource for Speech contains *no* information about the region where the resource is deployed. So the application logic described earlier will *not* work and needs to be altered.

Speech to text REST API for short audio and Text to speech REST API

The [Speech to text REST API for short audio](#) and the [Text to speech REST API](#) use two types of endpoints:

- [Azure AI services regional endpoints](#) for communicating with the Azure AI services REST API to obtain an authorization token
- Special endpoints for all other operations

! Note

See [this article](#) for Azure Government and Azure operated by 21Vianet endpoints.

The detailed description of the special endpoints and how their URL should be transformed for a private-endpoint-enabled Speech resource is provided in [this subsection](#) about usage with the Speech SDK. The same principle described for the SDK applies for the Speech to text REST API for short audio and the Text to speech REST API.

Get familiar with the material in the subsection mentioned in the previous paragraph and see the following example. The example describes the Text to speech REST API. Usage of the Speech to text REST API for short audio is fully equivalent.

! Note

When you're using the Speech to text REST API for short audio and Text to speech REST API in private endpoint scenarios, use a resource key passed through the `Ocp-Apim-Subscription-Key` header. (See details for [Speech to text REST API for short audio](#) and [Text to speech REST API](#))

Using an authorization token and passing it to the special endpoint via the `Authorization` header will work *only* if you've turned on the **All networks** access option in the

Networking section of your Speech resource. In other cases you will get either `Forbidden` or `BadRequest` error when trying to obtain an authorization token.

Text to speech REST API usage example

We use West Europe as a sample Azure region and `my-private-link-speech.cognitiveservices.azure.com` as a sample Speech resource DNS name (custom domain). The custom domain name `my-private-link-speech.cognitiveservices.azure.com` in our example belongs to the Speech resource created in the West Europe region.

To get the list of the voices supported in the region, perform the following request:

HTTP

```
https://westeurope.tts.speech.microsoft.com/cognitiveservices/voices/list
```

See more details in the [Text to speech REST API documentation](#).

For the private-endpoint-enabled Speech resource, the endpoint URL for the same operation needs to be modified. The same request looks like this:

HTTP

```
https://my-private-link-speech.cognitiveservices.azure.com/tts/cognitiveservices/voices/list
```

See a detailed explanation in the [Construct endpoint URL](#) subsection for the Speech SDK.

Speech resource with a custom domain name and a private endpoint: Usage with the Speech SDK

Using the Speech SDK with a custom domain name and private-endpoint-enabled Speech resources requires you to review and likely change your application code.

We use `my-private-link-speech.cognitiveservices.azure.com` as a sample Speech resource DNS name (custom domain) for this section.

Construct endpoint URL

Usually in SDK scenarios (and in the speech to text REST API for short audio and text to speech REST API scenarios), Speech resources use the dedicated regional endpoints for different service offerings. The DNS name format for these endpoints is:

```
{region}.{speech service offering}.speech.microsoft.com
```

An example DNS name is:

```
westeurope.stt.speech.microsoft.com
```

All possible values for the region (first element of the DNS name) are listed in [Speech service supported regions](#). (See [this article](#) for Azure Government and Azure operated by 21Vianet endpoints.) The following table presents the possible values for the Speech service offering (second element of the DNS name):

 [Expand table](#)

DNS name value	Speech service offering
s2s	Speech Translation
stt	Speech to text
tts	Text to speech
voice	Custom voice

So the earlier example (`westeurope.stt.speech.microsoft.com`) stands for a Speech to text endpoint in West Europe.

Private-endpoint-enabled endpoints communicate with Speech service via a special proxy. Because of that, *you must change the endpoint connection URLs*.

A "standard" endpoint URL looks like:

```
{region}.{speech service offering}.speech.microsoft.com/{URL path}
```

A private endpoint URL looks like:

```
{your custom name}.cognitiveservices.azure.com/{URL path}
```

The Speech SDK automatically will configure the `/{URL path}` depending on the service used. Therefore only the `/{baseUrl}` must be configured as described.

Modifying applications

Follow these steps to modify your code:

1. Determine the application endpoint URL from the 'Keys and Endpoints' menu of your resource on Azure portal. In this example it would be `my-private-link-`

speech.cognitiveservices.azure.com.

2. Create a `SpeechConfig` instance by using an endpoint URL:

- a. Modify the endpoint that you determined, as described in the earlier [Construct endpoint URL](#) section.
- b. Modify how you create the instance of `SpeechConfig`. Most likely, your application is using something like this:

C#

```
var config = SpeechConfig.FromSubscription(speechKey, azureRegion);
```

This example doesn't work for a private-endpoint-enabled Speech resource because of the host name and URL changes that we described in the previous sections. If you try to run your existing application without any modifications by using the key of a private-endpoint-enabled resource, you get an authentication error (401).

To make it work, modify how you instantiate the `SpeechConfig` class and use "from endpoint"/"with endpoint" initialization. Suppose we have the following two variables defined:

- `speechKey` contains the key of the private-endpoint-enabled Speech resource.
- `endPoint` contains the *modified* endpoint URL (using the type required by the corresponding programming language). In our example, this variable should contain:

```
wss://my-private-link-speech.cognitiveservices.azure.com
```

Create a `SpeechConfig` instance:

C#

```
var config = SpeechConfig.FromEndpoint(endPoint, speechKey);
```

C++

```
auto config = SpeechConfig::FromEndpoint(endPoint, speechKey);
```

Java

```
SpeechConfig config = SpeechConfig.fromEndpoint(endPoint, speechKey);
```

Python

```
import azure.cognitiveservices.speech as speechsdk
config = speechsdk.SpeechConfig(endpoint=endPoint, subscription=speechKey)
```

objectivec

```
SPXSpeechConfiguration *config = [[SPXSpeechConfiguration alloc]
initWithEndpoint:endPoint subscription:speechKey];
```

JavaScript

```
import * as sdk from "microsoft.cognitiveservices.speech.sdk";
config: sdk.SpeechConfig = sdk.SpeechConfig.fromEndpoint(new URL(endPoint),
speechKey);
```

After this modification, your application should work with the private-endpoint-enabled Speech resources. We're working on more seamless support of private endpoint scenarios.

Speech resource with a custom domain name and without private endpoints: Usage with the Speech SDK

Using the Speech SDK with custom-domain-enabled Speech resources *without* private endpoints is equivalent to the configuration described *with* private endpoints in this document.

Use of Speech Studio

[Speech Studio](#) is a web portal with tools for building and integrating Azure AI Speech service in your application. When you work in Speech Studio projects, network connections and API calls to the corresponding Speech resource are made on your behalf. Working with [private endpoints](#), [virtual network service endpoints](#), and other network security options can limit the availability of Speech Studio features. You normally use Speech Studio when working with features, like [custom speech](#), [custom voice](#) and [audio content creation](#).

Reaching Speech Studio web portal from a Virtual network

To use Speech Studio from a virtual machine within an Azure Virtual network, you must allow outgoing connections to the required set of [service tags](#) for this virtual network. See details

here.

Access to the Speech resource endpoint is *not* equal to access to Speech Studio web portal. Access to Speech Studio web portal via private or Virtual Network service endpoints is not supported.

Working with Speech Studio projects

This section describes working with the different kind of Speech Studio projects for the different network security options of the Speech resource. It's expected that the web browser connection to Speech Studio is established. Speech resource network security settings are set in Azure portal.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Speech resource.
3. In the **Resource Management** group in the left pane, select **Networking > Firewalls and virtual networks**.
4. Select one option from **All networks**, **Selected Networks** and **Private Endpoints**, or **Disabled**.

Custom speech, Custom voice and Audio Content Creation

The following table describes custom speech/custom voice/audio content creation project accessibility per Speech resource **Networking > Firewalls and virtual networks** security setting.

ⓘ Note

If you allow only private endpoints via the **Networking > Private endpoint connections** tab, then you can't use Speech Studio with the Speech resource. You can still use the Speech resource outside of Speech Studio.

 [Expand table](#)

Speech resource network security setting	Speech Studio project accessibility
All networks	No restrictions
Selected Networks and Private Endpoints	Accessible from allowed public IP addresses
Disabled	Not accessible

If you select **Selected Networks and private endpoints**, then you will see a tab with **Virtual networks** and **Firewall** access configuration options. In the **Firewall** section, you must allow at least one public IP address and use this address for the browser connection with Speech Studio.

If you allow only access via **Virtual network**, then in effect you don't allow access to the Speech resource through Speech Studio. You can still use the Speech resource outside of Speech Studio.

To use custom speech without relaxing network access restrictions on your production Speech resource, consider one of these workarounds.

- Create another Speech resource for development that can be used on a public network. Prepare your custom model in Speech Studio on the development resource, and then copy the model to your production resource. See the [Models_CopyTo](#) REST request with [Speech to text REST API](#).
- You have the option to not use Speech Studio for custom speech. Use the [Speech to text REST API](#) for all custom speech operations.

To use custom voice without relaxing network access restrictions on your production Speech resource, consider Use the [Custom voice REST API](#) for all custom voice operations.

Adjust an application to use an AI Foundry resource for Speech without private endpoints

In this article, we noted several times that enabling a custom domain for an AI Foundry resource for Speech is irreversible. Such a resource uses a different way of communicating with Speech service, compared to the ones that are using [regional endpoint names](#).

This section explains how to use an AI Foundry resource for Speech with a custom domain name but without any private endpoints with the Speech service REST APIs and [Speech SDK](#). This might be a resource that was once used in a private endpoint scenario, but then had its private endpoints deleted.

DNS configuration

Remember how a custom domain DNS name of the private-endpoint-enabled Speech resource is [resolved from public networks](#). In this case, the IP address resolved points to a proxy endpoint for a virtual network. That endpoint is used for dispatching the network traffic to the private-endpoint-enabled Azure AI Foundry resource.

However, when *all* resource private endpoints are removed (or right after the enabling of the custom domain name), the CNAME record of the Speech resource is reprovisioned. It now points to the IP address of the corresponding [Azure AI services regional endpoint](#).

So the output of the `nslookup` command looks like this:

```
dos

C:\>nslookup my-private-link-speech.cognitiveservices.azure.com
Server: UnKnown
Address: fe80::1

Non-authoritative answer:
Name: apimgmthskquihpkz6d90kmhvnbrx3ms3pdubscpdfk1tsx3a.cloudapp.net
Address: 13.93.122.1
Aliases: my-private-link-speech.cognitiveservices.azure.com
          westeurope.api.cognitive.microsoft.com
          cognitiveweprod.trafficmanager.net
          cognitiveweprod.azure-api.net
          apimgttmdjylckcx6clmh2isu2wr38uqzm63s8n4ub2y3e6xs.trafficmanager.net
          cognitiveweprod-westeurope-01.regional.azure-api.net
```

Compare it with the output from [this section](#).

Speech resource with a custom domain name and without private endpoints: Usage with the REST APIs

Speech to text REST API

Speech to text REST API usage is fully equivalent to the case of [private-endpoint-enabled Speech resources](#).

Speech to text REST API for short audio and Text to speech REST API

In this case, usage of the Speech to text REST API for short audio and usage of the Text to speech REST API have no differences from the general case, with one exception. (See the following note.) You should use both APIs as described in the [Speech to text REST API for short audio](#) and [Text to speech REST API](#) documentation.

Note

When you're using the Speech to text REST API for short audio and Text to speech REST API in custom domain scenarios, use an API key passed through the `Ocp-Apim-`

`Subscription-Key` header. (See details for [Speech to text REST API for short audio](#) and [Text to speech REST API](#))

Using an authorization token and passing it to the special endpoint via the `Authorization` header will work *only* if you've turned on the **All networks** access option in the **Networking** section of your Speech resource. In other cases you will get either `Forbidden` or `BadRequest` error when trying to obtain an authorization token.

Simultaneous use of private endpoints and Virtual Network service endpoints

You can use [private endpoints](#) and [Virtual Network service endpoints](#) to access to the same Speech resource simultaneously. To enable this simultaneous use, you need to use the **Selected Networks and Private Endpoints** option in the networking settings of the Speech resource in the Azure portal. Other options aren't supported for this scenario.

Pricing

For pricing details, see [Azure Private Link pricing](#).

Learn more

- [Use Speech service through a Virtual Network service endpoint](#)
- [Azure Private Link](#)
- [Azure VNet service endpoint](#)
- [Speech SDK](#)
- [Speech to text REST API](#)
- [Text to speech REST API](#)

Use Speech service through a Virtual Network service endpoint

08/07/2025

Azure Virtual Network service endpoints help to provide secure and direct connectivity to Azure services over an optimized route on the Azure backbone network. Endpoints help you secure your critical Azure service resources to only your virtual networks. Service endpoints enable private IP addresses in the virtual network to reach the endpoint of an Azure service without needing a public IP address on the virtual network.

This article explains how to set up and use Virtual Network service endpoints with Speech service in Azure AI services.

 Note

Before you start, review [how to use virtual networks with Azure AI services](#).

This article also describes [how to remove Virtual Network service endpoints later but still use the Speech resource](#).

To set up an AI Foundry resource for Speech for Virtual Network service endpoint scenarios, you need to:

1. [Create a custom domain name for the Speech resource](#).
2. [Configure virtual networks and networking settings for the Speech resource](#).
3. [Adjust existing applications and solutions](#).

 Note

Setting up and using Virtual Network service endpoints for Speech service is similar to setting up and using private endpoints. In this article, we refer to the corresponding sections of the [article on using private endpoints](#) when the procedures are the same.

Private endpoints and Virtual Network service endpoints

Azure provides private endpoints and Virtual Network service endpoints for traffic that tunnels via the [private Azure backbone network](#). The purpose and underlying technologies of these

endpoint types are similar. But there are differences between the two technologies. We recommend that you learn about the pros and cons of both before you design your network.

There are a few things to consider when you decide which technology to use:

- Both technologies ensure that traffic between the virtual network and the Speech resource doesn't travel over the public internet.
- A private endpoint provides a dedicated private IP address for your Speech resource. This IP address is accessible only within a specific virtual network and subnet. You have full control of the access to this IP address within your network infrastructure.
- Virtual Network service endpoints don't provide a dedicated private IP address for the Speech resource. Instead, they encapsulate all packets sent to the Speech resource and deliver them directly over the Azure backbone network.
- Both technologies support on-premises scenarios. By default, when they use Virtual Network service endpoints, Azure service resources secured to virtual networks can't be reached from on-premises networks. But you can [change that behavior](#).
- Virtual Network service endpoints are often used to restrict the access for an AI Foundry resource for Speech based on the virtual networks from which the traffic originates.
- For Azure AI services, enabling the Virtual Network service endpoint forces the traffic for all Azure AI Foundry resources to go through the private backbone network. That requires explicit network access configuration. (For more information, see [Configure virtual networks and the Speech resource networking settings](#).) Private endpoints don't have this limitation and provide more flexibility for your network configuration. You can access one resource through the private backbone and another through the public internet by using the same subnet of the same virtual network.
- Private endpoints incur [extra costs ↗](#). Virtual Network service endpoints are free.
- Private endpoints require [extra DNS configuration](#).
- One Speech resource can work simultaneously with both private endpoints and Virtual Network service endpoints.

We recommend that you try both endpoint types before you make a decision about your production design.

For more information, see these resources:

- [Azure Private Link and private endpoint documentation](#)
- [Virtual Network service endpoints documentation](#)

This article describes how to use Virtual Network service endpoints with Speech service. For information about private endpoints, see [Use Speech service through a private endpoint](#).

Create a custom domain name

Virtual Network service endpoints require a [custom subdomain name for Azure AI services](#). Create a custom domain by following the [guidance](#) in the private endpoint article. All warnings in the section also apply to Virtual Network service endpoints.

Configure virtual networks and the Speech resource networking settings

You need to add all virtual networks that are allowed access via the service endpoint to the Speech resource networking properties.

Note

To access an AI Foundry resource for Speech via the Virtual Network service endpoint, you need to enable the `Microsoft.CognitiveServices` service endpoint type for the required subnets of your virtual network. Doing so will route all subnet traffic related to Azure AI services through the private backbone network. If you intend to access any other Azure AI Foundry resources from the same subnet, make sure these resources are configured to allow your virtual network.

If a virtual network isn't added as *allowed* in the Speech resource networking properties, it won't have access to the Speech resource via the service endpoint, even if the `Microsoft.CognitiveServices` service endpoint is enabled for the virtual network. And if the service endpoint is enabled but the virtual network isn't allowed, the Speech resource won't be accessible for the virtual network through a public IP address, no matter what the Speech resource's other network security settings are. That's because enabling the `Microsoft.CognitiveServices` endpoint routes all traffic related to Azure AI services through the private backbone network, and in this case the virtual network should be explicitly allowed to access the resource. This guidance applies for all Azure AI Foundry resources, not just for Speech resources.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Speech resource.
3. In the **Resource Management** group in the left pane, select **Networking**.
4. On the **Firewalls and virtual networks** tab, select **Selected Networks and Private Endpoints**.

Note

To use Virtual Network service endpoints, you need to select the **Selected Networks and Private Endpoints** network security option. No other options are supported. If your scenario requires the **All networks** option, consider using [private endpoints](#), which support all three network security options.

5. Select **Add existing virtual network** or **Add new virtual network** and provide the required parameters. Select **Add** for an existing virtual network or **Create** for a new one. If you add an existing virtual network, the `Microsoft.CognitiveServices` service endpoint is automatically enabled for the selected subnets. This operation can take up to 15 minutes. Also, see the note at the beginning of this section.

Enabling service endpoint for an existing virtual network

As described in the previous section, when you configure a virtual network as *allowed* for the Speech resource, the `Microsoft.CognitiveServices` service endpoint is automatically enabled. If you later disable it, you need to re-enable it manually to restore the service endpoint access to the Speech resource (and to other Azure AI Foundry resources):

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the virtual network.
3. In the **Settings** group in the left pane, select **Subnets**.
4. Select the required subnet.
5. A new panel appears on the right side of the window. In this panel, in the **Service Endpoints** section, select `Microsoft.CognitiveServices` in the **Services** list.
6. Select **Save**.

Adjust existing applications and solutions

an AI Foundry resource for Speech that has a custom domain enabled interacts with the Speech service in a different way. This is true for a custom-domain-enabled Speech resource regardless of whether service endpoints are configured. Information in this section applies to both scenarios.

Use an AI Foundry resource for Speech that has a custom domain name and allowed virtual networks

In this scenario, the **Selected Networks and Private Endpoints** option is selected in the networking settings of the Speech resource and at least one virtual network is allowed. This scenario is equivalent to [using an AI Foundry resource for Speech that has a custom domain name and a private endpoint enabled](#).

Use an AI Foundry resource for Speech that has a custom domain name but that doesn't have allowed virtual networks

In this scenario, private endpoints aren't enabled and one of these statements is true:

- The **Selected Networks and Private Endpoints** option is selected in the networking settings of the Speech resource, but no allowed virtual networks are configured.
- The **All networks** option is selected in the networking settings of the Speech resource.

This scenario is equivalent to [using an AI Foundry resource for Speech that has a custom domain name and that doesn't have private endpoints](#).

Use of Speech Studio

[Speech Studio](#) is a web portal with tools for building and integrating Azure AI Speech service in your application. When you work in Speech Studio projects, network connections and API calls to the corresponding Speech resource are made on your behalf. Working with [private endpoints](#), [virtual network service endpoints](#), and other network security options can limit the availability of Speech Studio features. You normally use Speech Studio when working with features, like [custom speech](#), [custom voice](#) and [audio content creation](#).

Reaching Speech Studio web portal from a Virtual network

To use Speech Studio from a virtual machine within an Azure Virtual network, you must allow outgoing connections to the required set of [service tags](#) for this virtual network. See details [here](#).

Access to the Speech resource endpoint is *not* equal to access to Speech Studio web portal. Access to Speech Studio web portal via private or Virtual Network service endpoints is not supported.

Working with Speech Studio projects

This section describes working with the different kind of Speech Studio projects for the different network security options of the Speech resource. It's expected that the web browser connection to Speech Studio is established. Speech resource network security settings are set in Azure portal.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Speech resource.

3. In the **Resource Management** group in the left pane, select **Networking > Firewalls and virtual networks**.
4. Select one option from **All networks**, **Selected Networks** and **Private Endpoints**, or **Disabled**.

Custom speech, Custom voice and Audio Content Creation

The following table describes custom speech/custom voice/audio content creation project accessibility per Speech resource **Networking > Firewalls and virtual networks** security setting.

 **Note**

If you allow only private endpoints via the **Networking > Private endpoint connections** tab, then you can't use Speech Studio with the Speech resource. You can still use the Speech resource outside of Speech Studio.

 Expand table

Speech resource network security setting	Speech Studio project accessibility
All networks	No restrictions
Selected Networks and Private Endpoints	Accessible from allowed public IP addresses
Disabled	Not accessible

If you select **Selected Networks and private endpoints**, then you will see a tab with **Virtual networks** and **Firewall** access configuration options. In the **Firewall** section, you must allow at least one public IP address and use this address for the browser connection with Speech Studio.

If you allow only access via **Virtual network**, then in effect you don't allow access to the Speech resource through Speech Studio. You can still use the Speech resource outside of Speech Studio.

To use custom speech without relaxing network access restrictions on your production Speech resource, consider one of these workarounds.

- Create another Speech resource for development that can be used on a public network. Prepare your custom model in Speech Studio on the development resource, and then copy the model to your production resource. See the [Models_CopyTo](#) REST request with [Speech to text REST API](#).

- You have the option to not use Speech Studio for custom speech. Use the [Speech to text REST API](#) for all custom speech operations.

To use custom voice without relaxing network access restrictions on your production Speech resource, consider Use the [Custom voice REST API](#) for all custom voice operations.

Simultaneous use of private endpoints and Virtual Network service endpoints

You can use [private endpoints](#) and [Virtual Network service endpoints](#) to access to the same Speech resource simultaneously. To enable this simultaneous use, you need to use the **Selected Networks and Private Endpoints** option in the networking settings of the Speech resource in the Azure portal. Other options aren't supported for this scenario.

Learn more

- [Use Speech service through a private endpoint](#)
- [Azure Virtual Network service endpoints](#)
- [Azure Private Link](#)
- [Speech SDK](#)
- [Speech to text REST API](#)
- [Text to speech REST API](#)

How to monitor and control service connections with the Speech SDK

08/07/2025

The `SpeechRecognizer` and other objects in the Speech SDK automatically connect to the Speech service when it's appropriate. Sometimes, you either want extra control over when connections begin and end or you want more information about when the Speech SDK establishes or loses its connection. The supporting `Connection` class provides this capability.

Retrieve a Connection object

A `Connection` can be obtained from most top-level Speech SDK objects via a static `From...` factory method, for example, `Connection::FromRecognizer(recognizer)` for `SpeechRecognizer`.

```
var connection = Connection.FromRecognizer(recognizer);
```

Monitor for connections and disconnections

A `Connection` raises `Connected` and `Disconnected` events when the corresponding status change happens in the Speech SDK's connection to the Speech service. You can listen to these events to know the latest connection state.

```
connection.Connected += (sender, connectedEventArgs) =>
{
    Console.WriteLine($"Successfully connected, sessionId:
{connectedEventArgs.SessionId}");
};

connection.Disconnected += (sender, disconnectedEventArgs) =>
{
    Console.WriteLine($"Disconnected, sessionId:
{disconnectedEventArgs.SessionId}");
};
```

Connect and disconnect

`Connection` has explicit methods to start or end a connection to the Speech service. Reasons you might want to control the connection include:

- Preconnecting to the Speech service to allow the first interaction to start as quickly as possible.
- Establishing connection at a specific time in your application's logic to gracefully and predictably handle initial connection failures.
- Disconnecting to clear an idle connection when you don't expect immediate reconnection but also don't want to destroy the object.

Some important notes on the behavior when manually modifying connection state:

- Trying to connect when already connected doesn't generate an error. Monitor the `Connected` and `Disconnected` events if you want to know the current state of the connection.
- A failure to connect that originates from a problem that has no involvement with the Speech service--such as attempting to do so from an invalid state--results in an error as appropriate to the programming language. Failures that require network resolution--such as authentication failures--doesn't result in an error but instead generate a `Canceled` event on the top-level object the `Connection` was created from.
- Manually disconnecting from the Speech service during an ongoing interaction results in a connection error and loss of data for that interaction. Connection errors are surfaced on the appropriate top-level object's `Canceled` event.

```
try
{
    connection.Open(forContinuousRecognition: false);
}
catch (ApplicationException ex)
{
    Console.WriteLine($"Couldn't pre-connect. Details: {ex.Message}");
}
// ... Use the SpeechRecognizer
connection.Close();
```

Next steps

[Explore our samples on GitHub](#)

Configure OpenSSL for Linux

08/07/2025

With the Speech SDK, [OpenSSL](#) is dynamically configured to the host-system version.

! Note

This article is only applicable where the Speech SDK is [supported on Linux](#).

To ensure connectivity, verify that OpenSSL certificates are installed in your system. Run a command:

Bash

```
openssl version -d
```

The output on Ubuntu/Debian based systems should be:

```
OPENSSLDIR: "/usr/lib/ssl"
```

Check whether there's a `certs` subdirectory under OPENSSLDIR. In the previous example, it would be `/usr/lib/ssl/certs`.

- If the `/usr/lib/ssl/certs` exists, and if it contains many individual certificate files (with `.crt` or `.pem` extension), there's no need for further actions.
- If OPENSSLDIR is something other than `/usr/lib/ssl` or there's a single certificate bundle file instead of multiple individual files, you need to set an appropriate SSL environment variable to indicate where the certificates can be found.

Examples

Here are some example environment variables to configure per OpenSSL directory.

- OPENSSLDIR is `/opt/ssl`. There's a `certs` subdirectory with many `.crt` or `.pem` files. Set the environment variable `SSL_CERT_DIR` to point at `/opt/ssl/certs` before using the Speech SDK. For example:

Bash

```
export SSL_CERT_DIR=/opt/ssl/certs
```

- OPENSSLDIR is `/etc/pki/tls` (like on RHEL based systems). There's a `certs` subdirectory with a certificate bundle file, for example `ca-bundle.crt`. Set the environment variable `SSL_CERT_FILE` to point at that file before using the Speech SDK. For example:

Bash

```
export SSL_CERT_FILE=/etc/pki/tls/certs/ca-bundle.crt
```

Certificate revocation checks

When the Speech SDK connects to the Speech service, it checks the Transport Layer Security (TLS/SSL) certificate. The Speech SDK verifies that the certificate reported by the remote endpoint is trusted and isn't revoked. This verification provides a layer of protection against attacks involving spoofing and other related vectors. The check is accomplished by retrieving a certificate revocation list (CRL) from a certificate authority (CA) used by Azure. A list of Azure CA download locations for updated TLS CRLs can be found in [this document](#).

If a destination posing as the Speech service reports a revoked certificate in a retrieved CRL, the SDK terminates the connection and reports an error via a `canceled` event. The authenticity of a reported certificate can't be checked without an updated CRL. Therefore, the Speech SDK also treats a failure to download a CRL from an Azure CA location as an error.

Warning

If your solution uses proxy or firewall it should be configured to allow access to all certificate revocation list URLs used by Azure. Note that many of these URLs are outside of `microsoft.com` domain, so allowing access to `*.microsoft.com` is not enough. See [this document](#) for details. In exceptional cases you may ignore CRL failures (see [the correspondent section](#)), but such configuration is strongly not recommended, especially for production scenarios.

Large CRL files (>10 MB)

One cause of CRL-related failures is the use of large CRL files. This class of error is typically only applicable to special environments with extended CA chains. Standard public endpoints shouldn't encounter this class of issue.

The default maximum CRL size used by the Speech SDK (10 MB) can be adjusted per config object. The property key for this adjustment is `CONFIG_MAX_CRL_SIZE_KB` and the value, specified as a string, is by default "10000" (10 MB). For example, when creating a `SpeechRecognizer` object (that manages a connection to the Speech service), you can set this property in its `SpeechConfig`. In the following code snippet, the configuration is adjusted to permit a CRL file size up to 15 MB.

```
C#
```

```
config SetProperty("CONFIG_MAX_CRL_SIZE_KB", "15000");
```

Bypassing or ignoring CRL failures

If an environment can't be configured to access an Azure CA location, the Speech SDK can't retrieve an updated CRL. You can configure the SDK either to continue and log download failures or to bypass all CRL checks.

Warning

CRL checks are a security measure and bypassing them increases susceptibility to attacks. They should not be bypassed without thorough consideration of the security implications and alternative mechanisms for protecting against the attack vectors that CRL checks mitigate.

To continue with the connection when a CRL can't be retrieved, set the property `"OPENSSL_CONTINUE_ON_CRL_DOWNLOAD_FAILURE"` to `"true"`. An attempt is still made to retrieve a CRL and failures is still emitted in logs, but connection attempts are allowed to continue.

```
C#
```

```
config SetProperty("OPENSSL_CONTINUE_ON_CRL_DOWNLOAD_FAILURE", "true");
```

To turn off certificate revocation checks, set the property `"OPENSSL_DISABLE_CRL_CHECK"` to `"true"`. Then, while connecting to the Speech service, there's no attempt to check or download a CRL and no automatic verification of a reported TLS/SSL certificate.

```
C#
```

```
config SetProperty("OPENSSL_DISABLE_CRL_CHECK", "true");
```

CRL caching and performance

By default, the Speech SDK will cache a successfully downloaded CRL on disk to improve the initial latency of future connections. When no cached CRL is present or when the cached CRL is expired, a new list is downloaded.

Some Linux distributions don't have a `TMP` or `TMPDIR` environment variable defined, so the Speech SDK doesn't cache downloaded CRLs. Without `TMP` or `TMPDIR` environment variable defined, the Speech SDK downloads a new CRL for each connection. To improve initial connection performance in this situation, you can [create a `TMPDIR` environment variable and set it to the accessible path of a temporary directory](#).

Related content

- [Speech SDK overview](#)
- [Install the Speech SDK](#)

Configure Azure AI services virtual networks

05/19/2025

Azure AI services provide a layered security model. This model enables you to secure your Azure AI services accounts to a specific subset of networks. When network rules are configured, only applications that request data over the specified set of networks can access the account. You can limit access to your resources with *request filtering*, which allows requests that originate only from specified IP addresses, IP ranges, or from a list of subnets in [Azure Virtual Networks](#).

An application that accesses an AI Foundry resource when network rules are in effect requires authorization. Authorization is supported with [Microsoft Entra ID](#) credentials or with a valid API key.

Important

Turning on firewall rules for your Azure AI services account blocks incoming requests for data by default. To allow requests through, one of the following conditions needs to be met:

- The request originates from a service that operates within an Azure Virtual Network on the allowed subnet list of the target Azure AI services account. The endpoint request that originated from the virtual network needs to be set as the [custom subdomain](#) of your Azure AI services account.
- The request originates from an allowed list of IP addresses.

Requests that are blocked include those from other Azure services, from the Azure portal, and from logging and metrics services.

Note

We recommend that you use the Azure Az PowerShell module to interact with Azure. To get started, see [Install Azure PowerShell](#). To learn how to migrate to the Az PowerShell module, see [Migrate Azure PowerShell from AzureRM to Az](#).

Scenarios

To secure your Azure AI services resource, you should first configure a rule to deny access to traffic from all networks, including internet traffic, by default. Then, configure rules that grant access to traffic from specific virtual networks. This configuration enables you to build a secure network boundary for your applications. You can also configure rules to grant access to traffic from select public internet IP address ranges and enable connections from specific internet or on-premises clients.

Network rules are enforced on all network protocols to Azure AI services, including REST and WebSocket. To access data by using tools such as the Azure test consoles, explicit network rules must be configured. You can apply network rules to existing Azure AI services resources, or when you create new Azure AI services resources. After network rules are applied, they're enforced for all requests.

Supported regions and service offerings

Virtual networks are supported in [regions where Azure AI services are available](#). Azure AI services support service tags for network rules configuration. The services listed here are included in the `CognitiveServicesManagement` service tag.

- ✓ Anomaly Detector
- ✓ Azure OpenAI
- ✓ Content Moderator
- ✓ Custom Vision
- ✓ Face
- ✓ Language Understanding (LUIS)
- ✓ Personalizer
- ✓ Speech service
- ✓ Language
- ✓ QnA Maker
- ✓ Translator

Note

If you use Azure OpenAI, LUIS, Speech Services, or Language services, the `CognitiveServicesManagement` tag only enables you to use the service by using the SDK or REST API. To access and use the [Azure AI Foundry portal](#), LUIS portal, Speech Studio, or Language Studio from a virtual network, you need to use the following tags:

- `AzureActiveDirectory`
- `AzureFrontDoor.Frontend`
- `AzureResourceManager`

- CognitiveServicesManagement
- CognitiveServicesFrontEnd
- Storage (Speech Studio only)

For information on [Azure AI Foundry portal](#) configurations, see the [Azure AI Foundry documentation](#).

Change the default network access rule

By default, Azure AI services resources accept connections from clients on any network. To limit access to selected networks, you must first change the default action.

Warning

Making changes to network rules can impact your applications' ability to connect to Azure AI services. Setting the default network rule to *deny* blocks all access to the data unless specific network rules that *grant* access are also applied.

Before you change the default rule to deny access, be sure to grant access to any allowed networks by using network rules. If you allow listing for the IP addresses for your on-premises network, be sure to add all possible outgoing public IP addresses from your on-premises network.

Manage default network access rules

You can manage default network access rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

Azure portal

1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.

The screenshot shows the Azure portal interface for a 'contoso-custom-vision' resource. On the left, a navigation menu is open with 'Resource Management' expanded, and 'Networking' is highlighted with a red box. In the main content area, the 'Firewalls and virtual networks' tab is selected. Under 'Allow access from', the radio button for 'Selected Networks and Private Endpoints' is selected, while 'All networks' and 'Disabled' are unselected. A note at the bottom states: 'Access control settings allowing access to Azure AI services account will remain in effect for up to three minutes after saving updated settings restricting access.' There are tabs for 'Virtual networks' and 'Firewall' with their respective configuration options.

3. To deny access by default, under Firewalls and virtual networks, select Selected Networks and Private Endpoints.

With this setting alone, unaccompanied by configured virtual networks or address ranges, all access is effectively denied. When all access is denied, requests that attempt to consume the Azure AI services resource aren't permitted. The Azure portal, Azure PowerShell, or the Azure CLI can still be used to configure the Azure AI services resource.

4. To allow traffic from all networks, select All networks.

This screenshot shows the same 'contoso-custom-vision' resource in the Azure portal. The 'Networking' section is visible, and the 'Allow access from' dropdown shows that the 'All networks' option is selected, indicated by a red box around the radio button. A note below says: 'All networks, including the internet, can access this resource.' The rest of the interface is identical to the previous screenshot, with the 'Selected Networks and Private Endpoints' option being the alternative choice.

5. Select Save to apply your changes.

Grant access from a virtual network

You can configure Azure AI services resources to allow access from specific subnets only. The allowed subnets might belong to a virtual network in the same subscription or in a different subscription. The other subscription can belong to a different Microsoft Entra tenant. When the subnet belongs to a different subscription, the `Microsoft.CognitiveServices` resource provider needs to be also registered for that subscription.

Enable a *service endpoint* for Azure AI services within the virtual network. The service endpoint routes traffic from the virtual network through an optimal path to the Azure AI service. For more information, see [Virtual Network service endpoints](#).

The identities of the subnet and the virtual network are also transmitted with each request. Administrators can then configure network rules for the Azure AI services resource to allow requests from specific subnets in a virtual network. Clients granted access by these network rules must continue to meet the authorization requirements of the Azure AI services resource to access the data.

Each Azure AI services resource supports up to 100 virtual network rules, which can be combined with IP network rules. For more information, see [Grant access from an internet IP range](#) later in this article.

Set required permissions

To apply a virtual network rule to an AI Foundry resource, you need the appropriate permissions for the subnets to add. The required permission is the default *Contributor* role or the *Cognitive Services Contributor* role. Required permissions can also be added to custom role definitions.

The Azure AI services resource and the virtual networks that are granted access might be in different subscriptions, including subscriptions that are part of a different Microsoft Entra tenant.

Note

Configuration of rules that grant access to subnets in virtual networks that are a part of a different Microsoft Entra tenant are currently supported only through PowerShell, the Azure CLI, and the REST APIs. You can view these rules in the Azure portal, but you can't configure them.

Configure virtual network rules

You can manage virtual network rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

Azure portal

To grant access to a virtual network with an existing network rule:

1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.
3. Confirm that you selected **Selected Networks and Private Endpoints**.
4. Under **Allow access from**, select **Add existing virtual network**.

The screenshot shows the Azure portal interface for managing networking rules. The main title is 'contoso-custom-vision | Networking'. On the left, there's a sidebar with various options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Resource Management (which is expanded), Keys and Endpoint, Encryption, Pricing tier, Networking (which is selected and highlighted in grey), Identity, Cost analysis, Properties, and Locks. The main content area has tabs for 'Firewalls and virtual networks' (selected) and 'Private endpoint connections'. Below that, there are buttons for Save, Discard, and Refresh. Under 'Allow access from', there are three radio buttons: 'All networks' (unchecked), 'Selected Networks and Private Endpoints' (checked and highlighted with a red box), and 'Disabled' (unchecked). A note says 'Configure network security for your Azure AI services account. Learn more.' Below this is a section for 'Virtual networks' with a table header: Virtual Network, Subnet, Address range, Endpoint Status, and Resource group. A note says 'Secure your Azure AI services account with virtual networks.' with buttons '+ Add existing virtual network' (highlighted with a red box) and '+ Add new virtual network'. The table below shows 'No network selected.' At the bottom, there's a 'Firewall' section with a note 'Add IP ranges to allow access from the internet or your on-premises networks. Learn more.' and a checkbox 'Add your client IP address'. There's also an 'Address range' input field with placeholder 'IP address or CIDR' and a search icon.

5. Select the **Virtual networks** and **Subnets** options, and then select **Enable**.

Add networks

X

Subscription *

Contoso Subscription

Virtual networks *

contoso-rg

Subnets *

default (Service endpoint required)

i The following networks don't have service endpoints enabled for 'Microsoft.CognitiveServices'. Enabling access will take up to 15 minutes to complete. After starting this operation, it is safe to leave and return later if you do not wish to wait.

Virtual network	Service endpoint status
▼ contoso-rg	...
default	Not enabled

Enable

! Note

If a service endpoint for Azure AI services wasn't previously configured for the selected virtual network and subnets, you can configure it as part of this operation.

Currently, only virtual networks that belong to the same Microsoft Entra tenant are available for selection during rule creation. To grant access to a subnet in a virtual network that belongs to another tenant, use PowerShell, the Azure CLI, or the REST APIs.

6. Select **Save** to apply your changes.

To create a new virtual network and grant it access:

1. On the same page as the previous procedure, select **Add new virtual network**.

The screenshot shows the Azure portal interface for managing networking. The left sidebar has a 'Networking' section selected. The main area shows the 'Firewalls and virtual networks' tab. Under 'Allow access from', the 'Selected Networks and Private Endpoints' radio button is selected. Below this, there's a table for 'Virtual networks' with columns: Virtual Network, Subnet, Address range, Endpoint Status, and Resource group. A red box highlights the '+ Add new virtual network' button. On the right, there's a 'Firewall' section with an 'Address range' input field and a search icon.

2. Provide the information necessary to create the new virtual network, and then select **Create**.

Create virtual network

X

* Name

widgets-vnet



* Address space ⓘ

10.1.0.0/16

10.1.0.0 - 10.1.255.255 (65536 addresses)

* Subscription

widgets-subscription



* Resource group

widgets-resource-group



[Create new](#)

* Location

(US) West US 2



Subnet

* Name

default

* Address range ⓘ

10.1.0.0/24



10.1.0.0 - 10.1.0.255 (256 addresses)

DDoS protection ⓘ

Basic Standard

Service endpoint ⓘ

Microsoft.CognitiveServices

Firewall ⓘ

[Create](#)

3. Select **Save** to apply your changes.

To remove a virtual network or subnet rule:

1. On the same page as the previous procedures, select ...(**More options**) to open the context menu for the virtual network or subnet, and select **Remove**.

The screenshot shows the 'Firewalls and virtual networks' section of the Azure portal. At the top, there are buttons for 'Save', 'Discard', and 'Refresh'. Below that, it says 'Allow access from' with three options: 'All networks' (radio button), 'Selected Networks and Private Endpoints' (selected radio button), and 'Disabled'. A note says 'Configure network security for your Azure AI services account. [Learn more](#)'. Under 'Virtual networks', there's a table with columns 'Virtual Network', 'Subnet', 'Address range', 'Endpoint Status', 'Resource group', and 'Subscription'. One row is shown: 'contoso-01-vnet', '1', 'Address range' (empty), 'Enabled', 'contoso-rg', and 'Subscription'. To the right of this row are 'Remove' and '...' buttons, both with red boxes around them. Below the table is a 'Firewall' section with a note to 'Add IP ranges to allow access from the internet or your on-premises networks. [Learn more](#)' and a checkbox for 'Add your client IP address'. There's also an 'Address range' section with a search icon.

2. Select **Save** to apply your changes.

ⓘ Important

Be sure to [set the default rule](#) to *deny*, or network rules have no effect.

Grant access from an internet IP range

You can configure Azure AI services resources to allow access from specific public internet IP address ranges. This configuration grants access to specific services and on-premises networks, which effectively block general internet traffic.

You can specify the allowed internet address ranges by using [CIDR format \(RFC 4632\)](#) in the form `192.168.0.0/16` or as individual IP addresses like `192.168.0.1`.

💡 Tip

Small address ranges that use `/31` or `/32` prefix sizes aren't supported. Configure these ranges by using individual IP address rules.

IP network rules are only allowed for *public internet* IP addresses. IP address ranges reserved for private networks aren't allowed in IP rules. Private networks include addresses that start with `10.*`, `172.16.*` - `172.31.*`, and `192.168.*`. For more information, see [Private Address Space \(RFC 1918\)](#).

Currently, only IPv4 addresses are supported. Each Azure AI services resource supports up to 100 IP network rules, which can be combined with [virtual network rules](#).

Configure access from on-premises networks

To grant access from your on-premises networks to your Azure AI services resource with an IP network rule, identify the internet-facing IP addresses used by your network. Contact your network administrator for help.

If you use Azure ExpressRoute on-premises for Microsoft peering, you need to identify the NAT IP addresses. For more information, see [What is Azure ExpressRoute](#).

For Microsoft peering, the NAT IP addresses that are used are either customer provided or supplied by the service provider. To allow access to your service resources, you must allow these public IP addresses in the resource IP firewall setting.

Managing IP network rules

You can manage IP network rules for Azure AI services resources through the Azure portal, PowerShell, or the Azure CLI.

Azure portal

1. Go to the Azure AI services resource you want to secure.
2. Select **Resource Management** to expand it, then select **Networking**.
3. Confirm that you selected **Selected Networks and Private Endpoints**.
4. Under **Firewalls and virtual networks**, locate the **Address range** option. To grant access to an internet IP range, enter the IP address or address range (in [CIDR format](#)). Only valid public IP (nonreserved) addresses are accepted.

Firewalls and virtual networks Private endpoint connections

Save Discard Refresh

Allow access from:

All networks Selected Networks and Private Endpoints Disabled

Configure network security for your Azure AI services account. [Learn more.](#)

Virtual networks

Secure your Azure AI services account with virtual networks. [+ Add existing virtual network](#) [+ Add new virtual network](#)

Virtual Network	Subnet	Address range	Endpoint Status	Resource group
No network selected.				

Firewall

Add IP ranges to allow access from the internet or your on-premises networks. [Learn more.](#)

Add your client IP address [?](#)

Address range



To remove an IP network rule, select the trash can  icon next to the address range.

5. Select **Save** to apply your changes.

 **Important**

Be sure to [set the default rule](#) to *deny*, or network rules have no effect.

Use private endpoints

You can use [private endpoints](#) for your Azure AI services resources to allow clients on a virtual network to securely access data over [Azure Private Link](#). The private endpoint uses an IP address from the virtual network address space for your Azure AI services resource. Network traffic between the clients on the virtual network and the resource traverses the virtual network and a private link on the Microsoft Azure backbone network, which eliminates exposure from the public internet.

Private endpoints for Azure AI services resources let you:

- Secure your Azure AI services resource by configuring the firewall to block all connections on the public endpoint for the Azure AI service.
- Increase security for the virtual network, by enabling you to block exfiltration of data from the virtual network.

- Securely connect to Azure AI services resources from on-premises networks that connect to the virtual network by using [Azure VPN Gateway](#) or [ExpressRoutes](#) with private-peering.

Understand private endpoints

A private endpoint is a special network interface for an Azure resource in your [virtual network](#). Creating a private endpoint for your Azure AI services resource provides secure connectivity between clients in your virtual network and your resource. The private endpoint is assigned an IP address from the IP address range of your virtual network. The connection between the private endpoint and the Azure AI service uses a secure private link.

Applications in the virtual network can connect to the service over the private endpoint seamlessly. Connections use the same connection strings and authorization mechanisms that they would use otherwise. The exception is Speech Services, which require a separate endpoint. For more information, see [Private endpoints with the Speech Services](#) in this article. Private endpoints can be used with all protocols supported by the Azure AI services resource, including REST.

Private endpoints can be created in subnets that use service endpoints. Clients in a subnet can connect to one Azure AI services resource using private endpoint, while using service endpoints to access others. For more information, see [Virtual Network service endpoints](#).

When you create a private endpoint for an AI Foundry resource in your virtual network, Azure sends a consent request for approval to the Azure AI services resource owner. If the user who requests the creation of the private endpoint is also an owner of the resource, this consent request is automatically approved.

Azure AI services resource owners can manage consent requests and the private endpoints through the **Private endpoint connection** tab for the Azure AI services resource in the [Azure portal](#) ↗.

Specify private endpoints

When you create a private endpoint, specify the Azure AI services resource that it connects to. For more information on creating a private endpoint, see:

- [Create a private endpoint by using the Azure portal](#)
- [Create a private endpoint by using Azure PowerShell](#)
- [Create a private endpoint by using the Azure CLI](#)

Connect to private endpoints

(!) Note

Azure OpenAI in Azure AI Foundry Models uses a different private DNS zone and public DNS zone forwarder than other Azure AI services. For the correct zone and forwarder names, see [Azure services DNS zone configuration](#).

Clients on a virtual network that use the private endpoint use the same connection string for the Azure AI services resource as clients connecting to the public endpoint. The exception is the Speech service, which requires a separate endpoint. For more information, see [Use private endpoints with the Speech service](#) in this article. DNS resolution automatically routes the connections from the virtual network to the Azure AI services resource over a private link.

By default, Azure creates a [private DNS zone](#) attached to the virtual network with the necessary updates for the private endpoints. If you use your own DNS server, you might need to make more changes to your DNS configuration. For updates that might be required for private endpoints, see [Apply DNS changes for private endpoints](#) in this article.

Use private endpoints with the Speech service

See [Use Speech service through a private endpoint](#).

Apply DNS changes for private endpoints

When you create a private endpoint, the DNS `CNAME` resource record for the Azure AI services resource is updated to an alias in a subdomain with the prefix `privatelink`. By default, Azure also creates a private DNS zone that corresponds to the `privatelink` subdomain, with the DNS A resource records for the private endpoints. For more information, see [What is Azure Private DNS](#).

When you resolve the endpoint URL from outside the virtual network with the private endpoint, it resolves to the public endpoint of the Azure AI services resource. When it's resolved from the virtual network hosting the private endpoint, the endpoint URL resolves to the private endpoint's IP address.

This approach enables access to the Azure AI services resource using the same connection string for clients in the virtual network that hosts the private endpoints and clients outside the virtual network.

If you use a custom DNS server on your network, clients must be able to resolve the fully qualified domain name (FQDN) for the Azure AI services resource endpoint to the private

endpoint IP address. Configure your DNS server to delegate your private link subdomain to the private DNS zone for the virtual network.

💡 Tip

When you use a custom or on-premises DNS server, you should configure your DNS server to resolve the Azure AI services resource name in the `privatelink` subdomain to the private endpoint IP address. Delegate the `privatelink` subdomain to the private DNS zone of the virtual network. Alternatively, configure the DNS zone on your DNS server and add the DNS A records.

For more information on configuring your own DNS server to support private endpoints, see the following resources:

- [Name resolution that uses your own DNS server](#)
- [DNS configuration](#)

Grant access to trusted Azure services for Azure OpenAI

You can grant a subset of trusted Azure services access to Azure OpenAI, while maintaining network rules for other apps. These trusted services will then use managed identity to authenticate your Azure OpenAI service. The following table lists the services that can access Azure OpenAI if the managed identity of those services have the appropriate role assignment.

expand Expand table

Service	Resource provider name
Azure AI Services	<code>Microsoft.CognitiveServices</code>
Azure Machine Learning	<code>Microsoft.MachineLearningServices</code>
Azure AI Search	<code>Microsoft.Search</code>

You can grant networking access to trusted Azure services by creating a network rule exception using the REST API or Azure portal:

Using the Azure CLI

Bash

```

accessToken=$(az account get-access-token --resource https://management.azure.com
--query "accessToken" --output tsv)
rid="/subscriptions/<your subscription id>/resourceGroups/<your resource
group>/providers/Microsoft.CognitiveServices/accounts/<your Azure AI resource
name>"

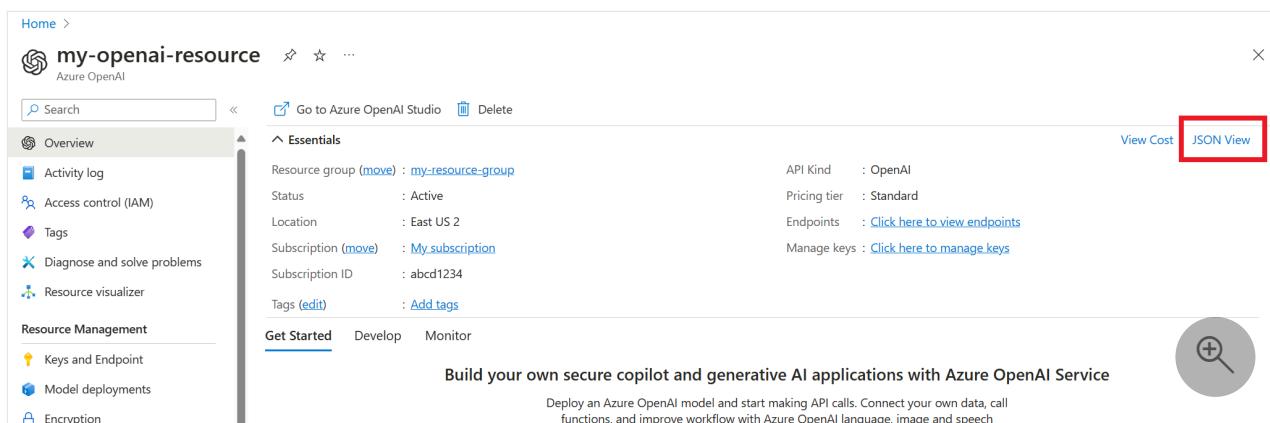
curl -i -X PATCH https://management.azure.com$rid?api-version=2023-10-01-preview \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $accessToken" \
-d \
'
{
    "properties": {
        "networkAcls": {
            "bypass": "AzureServices"
        }
    }
}
'

```

To revoke the exception, set `networkAcls.bypass` to `None`.

To verify if the trusted service has been enabled from the Azure portal,

1. Use the JSON View from the Azure OpenAI resource overview page



The screenshot shows the Azure OpenAI resource overview page for a resource named "my-openai-resource". The "Essentials" section displays resource details such as Resource group, Status, Location, Subscription, and Tags. On the right side of the page, there is a "View Cost" button and a "JSON View" button, which is highlighted with a red box. Below the essentials section, there are tabs for "Get Started", "Develop", and "Monitor". A promotional message at the bottom encourages building secure AI applications.

2. Choose your latest API version under **API versions**. Only the latest API version is supported, `2023-10-01-preview`.

Resource JSON

Resource ID

/subscriptions/

/resourceGroups/

/providers/Microsoft/

API Versions

2023-10-01-preview

```
75     "networkAcls": {  
76         "bypass": "AzureServices",  
77         "defaultAction": "Deny",  
78         "virtualNetworkRules": [],  
79         "ipRules": []  
80     },
```



Using the Azure portal

1. Navigate to your Azure OpenAI resource, and select **Networking** from the navigation menu.
2. Under **Exceptions**, select **Allow Azure services on the trusted services list to access this cognitive services account**.

Tip

You can view the **Exceptions** option by selecting either **Selected networks and private endpoints** or **Disabled** under **Allow access from**.

The screenshot shows the Azure portal interface for managing networking settings of an Azure OpenAI resource named 'test'. The left sidebar has a 'Networking' item selected, which is highlighted with a red box. The main content area shows the 'Firewalls and virtual networks' tab selected. Under 'Allow access from', the 'Selected Networks and Private Endpoints' radio button is selected and highlighted with a red box. At the bottom of the page, the 'Exceptions' checkbox is checked and highlighted with a red box, stating: 'Allow Azure services on the trusted services list to access this cognitive services account.'

Pricing

For pricing details, see [Azure Private Link pricing](#).

Next steps

- Explore the various [Azure AI services](#)
- Learn more about [Virtual Network service endpoints](#)

Speech service in sovereign clouds

08/07/2025

Azure Government (United States)

Available to US government entities and their partners only. See more information about Azure Government [here](#) and [here](#).

- **Azure portal:**
 - <https://portal.azure.us/> ↗
- **Regions:**
 - US Gov Arizona
 - US Gov Virginia
- **Available pricing tiers:**
 - Free (F0) and Standard (S0). See more details [here](#) ↗
- **Supported features:**
 - Speech to text
 - Custom speech (Acoustic Model (AM) and Language Model (LM) adaptation)
 - [Speech Studio](#) ↗
 - Text to speech
 - Standard voice
 - Neural voice
 - Speech translation
- **Unsupported features:**
 - Custom voice
 - Personal voice
 - Text to speech avatar
- **Supported languages:**
 - See the list of supported languages [here](#)

Endpoint information

This section contains Speech service endpoint information for the usage with [Speech SDK](#), [Speech to text REST API](#), and [Text to speech REST API](#).

Speech service REST API

Speech service REST API endpoints in Azure Government have the following format:

[Expand table](#)

REST API type / operation	Endpoint format
Access token	<code>https://<REGION_IDENTIFIER>.api.cognitive.microsoft.us/sts/v1.0/issueToken</code>
Speech to text REST API	<code>https://<REGION_IDENTIFIER>.api.cognitive.microsoft.us/<URL_PATH></code>
Speech to text REST API for short audio	<code>https://<REGION_IDENTIFIER>.stt.speech.azure.us/<URL_PATH></code>
Text to speech REST API	<code>https://<REGION_IDENTIFIER>.tts.speech.azure.us/<URL_PATH></code>

Replace `<REGION_IDENTIFIER>` with the identifier matching the region of your Speech resource from this table:

[Expand table](#)

Region identifier	
US Gov Arizona	<code>usgovarizona</code>
US Gov Virginia	<code>usgovvirginia</code>

Speech SDK

For [Speech SDK](#) in sovereign clouds, you need to use "from endpoint / with endpoint" instantiation of `SpeechConfig` class or `--endpoint` option of [Speech CLI](#).

`SpeechConfig` class should be instantiated like this:

```
C#  
  
C#  
  
var config = SpeechConfig.Endpoint(new Uri(usGovEndpoint), subscriptionKey);
```

Speech CLI should be used like this (note the `--endpoint` option):

```
dos  
  
spx recognize --endpoint "usGovEndpoint" --file myaudio.wav
```

Replace `subscriptionKey` with your Speech resource key. Replace `usGovEndpoint` with the endpoint from the Azure Portal.

Microsoft Azure operated by 21Vianet

Available to organizations with a business presence in China. See more information about Microsoft Azure operated by 21Vianet [here](#).

- **Azure portal:**
 - <https://portal.azure.cn/>
- **Regions:**
 - China East 2
 - China North 2
 - China North 3
- **Available pricing tiers:**
 - Free (F0) and Standard (S0). See more details [here](#)
- **Supported features:**
 - Speech to text
 - Custom speech (Acoustic Model (AM) and Language Model (LM) adaptation)
 - [Speech Studio](#)
 - [Pronunciation assessment](#)
 - Text to speech
 - Standard voice
 - Neural voice
 - Speech translator
- **Unsupported features:**
 - Custom voice
 - Personal voice
 - Text to speech avatar
- **Supported languages:**
 - See the list of supported languages [here](#)

Endpoint information

This section contains Speech service endpoint information for the usage with [Speech SDK](#), [Speech to text REST API](#), and [Text to speech REST API](#).

Speech service REST API

Speech service REST API endpoints in Azure operated by 21Vianet have the following format:

Expand table

REST API type / operation	Endpoint format
Access token	<code>https://<REGION_IDENTIFIER>.api.cognitive.azure.cn/sts/v1.0/issueToken</code>
Speech to text REST API	<code>https://<REGION_IDENTIFIER>.api.cognitive.azure.cn/<URL_PATH></code>
Speech to text REST API for short audio	<code>https://<REGION_IDENTIFIER>.stt.speech.azure.cn/<URL_PATH></code>
Text to speech REST API	<code>https://<REGION_IDENTIFIER>.tts.speech.azure.cn/<URL_PATH></code>

Replace `<REGION_IDENTIFIER>` with the identifier matching the region of your Speech resource from this table:

 Expand table

Region identifier	
China East 2	<code>chinaeast2</code>
China North 2	<code>chinanorth2</code>
China North 3	<code>chinanorth3</code>

Speech SDK

For [Speech SDK](#) in sovereign clouds, you need to use "from endpoint / with endpoint" instantiation of `SpeechConfig` class or `--endpoint` option of [Speech CLI](#).

`SpeechConfig` class should be instantiated like this:

C#

```
var config = SpeechConfig.Endpoint(new Uri(azCnEndpoint), subscriptionKey);
```

Speech CLI should be used like this (note the `--endpoint` option):

dos

```
spx recognize --endpoint "azCnEndpoint" --file myaudio.wav
```

Replace `subscriptionKey` with your Speech resource key. Replace `azCnEndpoint` with the endpoint from the Azure Portal.

Set up the Bring your own storage (BYOS) Speech resource

Bring your own storage (BYOS) is an Azure AI technology for customers, who have high requirements for data security and privacy. The core of the technology is the ability to associate an Azure Storage account, that the user owns and fully controls with the Speech resource. The Speech resource then uses this storage account for storing different artifacts related to the user data processing, instead of storing the same artifacts within the Speech service premises as it is done in the regular case. This approach allows using all set of security features of Azure Storage account, including encrypting the data with the Customer-managed keys, using Private endpoints to access the data, etc.

In BYOS scenarios, all traffic between the Speech resource and the Storage account is maintained using [Azure global network](#), in other words all communication is performed using private network, completely bypassing public internet. Speech resource in BYOS scenario is using [Azure Trusted services](#) mechanism to access the Storage account, relying on [System-assigned managed identities](#) as a method of authentication, and [Role-based access control \(RBAC\)](#) as a method of authorization.

There's one exception: if you use Text to speech, and your Speech resource and the associated Storage account are located in different Azure regions, then public internet is used for the operations, involving [User delegation SAS](#). See details in [this section](#).

BYOS can be used with several Azure AI services. For Speech, it can be used in the following scenarios:

Speech to text

- [Batch transcription](#)
- Real-time transcription with [audio and transcription result logging](#) enabled
- [Custom speech](#) - Fine-tuning of speech to text models with custom data

Text to speech

- [Audio Content Creation](#)
- [Custom voice](#) - Fine-tuning of text to speech models with custom data

One Speech / AI Services resource – Storage account combination can be used for all scenarios simultaneously in all combinations.

This article describes how to create and maintain BYOS-enabled Speech resource and is applicable to all mentioned scenarios. See the scenario-specific information in the [corresponding articles](#).

Note

For instruction on how to set up a BYOS-enabled AI Services resource go to [Connect your own storage for Speech and Language services](#) in AI Foundry.

BYOS-enabled Speech resource: Basic rules

Consider the following rules when planning BYOS-enabled Speech resource configuration:

- Speech resource can be BYOS-enabled only during creation. Existing Speech resource can't be converted to BYOS-enabled. BYOS-enabled Speech resource can't be converted to the "conventional" (non-BYOS) one.
- Storage account association with the Speech resource is declared during the Speech resource creation. It can't be changed later. That is, you can't change what Storage account is associated with the existing BYOS-enabled Speech resource. To use another Storage account, you have to create another BYOS-enabled Speech resource.
- When creating a BYOS-enabled Speech resource, you can use an existing Storage account or create one automatically during Speech resource provisioning (the latter is valid only when using Azure portal).
- One Storage account can be associated with many Speech resources. We recommend using one Storage account per one Speech resource.
- Storage account and the related BYOS-enabled Speech resource can be located in either the same or different Azure regions. We recommend using the same region to minimize latency. For the same reason, we don't recommend selecting too remote regions for multi-region configuration. (For example, we don't recommend placing Storage account in East US and the associated Speech resource in West Europe).

Create and configure BYOS-enabled Speech resource

This section describes how to create a BYOS enabled Speech resource.

Plan and prepare your Storage account

If you use Azure portal to create a BYOS-enabled Speech resource, an associated Storage account can be created automatically. For all other provisioning methods (Azure CLI, PowerShell, REST API Request) you need to use existing Storage account.

If you want to use existing Storage account and don't intend to use Azure portal method for BYOS-enabled Speech resource provisioning, note the following regarding this Storage account:

- You need the full Azure resource ID of the Storage account. To obtain it navigate to the Storage account in Azure portal, then select *Endpoints* menu from *Settings* group. Copy and store the value of *Storage account resource ID* field.
- To fully configure BYOS, you need at least *Resource Owner* right for the selected Storage account.

 **Note**

Storage account *Resource Owner* right or higher isn't required to use a BYOS-enabled Speech resource. However it's required during the one-time initial configuration of the Storage account for the usage in BYOS scenario. See details in [this section](#).

Create BYOS-enabled Speech resource

There are two ways of creating a BYOS-enabled Speech resource:

- With Azure portal.
- With Cognitive Services API (PowerShell, Azure CLI, REST request).

Azure portal option has tighter requirements:

- Account used for the BYOS-enabled Speech resource provisioning should have a right of the *Subscription Owner*.
- BYOS-associated Storage account should only be located in the same region as the Speech resource.

If any of these extra requirements don't fit your scenario, use Cognitive Services API option (PowerShell, Azure CLI, REST request).

To use any of the methods above, you need an Azure account that is assigned a role allowing to create resources in your subscription, like *Subscription Contributor*.

Azure portal

 **Note**

If you use Azure portal to create a BYOS-enabled Speech resource, we recommend selecting the option of creating a new Storage account.

To create a BYOS-enabled Speech resource with Azure portal, you need to access some portal preview features. Perform the following steps:

1. Navigate to *Create Speech* page using [this link](#).
2. Note the *Storage account* section at the bottom of the page.
3. Select *Yes* for *Bring your own storage* option.
4. Configure the required Storage account settings and proceed with the Speech resource creation.

If you used Azure portal for creating a BYOS-enabled Speech resource, it's fully ready to use. If you used any other method, you need to perform the role assignment for the Speech resource managed identity within the scope of the associated Storage account. In all cases, you also need to review different Storage account settings related to data security. See [this section](#).

(Optional) Verify Speech resource BYOS configuration

You can always check, whether any given Speech resource is BYOS enabled, and what is the associated Storage account. You can do it either via Azure portal, or via Cognitive Services API.

Azure portal

To check BYOS configuration of an AI Foundry resource for Speech with Azure portal, you need to access some portal preview features. Perform the following steps:

1. Navigate to *Create Speech* page using [this link](#).
2. Close *Create Speech* screen by pressing *X* in the right upper corner.
3. If asked agree to discard unsaved changes.
4. Navigate to the Speech resource you want to check.
5. Select *Storage* menu in the *Resource Management* group.
6. Check that:
 - a. *Attached storage* field contains the Azure resource ID of the BYOS-associated Storage account.
 - b. *Identity type* has *System Assigned* selected.

If *Storage* menu item is missing in the *Resource Management* group, the selected Speech resource isn't BYOS-enabled.

Configure BYOS-associated Storage account

To achieve high security and privacy of your data, you need to properly configure the settings of the BYOS-associated Storage account. In case you didn't use Azure portal to create your BYOS-enabled Speech resource, you also need to perform a mandatory step of role assignment.

Assign resource access role

This step is **mandatory** if you didn't use Azure portal to create your BYOS-enabled Speech resource.

BYOS uses the Blob storage of a Storage account. Because of this, BYOS-enabled Speech resource managed identity needs *Storage Blob Data Contributor* role assignment within the scope of BYOS-associated Storage account.

Caution

Don't use custom role assignments instead of built-in *Storage Blob Data Contributor* role.

Failure to do so very likely will result in hard to debug service errors and issues related to accessing BYOS-associated Storage account.

If you used Azure portal to create your BYOS-enabled Speech resource, you can skip the rest of this subsection. Your role assignment is already done. Otherwise, follow these steps.

Important

You need to be assigned the *Owner* role of the Storage account or higher scope (like Subscription) to perform the operation in the next steps. This is because only the *Owner* role can assign roles to others. See details [here](#).

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. Select *Access Control (IAM)* menu in the left pane.
4. Select *Add role assignment* in the *Grant access to this resource* tile.
5. Select *Storage Blob Data Contributor* under *Role* and then select *Next*.
6. Select *Managed identity* under *Members > Assign access to*.
7. Assign the managed identity of your Speech resource and then select *Review + assign*.
8. After confirming the settings, select *Review + assign*.

Configure Storage account security settings for Speech to text

This section describes how to set up Storage account security settings, if you intend to use BYOS-associated Storage account only for Speech to text scenarios. In case you use the BYOS-associated Storage account for Text to speech or a combination of both Speech to text and Text to speech, use [this section](#).

For Speech to text BYOS is using the [trusted Azure services security mechanism](#) to communicate with Storage account. The mechanism allows setting restricted storage account data access rules.

If you perform all actions in the section, your Storage account is in the following configuration:

- Access to all external network traffic is prohibited.
- Access to Storage account using Storage account key is prohibited.
- Access to Storage account blob storage using [shared access signatures \(SAS\)](#) is prohibited. (Except for [User delegation SAS](#))
- Access to the BYOS-enabled Speech resource is allowed using the resource [system assigned managed identity](#).

So in effect your Storage account becomes completely "locked" and can only be accessed by your Speech resource, which will be able to:

- Write artifacts of your Speech data processing (see details in the [correspondent articles](#)),
- Read the files that were already present by the time the new configuration was applied. For example, source audio files for the Batch transcription or Dataset files for Custom model training and testing.

You should consider this configuration as a model as far as the security of your data is concerned and customize it according to your needs.

For example, you can allow traffic from selected public IP addresses and Azure Virtual networks. You can also set up access to your Storage account using [private endpoints](#) (see as well [this tutorial](#)), re-enable access using Storage account key, allow access to other Azure trusted services, etc.

Note

Using [private endpoints for Speech](#) isn't required to secure the Storage account. Private endpoints for Speech secure the channels for Speech API requests, and can be used as an extra component in your solution.

Restrict access to the Storage account

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Settings* group in the left pane, select *Configuration*.
4. Select *Disabled* for *Allow Blob anonymous access*.
5. Select *Disabled* for *Allow storage account key access*
6. Select *Save*.

For more information, see [Prevent anonymous public read access to containers and blobs](#) and [Prevent Shared Key authorization for an Azure Storage account](#).

Configure Azure Storage firewall

Having restricted access to the Storage account, you need to grant networking access to your Speech resource managed identity. Follow these steps to add access for the Speech resource.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Security + networking* group in the left pane, select *Networking*.
4. In the *Firewalls and virtual networks* tab, select *Enabled from selected virtual networks and IP addresses*.
5. Deselect all check boxes.
6. Make sure *Microsoft network routing* is selected.
7. Under the *Resource instances* section, select *Microsoft.CognitiveServices/accounts* as the resource type and select your Speech resource as the instance name.
8. Select *Save*.

 **Note**

It may take up to 5 minutes for the network changes to propagate.

Configure Storage account security settings for Text to Speech

This section describes how to set up Storage account security settings, if you intend to use BYOS-associated Storage account for Text to speech or a combination of both Speech to text and Text to speech. In case you use the BYOS-associated Storage account for Speech to text only, use [this section](#).

(!) Note

Text to speech requires more relaxed settings of Storage account firewall, compared to Speech to text. If you use both Speech to text and Text to speech, and need maximally restricted Storage account security settings to protect your data, you can consider using different Storage accounts and the corresponding Speech resources for Speech to Text and Text to speech tasks.

If you perform all actions in the section, your Storage account is in the following configuration:

- External network traffic is allowed.
- Access to Storage account using Storage account key is prohibited.
- Access to Storage account blob storage using [shared access signatures \(SAS\)](#) is prohibited. (Except for [User delegation SAS](#))
- Access to the BYOS-enabled Speech resource is allowed using the resource [system assigned managed identity](#) and [User delegation SAS](#).

These are the most restricted security settings possible for the text to speech scenario. You can further customize them according to your needs.

Restrict access to the Storage account

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Settings* group in the left pane, select *Configuration*.
4. Select *Disabled* for *Allow Blob anonymous access*.
5. Select *Disabled* for *Allow storage account key access*
6. Select *Save*.

For more information, see [Prevent anonymous public read access to containers and blobs](#) and [Prevent Shared Key authorization for an Azure Storage account](#).

Configure Azure Storage firewall

Custom voice uses [User delegation SAS](#) to read the data for professional voice fine-tuning. It requires allowing external network traffic access to the Storage account.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Security + networking* group in the left pane, select *Networking*.
4. In the *Firewalls and virtual networks* tab, select *Enabled from all networks*.
5. Select *Save*.

Configure BYOS-associated Storage account for use with Speech Studio

Many [Speech Studio](#) operations like dataset upload, or custom model training and testing don't require any special configuration of a BYOS-enabled Speech resource.

However, if you need to read data stored within BYOS-associated Storage account through Speech Studio Web interface, you need to configure more settings of your BYOS-associated Storage account. For example, it's required to view the contents of a dataset.

Configure Cross-Origin Resource Sharing (CORS)

Speech Studio needs permission to make requests to the Blob storage of the BYOS-associated Storage account. To grant such permission, you use [Cross-Origin Resource Sharing \(CORS\)](#). Follow these steps.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Settings* group in the left pane, select *Resource sharing (CORS)*.
4. Ensure, that *Blob storage* tab is selected.
5. Configure the following record:
 - *Allowed origins:* `https://speech.microsoft.com`
 - *Allowed methods:* `GET, OPTIONS`
 - *Allowed headers:* `*`
 - *Exposed headers:* `*`
 - *Max age:* `1000`
6. Select *Save*.

Warning

Allowed origins field should contain URL **without** trailing slash. That is it should be `https://speech.microsoft.com`, and not `https://speech.microsoft.com/`. Adding trailing slash will result in Speech Studio not showing the details of datasets and model tests.

Configure Azure Storage firewall

You need to allow access for the machine, where you run the browser using Speech Studio. If your Storage account firewall settings allow public access from all networks, you can skip this

subsection. Otherwise, follow these steps.

1. Go to the [Azure portal](#) and sign in to your Azure account.
2. Select the Storage account.
3. In the *Security + networking* group in the left pane, select *Networking*.
4. In the *Firewall* section, enter either IP address of the machine where you run the web browser or IP subnet, to which the IP address of the machine belongs.
5. Select *Save*.

Next steps

- [Use the Bring your own storage \(BYOS\) Speech resource for Speech to text](#)
 - AI Foundry [Connect to your own storage](#)
 - AI Foundry [Connect your own storage for Speech and Language services**](#)
-

Last updated on 10/31/2025

Use the Bring your own storage (BYOS) Speech resource for speech to text

08/07/2025

Bring your own storage (BYOS) can be used in the following speech to text scenarios:

- Batch transcription
- Real-time transcription with audio and transcription results logging enabled
- Custom speech

One pair of an AI Foundry resource for Speech and storage account can be used for all scenarios simultaneously.

This article explains in depth how to use a BYOS-enabled Speech resource in all speech to text scenarios. The article implies that you have [a fully configured BYOS-enabled Speech resource and associated Storage account](#).

Data storage

When using BYOS, the Speech service doesn't keep any customer artifacts after the data processing (transcription, model training, model testing) is complete. However, some metadata that isn't derived from the user content is stored within Speech service premises. For example, in the custom speech scenario, the Service keeps certain information about the custom endpoints, like which models they use.

BYOS-associated Storage account stores the following data:

ⓘ Note

Optional in this section means that it's possible, but not required to store the particular artifacts in the BYOS-associated Storage account. If needed, they can be stored elsewhere.

Batch transcription

- Source audio (optional)
- Batch transcription results

Real-time transcription with audio and transcription result logging enabled

- Audio and transcription result logs

Custom speech

- Source files of datasets for model training and testing (optional)
- All data and metadata related to Custom models hosted by the BYOS-enabled Speech resource (including copies of datasets for model training and testing)

Batch transcription

Batch transcription is used to transcribe a large amount of audio data in storage. If you're unfamiliar with Batch transcription, see [this article](#) first.

Perform these steps to execute Batch transcription with BYOS-enabled Speech resource:

1. Start Batch transcription as described in [this guide](#).

Important

Don't use `destinationContainerUrl` parameter in your transcription request. If you use BYOS, the transcription results are stored in the BYOS-associated Storage account automatically.

If you use `destinationContainerUrl` parameter, it will work, but provide significantly less security for your data, because of ad hoc SAS usage. See details [here](#).

2. When transcription is complete, get transcription results according to [this guide](#). Consider using `sasValidityInSeconds` parameter (see the following section).

Speech service uses `customspeech-artifacts` Blob container in the BYOS-associated Storage account for storing intermediate and final transcription results.

Caution

Speech service relies on pre-defined Blob container paths and file names for Batch transcription module to correctly function. Don't move, rename or in any way alter the contents of `customspeech-artifacts` container.

Failure to do so very likely will result in hard to debug 4xx and 5xx Service errors.

Also don't build solutions that directly use files and folders of `customspeech-artifacts` container. Use standard tools to interact with Batch transcription. See details in [Batch transcription section](#).

Get Batch transcription results via REST API

Speech to text REST API fully supports BYOS-enabled Speech resources. However, because the data is now stored within the BYOS-enabled Storage account, requests like [Get Transcription Files](#) interact with the BYOS-associated Storage account Blob storage, instead of Speech service internal resources. It allows using the same REST API based code for both "regular" and BYOS-enabled Speech resources.

For maximum security use the `sasValidityInSeconds` parameter with the value set to `0` in the requests, that return data file URLs, like [Get Transcription Files](#) request. Here's an example request URL:

```
https
```

```
https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/transcriptions/aaaabb  
bb-0000-cccc-1111-dddd2222eeee/files?sasValidityInSeconds=0
```

Such a request returns direct Storage Account URLs to data files (without SAS or other additions). For example:

```
JSON
```

```
"links": {  
    "contentUrl":  
        "https://<BYOS_storage_account_name>.blob.core.windows.net/customspeech-  
        artifacts/TranscriptionData/3b24ca19-2eb1-4a2a-b964-35d89eca486b_0_0.json"  
}
```

URL of this format ensures that only Microsoft Entra identities (users, service principals, managed identities) with sufficient access rights (like *Storage Blob Data Reader* role) can access the data from the URL.

⚠ Warning

If `sasValidityInSeconds` parameter is omitted in [Get Transcription Files](#) request or similar ones, then a [User delegation SAS](#) with the validity of 5 days will be generated for each data file URL returned. This SAS is signed by the system assigned managed identity of your BYOS-enabled Speech resource. Because of it, the SAS allows access to the data, even if storage account key access is disabled. See details [here](#).

Real-time transcription with audio and transcription result logging enabled

You can enable logging for both audio input and recognized speech when using speech to text or speech translation. See the complete description [in this article](#).

If you use BYOS, then you find the logs in `customspeech-audiologs` Blob container in the BYOS-associated Storage account.

Warning

Logging data is kept for 5 days. After this period the logs are automatically deleted. This is valid for BYOS-enabled Speech resources as well. If you want to keep the logs longer, copy the correspondent files and folders from `customspeech-audiologs` Blob container directly or use REST API.

Get real-time transcription logs via REST API

[Speech to text REST API](#) fully supports BYOS-enabled Speech resources. However, because the data is now stored within the BYOS-enabled Storage account, requests like [Get Base Model Logs](#) interact with the BYOS-associated Storage account Blob storage, instead of Speech service internal resources. It allows using the same REST API based code for both "regular" and BYOS-enabled Speech resources.

For maximum security use the `sasValidityInSeconds` parameter with the value set to `0` in the requests, that return data file URLs, like [Get Base Model Logs](#) request. Here's an example request URL:

```
https
```

```
https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/endpoints/base/en-US/files/logs?sasValidityInSeconds=0
```

Such a request returns direct Storage Account URLs to data files (without SAS or other additions). For example:

```
JSON
```

```
"links": {
    "contentUrl":
"https://<BYOS_storage_account_name>.blob.core.windows.net/customspeech-
audiologs/be172190e1334399852185c0addee9d6/en-US/2023-07-06/152339_fcf52189-0d3f-
4415-becd-5f639fd7fd6b.v2.json"
}
```

URL of this format ensures that only Microsoft Entra identities (users, service principals, managed identities) with sufficient access rights (like *Storage Blob Data Reader* role) can access the data from the URL.

Warning

If `sasValidityInSeconds` parameter is omitted in [Get Base Model Logs](#) request or similar ones, then a [User delegation SAS](#) with the validity of 5 days will be generated for each data file URL returned. This SAS is signed by the system assigned managed identity of your BYOS-enabled Speech resource. Because of it, the SAS allows access to the data, even if storage account key access is disabled. See details [here](#).

Custom speech

With custom speech, you can evaluate and improve the accuracy of speech recognition for your applications and products. A custom speech model can be used for real-time speech to text, speech translation, and batch transcription. For more information, see the [custom speech overview](#).

There's nothing specific about how you use custom speech with BYOS-enabled Speech resource. The only difference is where all custom model related data, which Speech service collects and produces for you, is stored. The data is stored in the following Blob containers of BYOS-associated Storage account:

- `customspeech-models` - Location of custom speech models
- `customspeech-artifacts` - Location of all other custom speech related data

The Blob container structure is provided for your information only and subject to change without a notice.

Caution

Speech service relies on pre-defined Blob container paths and file names for custom speech module to correctly function. Don't move, rename or in any way alter the contents of `customspeech-models` container and custom speech related folders of `customspeech-artifacts` container.

Failure to do so very likely will result in hard to debug errors and may lead to the necessity of custom model retraining.

Also don't build solutions that directly use files and folders of `customspeech-artifacts` container. Use standard tools, like REST API and Speech Studio to interact with the custom speech related data. See details in [custom speech section](#).

Use of REST API with custom speech

[Speech to text REST API](#) fully supports BYOS-enabled Speech resources. However, because the data is now stored within the BYOS-enabled Storage account, requests like [Datasets_ListFiles](#) interact with the BYOS-associated Storage account Blob storage, instead of Speech service internal resources. It allows using the same REST API based code for both "regular" and BYOS-enabled Speech resources.

For maximum security use the `sasValidityInSeconds` parameter with the value set to `0` in the requests, that return data file URLs, like [Get Dataset Files](#) request. Here's an example request URL:

```
https
```

```
https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/datasets/bbbbcccc-1111-dddd-2222-eeee3333ffff/files?sasValidityInSeconds=0
```

Such a request returns direct Storage Account URLs to data files (without SAS or other additions). For example:

```
JSON
```

```
"links": {  
    "contentUrl":  
        "https://<BYOS_storage_account_name>.blob.core.windows.net/customspeech-  
        artifacts/AcousticData/8427b92a-cb50-4cda-bf04-964ea1b1781b/4a61ddac-5b1c-4c21-  
        b87d-22001b0f18ab.zip"  
}
```

URL of this format ensures that only Microsoft Entra identities (users, service principals, managed identities) with sufficient access rights (like *Storage Blob Data Reader* role) can access the data from the URL.

⚠ Warning

If `sasValidityInSeconds` parameter is omitted in [Get Dataset Files](#) request or similar ones, then a [User delegation SAS](#) with the validity of 5 days will be generated for each data file URL returned. This SAS is signed by the system assigned managed identity of your BYOS-

enabled Speech resource. Because of it, the SAS allows access to the data, even if storage account key access is disabled. See details [here](#).

Next steps

- Set up the Bring your own storage (BYOS) Speech resource
- Batch transcription overview
- How to log audio and transcriptions for speech recognition
- Custom speech overview

Speech service encryption of data at rest

08/07/2025

Speech service automatically encrypts your data when it's persisted to the cloud. Speech service encryption protects your data and to help you to meet your organizational security and compliance commitments.

About Azure AI services encryption

Data is encrypted and decrypted using [FIPS 140-2](#) compliant [256-bit AES](#) encryption. Encryption and decryption are transparent, meaning encryption and access are managed for you. Your data is secure by default and you don't need to modify your code or applications to take advantage of encryption.

About encryption key management

When you use custom speech and custom voice, Speech service might store the following data in the cloud:

- Speech trace data - only if you turn the trace on for your custom endpoint
- Uploaded training and test data

By default, your data are stored in Microsoft's storage and your Speech resource uses Microsoft-managed encryption keys. You also have an option to prepare your own storage account. Access to the store is managed by the Managed Identity, and Speech service can't directly access to your own data, such as speech trace data, customization training data and custom models.

For more information about Managed Identity, see [What are managed identities](#).

Bring your own storage (BYOS)

Bring your own storage (BYOS) is an Azure AI technology for customers, who have high requirements for data security and privacy. The core of the technology is the ability to associate an Azure Storage account, that the user owns and fully controls with the Speech resource. The Speech resource then uses this storage account for storing different artifacts related to the user data processing, instead of storing the same artifacts within the Speech service premises as it is done in the regular case. This approach allows using all set of security features of Azure Storage account, including encrypting the data with the Customer-managed keys, using Private endpoints to access the data, etc.

The Speech service doesn't currently support Customer Lockbox. However, customer data can be stored using BYOS, allowing you to achieve similar data controls to [Customer Lockbox](#).

 **Important**

Microsoft **does not** use customer data to improve its Speech models. Additionally, if endpoint logging is disabled and no customizations are used, then no customer data is stored.

See detailed information on using BYOS-enabled Speech resource in [this article](#).

Next steps

- Set up the Bring your own storage (BYOS) Speech resource
- [What are managed identities](#).

Back up and recover speech customer resources

The Speech service is [available in various regions](#). Speech resource keys are tied to a single region. When you acquire a key, you select a specific region, where your data, model and deployments reside.

Datasets for customer-created data assets, such as customized speech models and custom voice fonts, are also [available only within the service-deployed region](#). Such assets are:

Custom speech

- Training audio/text data
- Test audio/text data
- Customized speech models
- Log data

Custom voice

- Training audio/text data
- Test audio/text data
- Custom voice fonts

While some customers use our default endpoints to transcribe audio or standard voices for speech synthesis, other customers create assets for customization.

These assets are backed up regularly and automatically by the repositories themselves, so **no data loss will occur** if a region becomes unavailable. However, you must take steps to ensure service continuity if there's a region outage.

How to monitor service availability

If you use the default endpoints, you should configure your client code to monitor for errors. If errors persist, be prepared to redirect to another region where you have an AI Foundry resource for Speech.

Follow these steps to configure your client to monitor for errors:

1. Find the [list of regionally available endpoints in our documentation](#).
2. Select a primary and one or more secondary/backup regions from the list.
3. From Azure portal, create Speech service resources for each region.

- If you have set a specific quota, you might also consider setting the same quota in the backup regions. See details in [Speech service Quotas and Limits](#).

4. Each region has its own STS token service. For the primary region and any backup regions your client configuration file needs to know the:

- Regional Speech service endpoints
- [Regional key and the region code](#)

5. Configure your code to monitor for connectivity errors (typically connection timeouts and service unavailability errors). Here's sample code in C#: [GitHub: Adding Sample for showing a possible candidate for switching regions ↗](#).

- Since networks experience transient errors, for single connectivity issue occurrences, the suggestion is to retry.
- For persistence, redirect traffic to the new STS token service and Speech service endpoint. For text to speech, reference sample code: [GitHub: TTS public voice switching region ↗](#).

The recovery from regional failures for this usage type can be instantaneous and at a low cost. All that is required is the development of this functionality on the client side. The data loss that incurs assuming no backup of the audio stream will be minimal.

Custom endpoint recovery

Data assets, models or deployments in one region can't be made visible or accessible in any other region.

You should create Speech service resources in both a main and a secondary region by following the same steps as used for default endpoints.

Custom speech

Custom speech service doesn't support automatic failover. We suggest the following steps to prepare for manual or automatic failover implemented in your client code. In these steps, you replicate custom models in a secondary region. With this preparation, your client code can switch to a secondary region when the primary region fails.

1. Create your custom model in one main region (Primary).
2. Run the [Models_CopyTo](#) operation to replicate the custom model to all prepared regions (Secondary).
3. Go to Speech Studio to load the copied model and create a new endpoint in the secondary region. See how to deploy a new model in [Deploy a custom speech model](#).

- If you have set a specific quota, also consider setting the same quota in the backup regions. See details in [Speech service Quotas and Limits](#).

4. Configure your client to fail over on persistent errors as with the default endpoints usage.

Your client code can monitor availability of your deployed models in your primary region, and redirect their audio traffic to the secondary region when the primary fails. If you don't require real-time failover, you can still follow these steps to prepare for a manual failover.

Offline failover

If you don't require real-time failover you can decide to import your data, create and deploy your models in the secondary region at a later time with the understanding that these tasks take time to complete.

Failover time requirements

This section provides general guidance about timing. The times were recorded to estimate offline failover using a [representative test data set ↗](#).

- Data upload to new region: **15mins**
- Acoustic/language model creation: **6 hours (depending on the data volume)**
- Model evaluation: **30 mins**
- Endpoint deployment: **10 mins**
- Model copy API call: **10 mins**
- Client code reconfiguration and deployment: **Depending on the client system**

It's nonetheless advisable to create keys for a primary and secondary region for production models with real-time requirements.

Custom voice

Custom voice doesn't support automatic failover. Handle real-time synthesis failures with these two options.

Option 1: Fail over to public voice in the same region.

When custom voice real-time synthesis fails, fail over to a public voice (client sample code: [GitHub: custom voice failover to public voice ↗](#)).

Check the [public voices available](#). You can also change the sample code above if you would like to fail over to a different voice or in a different region.

Option 2: Fail over to custom voice on another region.

1. Create and deploy your custom voice in one main region (primary).
 2. Copy your custom voice model to another region (the secondary region) in [Speech Studio](#).
 3. Go to Speech Studio and switch to the Speech resource in the secondary region. Load the copied model and create a new endpoint.
 - Voice model deployment usually finishes **in 3 minutes**.
 - Each endpoint is subject to extra charges. [Check the pricing for model hosting here](#).
 4. Configure your client to fail over to the secondary region. See sample code in C#: [GitHub: custom voice failover to secondary region](#).
-

Last updated on 10/21/2025

What is the Speech CLI?

08/07/2025

The Speech CLI is a command-line tool for using Speech service without having to write any code. The Speech CLI requires minimal setup. You can easily use it to experiment with key features of Speech service and see how it works with your use cases. Within minutes, you can run simple test workflows, such as batch speech-recognition from a directory of files or text to speech on a collection of strings from a file. Beyond simple workflows, the Speech CLI is production-ready, and you can scale it up to run larger processes by using automated `.bat` or shell scripts.

Most features in the Speech SDK are available in the Speech CLI, and some advanced features and customizations are simplified in the Speech CLI. As you're deciding when to use the Speech CLI or the Speech SDK, consider the following guidance.

Use the Speech CLI when:

- You want to experiment with Speech service features with minimal setup and without having to write code.
- You have relatively simple requirements for a production application that uses Speech service.

Use the Speech SDK when:

- You want to integrate Speech service functionality within a specific language or platform (for example, C#, Python, or C++).
- You have complex requirements that might require advanced service requests.
- You're developing custom behavior, including response streaming.

Core features

- **Speech recognition:** Convert speech to text either from audio files or directly from a microphone, or transcribe a recorded conversation.
- **Speech synthesis:** Convert text to speech either by using input from text files or by inputting directly from the command line. Customize speech output characteristics by using [Speech Synthesis Markup Language \(SSML\) configurations](#).
- **Speech translation:** Translate audio in a source language to text or audio in a target language.

- **Run on Azure compute resources:** Send Speech CLI commands to run on an Azure remote compute resource by using `spx webjob`.

Get started

To get started with the Speech CLI, see the [quickstart](#). This article shows you how to run some basic commands. It also gives you slightly more advanced commands for running batch operations for speech to text and text to speech. After you read the basics article, you should understand the syntax enough to start writing some custom commands or automate simple Speech service operations.

Next steps

- [Get started with the Azure AI Speech CLI](#)
- [Speech CLI configuration options](#)
- [Speech CLI batch operations](#)

Quickstart: Get started with the Azure AI Speech CLI

08/07/2025

In this article, you learn how to use the Azure AI Speech CLI (also called SPX) to access Speech services such as speech to text, text to speech, and speech translation, without having to write any code. The Speech CLI is production ready, and you can use it to automate simple workflows in the Speech service by using `.bat` or shell scripts.

This article assumes that you have working knowledge of the Command Prompt window, terminal, or PowerShell.

! Note

In PowerShell, the `stop-parsing token` (--) should follow `spx`. For example, run `spx --% config @region` to view the current region config value.

Download and install

Windows

Follow these steps to install the Speech CLI on Windows:

1. Install the [Microsoft Visual C++ Redistributable for Visual Studio](#) for your platform.
Installing it for the first time might require a restart.
2. Install [.NET 8](#).
3. Install the Speech CLI via the .NET CLI by entering this command:

.NET CLI

```
dotnet tool install --global Microsoft.CognitiveServices.Speech.CLI
```

To update the Speech CLI, enter this command:

.NET CLI

```
dotnet tool update --global Microsoft.CognitiveServices.Speech.CLI
```

Enter `spx` or `spx help` to see help for the Speech CLI.

Font limitations

On Windows, the Speech CLI can show only fonts that are available to the command prompt on the local computer. [Windows Terminal](#) supports all fonts that the Speech CLI produces interactively.

If you output to a file, a text editor like Notepad or a web browser like Microsoft Edge can also show all fonts.

Create a resource configuration

Terminal

To get started, you need an API key and region identifier (for example, `eastus`, `westus`). Create an AI Foundry resource for Speech on the [Azure portal](#). For more information, see [Create an AI Foundry resource](#).

To configure your resource key and region identifier, run the following commands:

Console

```
spx config @key --set SPEECH-KEY  
spx config @region --set SPEECH-REGION
```

The key and region are stored for future Speech CLI commands. To view the current configuration, run the following commands:

Console

```
spx config @key  
spx config @region
```

As needed, include the `clear` option to remove either stored value:

Console

```
spx config @key --clear  
spx config @region --clear
```

Basic usage

ⓘ Important

When you use the Speech CLI in a container, include the `--host` option. You must also specify `--key none` to ensure that the CLI doesn't try to use a Speech key for authentication. For example, run `spx recognize --key none --host wss://localhost:5000/ --file myaudio.wav` to recognize speech from an audio file in a [speech to text container](#).

This section shows a few basic SPX commands that are often useful for first-time testing and experimentation. Run the following command to view the in-tool help:

```
Console
```

```
spx
```

You can search help topics by keyword. For example, to see a list of Speech CLI usage examples, run the following command:

```
Console
```

```
spx help find --topics "examples"
```

To see options for the `recognize` command, run the following command:

```
Console
```

```
spx help recognize
```

More help commands are listed in the console output. You can enter these commands to get detailed help about subcommands.

Speech to text (speech recognition)

💡 Tip

If you get stuck or want to learn more about the Speech CLI recognition options, you can run `spx help recognize`.

Recognize speech from a microphone

1. Run the following command to start speech recognition from a microphone:

```
Console
```

```
spx recognize --microphone --source en-US
```

2. Speak into the microphone, and you see transcription of your words into text in real-time.

The Speech CLI stops after a period of silence, 30 seconds, or when you select **Ctrl+C**.

```
Output
```

```
Connection CONNECTED...
```

```
RECOGNIZED: I'm excited to try speech to text.
```

ⓘ Note

You can't use your computer's microphone when you run the Speech CLI within a Docker container. However, you can read from and save audio files in your local mounted directory.

Recognize speech from a file

To recognize speech from an audio file, use `--file` instead of `--microphone`. For compressed audio files such as MP4, install GStreamer and use `--format`. For more information, see [How to use compressed input audio](#).

```
Terminal
```

```
Console
```

```
spx recognize --file YourAudioFile.wav  
spx recognize --file YourAudioFile.mp4 --format any
```

Phrase lists

To improve recognition accuracy of specific words or utterances, use a [phrase list](#). You include a phrase list in-line or with a text file along with the `recognize` command:

Terminal

Console

```
spx recognize --microphone --phrases "Contoso; Jessie; Rehaan;"  
spx recognize --microphone --phrases @phrases.txt
```

Language support

To change the speech recognition language, replace `en-US` with another [supported language](#). For example, use `es-ES` for Spanish (Spain). If you don't specify a language, the default is `en-US`.

Console

```
spx recognize --microphone --source es-ES
```

Continuous recognition

For continuous recognition of audio longer than 30 seconds, append `--continuous`:

Console

```
spx recognize --microphone --source es-ES --continuous
```

Text to speech (speech synthesis)

Tip

If you get stuck or want to learn more about the Speech CLI recognition options, you can run `spx help synthesize`.

The following command takes text as input and then outputs the synthesized speech to the current active output device (for example, your computer speakers).

Console

```
spx synthesize --text "Testing synthesis using the Speech CLI" --speakers
```

You can also save the synthesized output to a file. In this example, let's create a file named *my-sample.wav* in the directory where you're running the command.

Console

```
spx synthesize --text "Enjoy using the Speech CLI." --audio output my-sample.wav
```

These examples presume that you're testing in English. However, Speech service supports speech synthesis in many languages. You can pull down a full list of voices either by running the following command or by visiting the [language support page](#).

Console

```
spx synthesize --voices
```

Here's a command for using one of the voices you discovered.

Console

```
spx synthesize --text "Bienvenue chez moi." --voice fr-FR-AlainNeural --speakers
```

Speech to text translation



If you get stuck or want to learn more about the Speech CLI translation options, you can run `spx help translate`.

Translate speech from a microphone

1. Run the following command to start speech translation from a microphone:

Console

```
spx translate --source en-US --target it --microphone
```

2. Speak into the microphone, and you see the transcription of your translated speech in real-time. The Speech CLI stops after a period of silence, 30 seconds, or when you select **Ctrl+C**.

Output

```
Connection CONNECTED...
TRANSLATING into 'it': Sono (from 'I'm')
TRANSLATING into 'it': Sono entusiasta (from 'I'm excited to')
TRANSLATING into 'it': Sono entusiasta di provare la parola (from 'I'm
excited to try speech')
TRANSLATED into 'it': Sono entusiasta di provare la traduzione vocale. (from
'I'm excited to try speech translation.')
```

ⓘ Note

You can't use your computer's microphone when you run the Speech CLI within a Docker container. However, you can read from and save audio files in your local mounted directory.

Translate speech from a file

To translate speech from an audio file, use `--file` instead of `--microphone`. For compressed audio files such as MP4, install GStreamer and use `--format`. For more information, see [How to use compressed input audio](#).

Terminal

Console

```
spx translate --source en-US --target it --file YourAudioFile.wav
spx translate --source en-US --target it --file YourAudioFile.mp4 --format any
```

Phrase lists

To improve recognition accuracy of specific words or utterances, use a [phrase list](#). You include a phrase list in-line or with a text file along with the `translate` command:

Terminal

Console

```
spx translate --source en-US --target it --microphone --phrases
"Contoso;Jessie;Rehaan;"
spx translate --source en-US --target it --microphone --phrases @phrases.txt
```

Language support

To change the speech recognition language, replace `en-US` with another [supported language](#). Specify the full locale with a dash (-) separator. For example, `es-ES` for Spanish (Spain). The default language is `en-US` if you don't specify a language.

```
Console
```

```
spx translate --microphone --source es-ES
```

To change the translation target language, replace `it` with another [supported language](#). With few exceptions you only specify the language code that precedes the locale dash (-) separator. For example, use `es` for Spanish (Spain) instead of `es-ES`. The default language is `en` if you don't specify a language.

```
Console
```

```
spx translate --microphone --target es
```

Multiple target languages

When you're translating into multiple languages, separate the language codes with a semicolon (;).

```
Console
```

```
spx translate --microphone --source en-US --target 'ru-RU;fr-FR;es-ES'
```

Save translation output

If you want to save the output of your translation, use the `--output` flag. In this example, you also read from a file.

```
Console
```

```
spx translate --file /some/file/path/input.wav --source en-US --target ru-RU --  
output file /some/file/path/russian_translation.txt
```

Continuous translation

For continuous translation of audio longer than 30 seconds, append `--continuous`:

Console

```
spx translate --source en-US --target it --microphone --continuous
```

Next steps

- [Install GStreamer to use the Speech CLI with MP3 and other formats](#)
- [Configuration options for the Speech CLI](#)
- [Batch operations with the Speech CLI](#)

Configure the Speech CLI datastore

08/07/2025

The [Speech CLI](#) can rely on settings in configuration files, which you can refer to using a `@` symbol. The Speech CLI saves a new setting in a new `./spx/data` subdirectory that is created in the current working directory for the Speech CLI. The Speech CLI first looks for a configuration value in your current working directory, then in the datastore at `./spx/data`, and then in other datastores, including a final read-only datastore in the `spx` binary.

In the [Speech CLI quickstart](#), you used the datastore to save your `@key` and `@region` values, so you didn't need to specify them with each `spx` command. Keep in mind, that you can use configuration files to store your own configuration settings, or even use them to pass URLs or other dynamic content generated at runtime.

For more details about datastore files, including use of default configuration files (`@spx.default`, `@default.config`, and `@*.default.config` for command-specific default settings), enter this command:

```
Console  
spx help advanced setup
```

nodefaults

The following example clears the `@my.defaults` configuration file, adds key-value pairs for `key` and `region` in the file, and uses the configuration in a call to `spx recognize`.

```
Console  
spx config @my.defaults --clear  
spx config @my.defaults --add key 000072626F6E20697320636F6F6C0000  
spx config @my.defaults --add region westus  
  
spx config @my.defaults  
  
spx recognize --nodefaults @my.defaults --file hello.wav
```

Dynamic configuration

You can also write dynamic content to a configuration file using the `--output` option.

For example, the following command creates a custom speech model and stores the URL of the new model in a configuration file. The next command waits until the model at that URL is ready for use before returning.

```
Console  
  
spx csr model create --name "Example 4" --datasets @my.datasets.txt --output url  
@my.model.txt  
spx csr model status --model @my.model.txt --wait
```

The following example writes two URLs to the `@my.datasets.txt` configuration file. In this scenario, `--output` can include an optional `add` keyword to create a configuration file or append to the existing one.

```
Console  
  
spx csr dataset create --name "LM" --kind Language --content  
https://crbn.us/data.txt --output url @my.datasets.txt  
spx csr dataset create --name "AM" --kind Acoustic --content  
https://crbn.us/audio.zip --output add url @my.datasets.txt  
  
spx config @my.datasets.txt
```

SPX config add

For readability, flexibility, and convenience, you can use a preset configuration with select output options.

For example, you might have the following requirements for [captioning](#):

- Recognize from the input file `caption.this.mp4`.
- Output WebVTT and SRT captions to the files `caption.vtt` and `caption.srt` respectively.
- Output the `offset`, `duration`, `resultid`, and `text` of each recognizing event to the file `each.result.tsv`.

You can create a preset configuration named `@caption.defaults` as shown here:

```
Console  
  
spx config @caption.defaults --clear  
spx config @caption.defaults --add output.each.recognizing.result.offset=true  
spx config @caption.defaults --add output.each.recognizing.result.duration=true  
spx config @caption.defaults --add output.each.recognizing.result.resultid=true  
spx config @caption.defaults --add output.each.recognizing.result.text=true  
spx config @caption.defaults --add output.each.file.name=each.result.tsv
```

```
spx config @caption.defaults --add output.srt.file.name=caption.srt  
spx config @caption.defaults --add output.vtt.file.name=caption.vtt
```

The settings are saved to the current directory in a file named `caption.defaults`. Here are the file contents:

```
output.each.recognizing.result.offset=true  
output.each.recognizing.result.duration=true  
output.each.recognizing.result.resultid=true  
output.each.recognizing.result.text=true  
output.all.file.name=output.result.tsv  
output.each.file.name=each.result.tsv  
output.srt.file.name=caption.srt  
output.vtt.file.name=caption.vtt
```

Then, to generate [captions](#), you can run this command that imports settings from the `@caption.defaults` preset configuration:

Console

```
spx recognize --file caption.this.mp4 --format any --output vtt --output srt  
@caption.defaults
```

Using the preset configuration as shown previously is similar to running the following command:

Console

```
spx recognize --file caption.this.mp4 --format any --output vtt file caption.vtt -  
--output srt file caption.srt --output each file each.result.tsv --output all file  
output.result.tsv --output each recognizer recognizing result offset --output each  
recognizer recognizing duration --output each recognizer recognizing result  
resultid --output each recognizer recognizing text
```

Next steps

- [Batch operations with the Speech CLI](#)

Configure the Speech CLI output options

08/07/2025

The [Speech CLI](#) output can be written to standard output or specified files.

For contextual help in the Speech CLI, you can run any of the following commands:

Console

```
spx help recognize output examples  
spx help synthesize output examples  
spx help translate output examples  
spx help intent output examples
```

Standard output

If the file argument is a hyphen (-), the results are written to standard output as shown in the following example.

Console

```
spx recognize --file caption.this.mp4 --format any --output vtt file - --output  
srt file - --output each file - @output.each.detailed --property  
SpeechServiceResponse_StablePartialResultThreshold=0 --profanity masked
```

Default file output

If you omit the `file` option, output is written to default files in the current directory.

For example, run the following command to write WebVTT and SRT [captions](#) to their own default files:

Console

```
spx recognize --file caption.this.mp4 --format any --output vtt --output srt --  
output each text --output all duration
```

The default file names are as follows, where the `<EPOCH_TIME>` is replaced at run time.

- The default SRT file name includes the input file name and the local operating system epoch time: `output.caption.this.<EPOCH_TIME>.srt`

- The default Web VTT file name includes the input file name and the local operating system epoch time: `output.caption.this.<EPOCH_TIME>.vtt`
- The default `output each` file name, `each.<EPOCH_TIME>.tsv`, includes the local operating system epoch time. This file isn't created by default, unless you specify the `--output each` option.
- The default `output all` file name, `output.<EPOCH_TIME>.tsv`, includes the local operating system epoch time. This file is created by default.

Output to specific files

For output to files that you specify instead of the [default files](#), set the `file` option to the file name.

For example, to output both WebVTT and SRT [captions](#) to files that you specify, run the following command:

```
Console
spx recognize --file caption.this.mp4 --format any --output vtt file caption.vtt --
--output srt file caption.srt --output each text --output each file each.result.tsv
--output all file output.result.tsv
```

The preceding command also outputs the `each` and `all` results to the specified files.

Output to multiple files

For translations with `spx translate`, separate files are created for the source language (such as `--source en-US`) and each target language (such as `--target "de;fr;zh-Hant"`).

For example, to output translated SRT and WebVTT captions, run the following command:

```
Console
spx translate --source en-US --target "de;fr;zh-Hant" --file caption.this.mp4 --
format any --output vtt file caption.vtt --output srt file caption.srt
```

Captions should then be written to the following files: `caption.srt`, `caption.vtt`, `caption.de.srt`, `caption.de.vtt`, `caption.fr.srt`, `caption.fr.vtt`, `caption.zh-Hant.srt`, and `caption.zh-Hant.vtt`.

Suppress header

You can suppress the header line in the output file by setting the `has header false` option:

```
spx recognize --nodefaults @my.defaults --file audio.wav --output recognized_text  
--output file has header false
```

See [Configure the Speech CLI datastore](#) for more information about `--nodefaults`.

Next steps

- [Captioning quickstart](#)

Run batch operations with the Speech CLI

08/07/2025

Common tasks when using the Speech service, are batch operations. In this article, you learn how to do batch speech to text (speech recognition), batch text to speech (speech synthesis) with the Speech CLI. Specifically, you learn how to:

- Run batch speech recognition on a directory of audio files
- Run batch speech synthesis by iterating over a `.tsv` file

Batch speech to text (speech recognition)

The Speech service is often used to recognize speech from audio files. In this example, you learn how to iterate over a directory using the Speech CLI to capture the recognition output for each `.wav` file. The `--files` flag is used to point at the directory where audio files are stored, and the wildcard `*.wav` is used to tell the Speech CLI to run recognition on every file with the extension `.wav`. The output for each recognition file is written as a tab separated value in `speech_output.tsv`.

! Note

The `--threads` argument can be also used in the next section for `spx synthesize` commands, and the available threads will depend on the CPU and its current load percentage.

Console

```
spx recognize --files C:\your_wav_file_dir\*.wav --output file  
C:\output_dir\speech_output.tsv --threads 10
```

Here's an example of the output file structure.

Output

<code>audio.input.id</code>	<code>recognizer.session.started.sessionid</code>
<code>recognizer.recognized.result.text</code>	
<code>sample_1</code>	<code>07baa2f8d9fd4fbcb9faea451ce05475</code>
	A sample wave file.
<code>sample_2</code>	<code>8f9b378f6d0b42f99522f1173492f013</code>
	Sample text synthesized.

Batch text to speech (speech synthesis)

The easiest way to run batch text to speech is to create a new `.tsv` (tab-separated-value) file, and use the `--foreach` command in the Speech CLI. You can create a `.tsv` file using your favorite text editor, for this example, let's call it `text_synthesis.tsv`:

ⓘ Important

When you copy the contents of this text file, make sure that your file has a **tab** not spaces between the file location and the text. Sometimes, when copying the contents from this example, tabs are converted to spaces causing the `spx` command to fail when run.

Input

```
audio.output      text
C:\batch_wav_output\wav_1.wav    Sample text to synthesize.
C:\batch_wav_output\wav_2.wav    Using the Speech CLI to run batch-synthesis.
C:\batch_wav_output\wav_3.wav    Some more text to test capabilities.
```

Next, you run a command to point to `text_synthesis.tsv`, perform synthesis on each `text` field, and write the result to the corresponding `audio.output` path as a `.wav` file.

Console

```
spx synthesize --foreach in @C:\your\path\to\text_synthesis.tsv
```

This command is the equivalent of running `spx synthesize --text "Sample text to synthesize" --audio output C:\batch_wav_output\wav_1.wav` for each record in the `.tsv` file.

A couple things to note:

- The column headers, `audio.output` and `text`, correspond to the command-line arguments `--audio output` and `--text`, respectively. Multi-part command-line arguments like `--audio output` should be formatted in the file with no spaces, no leading dashes, and periods separating strings, for example, `audio.output`. Any other existing command-line arguments can be added to the file as more columns using this pattern.
- When the file is formatted in this way, no other arguments are required to be passed to `-foreach`.
- Ensure to separate each value in the `.tsv` with a **tab**.

However, if you have a `.tsv` file like the following example, with column headers that **do not match** command-line arguments:

Input

```
wav_path    str_text
C:\batch_wav_output\wav_1.wav  Sample text to synthesize.
C:\batch_wav_output\wav_2.wav  Using the Speech CLI to run batch-synthesis.
C:\batch_wav_output\wav_3.wav  Some more text to test capabilities.
```

You can override these field names to the correct arguments using the following syntax in the `--foreach` call. This command makes the same call as before.

Console

```
spx synthesize --foreach audio.output;text in @C:\your\path\to\text_synthesis.tsv
```

Next steps

- [Speech CLI overview](#)
- [Speech CLI quickstart](#)

What is the Speech SDK?

08/07/2025

The Speech SDK (software development kit) exposes many of the [Speech service capabilities](#), so you can develop speech-enabled applications. The Speech SDK is available [in many programming languages](#) and across platforms. The Speech SDK is ideal for both real-time and non-real-time scenarios, by using local devices, files, Azure Blob Storage, and input and output streams.

In some cases, you can't or shouldn't use the [Speech SDK](#). In those cases, you can use REST APIs to access the Speech service. For example, use the [Speech to text REST API](#) for [batch transcription](#) and [custom speech model management](#).

Supported languages

The Speech SDK supports the following languages and platforms:

[] [Expand table](#)

Programming language	Reference	Platform support
C# ¹	.NET	Windows, Linux, macOS, Mono, UWP, Unity
C++ ²	C++	Windows, Linux, macOS
Go	Go ↗	Linux
Java	Java	Android, Windows, Linux, macOS
JavaScript	JavaScript	Browser, Node.js
Objective-C	Objective-C	iOS, macOS
Python	Python	Windows, Linux, macOS
Swift	Objective-C ³	iOS, macOS

1 C# code samples are available in the documentation. The Speech SDK for C# is based on .NET Standard 2.0, so it supports many platforms and programming languages. For more information, see [.NET implementation support](#).

2 C isn't a supported programming language for the Speech SDK.

3 The Speech SDK for Swift shares client libraries and reference documentation with the Speech SDK for Objective-C.

(i) **Important**

By downloading any of the Azure AI Speech SDKs, you acknowledge its license. For more information, see:

- [Microsoft software license terms for the Speech SDK ↗](#)
- [Third-party software notices ↗](#)

Speech SDK demo

The following video shows how to install the [Speech SDK for C#](#) and write a .NET console application for speech to text.

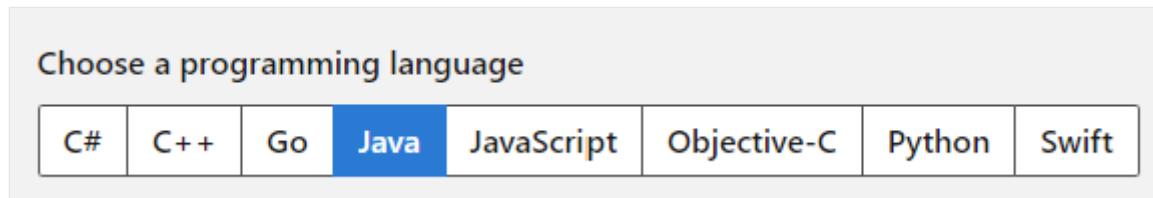
<https://learn-video.azurefd.net/vod/player?id=c20d3b0c-e96a-4154-9299-155e27db7117&locale=en-us&embedUrl=%2Fazure%2Fai-services%2Fspeech-service%2Fspeech-sdk> ↗

Code samples

Speech SDK code samples are available in the documentation and GitHub.

Docs samples

At the top of documentation pages that contain samples, options to select include C#, C++, Go, Java, JavaScript, Objective-C, Python, or Swift.



If a sample isn't available in your preferred programming language, you can select another programming language to get started and learn about the concepts, or see the reference and samples linked from the beginning of the article.

GitHub samples

In depth samples are available in the [Azure-Samples/cognitive-services-speech-sdk](#) ↗ repository on GitHub. There are samples for C# (including UWP and Unity), C++, Java, JavaScript (including Browser and Node.js), Objective-C, Python, and Swift. Code samples for Go are available in the [Microsoft/cognitive-services-speech-sdk-go](#) ↗ repository on GitHub.

Help options

The [Microsoft Q&A](#) and [Stack Overflow ↗](#) forums are available for the developer community to ask and answer questions about Azure Cognitive Speech and other services. Microsoft monitors the forums and replies to questions that the community hasn't yet answered. To make sure that we see your question, tag it with 'azure-speech'.

You can suggest an idea or report a bug by creating an issue on GitHub:

- [Azure-Samples/cognitive-services-speech-sdk ↗](#)
- [Microsoft/cognitive-services-speech-sdk-go ↗](#)
- [Microsoft/cognitive-services-speech-sdk-js ↗](#)

See also [Azure AI services support and help options](#) to get support, stay up-to-date, give feedback, and report bugs for Azure AI services.

Next steps

- [Install the SDK](#)
- [Try the speech to text quickstart](#)

Quickstart: Install the Speech SDK

07/25/2025

[Reference documentation](#) | [Package \(NuGet\)](#) ↗ | [Additional samples on GitHub](#) ↗

In this quickstart, you install the [Speech SDK](#) for C#.

Platform requirements

The Speech SDK for C# is compatible with Windows, Linux, and macOS.

Windows

On Windows, you must use the 64-bit target architecture. Windows 10 or later is required.

Install the [Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017, 2019, and 2022](#) for your platform. Installing this package for the first time might require a restart.

Install the Speech SDK for C#

The Speech SDK for C# is available as a NuGet package and implements .NET Standard 2.0. For more information, see [Microsoft.CognitiveServices.Speech](#) ↗.

Terminal

The Speech SDK for C# can be installed from the .NET CLI by using the following `dotnet add` command:

.NET CLI

```
dotnet add package Microsoft.CognitiveServices.Speech
```

Code samples

Code samples are available in the [Azure-Samples/cognitive-services-speech-sdk](#) ↗ repository on GitHub. There are samples for C# (including Universal Windows Platform (UWP) and Unity), C++, Java, JavaScript (including Browser and Node.js), Objective-C, Python, and Swift. Code

samples for Go are available in the [Microsoft/cognitive-services-speech-sdk-go](#) repository on GitHub.

Related content

- [Speech to text quickstart](#)
- [Text to speech quickstart](#)
- [Speech translation quickstart](#)

Enable logging in the Speech SDK

08/07/2025

Logging to file is an optional feature for the Speech SDK. During development, logging provides additional information and diagnostics from the Speech SDK's core components. It can be enabled by setting the `Speech_LogFilename` property on a speech configuration object to the location and name of the log file. Logging is handled by a static class in Speech SDK's native library. You can turn on logging for any Speech SDK recognizer or synthesizer instance. All instances in the same process write log entries to the same log file.

Sample

The log file name is specified on a configuration object. Taking the `SpeechConfig` as an example and assuming that you created an instance called `speechConfig`:

C#

```
speechConfig SetProperty(PropertyId.Speech_LogFilename, "LogfilePathAndName");
```

Java

```
speechConfig.setProperty(PropertyId.Speech_LogFilename, "LogfilePathAndName");
```

C++

```
speechConfig->SetProperty(PropertyId::Speech_LogFilename, "LogfilePathAndName");
```

Python

```
speech_config.set_property(speechsdk.PropertyId.Speech_LogFilename,  
"LogfilePathAndName")
```

Objective-C

```
[speechConfig SetPropertyTo:@"LogfilePathAndName" byId:SPXSpeechLogFilename];
```

Go

```
import ("github.com/Microsoft/cognitive-services-speech-sdk-go/common")
```

```
speechConfig SetProperty(common.SpeechLogFilename, "LogfilePathAndName")
```

You can create a recognizer from the configuration object. This enables logging for all recognizers.

(!) Note

If you create a `SpeechSynthesizer` from the configuration object, it will not enable logging.

If logging is enabled though, you will also receive diagnostics from the `SpeechSynthesizer`.

JavaScript is an exception where the logging is enabled via SDK diagnostics as shown in the following code snippet:

JavaScript

```
sdk.Diagnostics.SetLoggingLevel(sdk.LogLevel.Debug);  
sdk.Diagnostics.SetLogOutputPath("LogfilePathAndName");
```

Create a log file on different platforms

For Windows or Linux, the log file can be in any path the user has write permission for. Write permissions to file system locations in other operating systems might be limited or restricted by default.

Universal Windows Platform (UWP)

UWP applications need to place log files in one of the application data locations (local, roaming, or temporary). A log file can be created in the local application folder:

C#

```
StorageFolder storageFolder = ApplicationData.Current.LocalFolder;  
StorageFile logFile = await storageFolder.CreateFileAsync("logfile.txt",  
CreationCollisionOption.ReplaceExisting);  
speechConfig SetProperty(PropertyId.Speech_LogFilename, logFile.Path);
```

Within a Unity UWP application, a log file can be created using the application persistent data path folder as follows:

C#

```
#if ENABLE_WINMD_SUPPORT
    string logFile = Application.persistentDataPath + "/logFile.txt";
    speechConfig SetProperty(PropertyId.Speech_LogFilename, logFile);
#endif
```

For more about file access permissions in UWP applications, see [File access permissions](#).

Android

You can save a log file to either internal storage, external storage, or the cache directory. Files created in the internal storage or the cache directory are private to the application. It's preferable to create a log file in external storage.

Java

```
File dir = context.getExternalFilesDir(null);
File logFile = new File(dir, "logfile.txt");
speechConfig.setProperty(PropertyId.Speech_LogFilename,
    logFile.getAbsolutePath());
```

The code above will save a log file to the external storage in the root of an application-specific directory. A user can access the file with the file manager (usually in `Android/data/ApplicationName/logfile.txt`). The file is deleted when the application is uninstalled.

You also need to request `WRITE_EXTERNAL_STORAGE` permission in the manifest file:

XML

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="...">
    ...
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

Within a Unity Android application, the log file can be created using the application persistent data path folder as follows:

C#

```
string logFile = Application.persistentDataPath + "/logFile.txt";
speechConfig SetProperty(PropertyId.Speech_LogFilename, logFile);
```

In addition, you need to also set write permission in your Unity Player settings for Android to "External (SDCard)". The log is written to a directory that you can get by using a tool such as AndroidStudio Device File Explorer. The exact directory path can vary between Android devices. The location is typically the `sdcard/Android/data/your-app-pagename/files` directory.

More about data and file storage for Android applications is available [here ↗](#).

iOS

Only directories inside the application sandbox are accessible. Files can be created in the documents, library, and temp directories. Files in the documents directory can be made available to a user.

If you're using Objective-C on iOS, use the following code snippet to create a log file in the application document directory:

Objective-C

```
NSString *filePath = [
    [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)
firstObject]
    stringByAppendingPathComponent:@"logfile.txt"];
[speechConfig setPropertyTo:filePath byId:SPXSpeechLogFilename];
```

To access a created file, add the below properties to the `Info.plist` property list of the application:

XML

```
<key>UIFileSharingEnabled</key>
<true/>
<key>LSSupportsOpeningDocumentsInPlace</key>
<true/>
```

If you're using Swift on iOS, use the following code snippet to enable logs:

Swift

```
let documentsDirectoryPathString =
NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask,
true).first!
let documentsDirectoryPath = NSURL(string: documentsDirectoryPathString)!
let logFilePath = documentsDirectoryPath.appendingPathComponent("swift.log")
self.speechConfig!.setPropertyTo(logFilePath!.absoluteString, by:
SPXPropertyId.speechLogFilename)
```

More about iOS File System is available [here](#).

Logging with multiple recognizers

Although a log file output path is specified as a configuration property into a `SpeechRecognizer` or other SDK object, SDK logging is a singleton, process-wide facility with no concept of individual instances. You can think of this as the `SpeechRecognizer` constructor (or similar) implicitly calling a static and internal "Configure Global Logging" routine with the property data available in the corresponding `SpeechConfig`.

This means that you can't, as an example, configure six parallel recognizers to output simultaneously to six separate files. Instead, the latest recognizer created will configure the global logging instance to output to the file specified in its configuration properties and all SDK logging is emitted to that file.

This also means that the lifetime of the object that configured logging isn't tied to the duration of logging. Logging won't stop in response to the release of an SDK object and will continue as long as no new logging configuration is provided. Once started, process-wide logging can be stopped by setting the log file path to an empty string when creating a new object.

To reduce potential confusion when configuring logging for multiple instances, it might be useful to abstract control of logging from objects doing real work. An example pair of helper routines:

C++

```
void EnableSpeechSdkLogging(const char* relativePath)
{
    auto configForLogging = SpeechConfig::FromSubscription("unused_key",
"unused_region");
    configForLogging->SetProperty(PropertyId::Speech_LogFilename, relativePath);
    auto emptyAudioConfig =
AudioConfig::FromStreamInput(AudioInputStream::CreatePushStream());
    auto temporaryRecognizer = SpeechRecognizer::FromConfig(configForLogging,
emptyAudioConfig);
}

void DisableSpeechSdkLogging()
{
    EnableSpeechSdkLogging("");
}
```

Logging with file logger, Memory logger and event logger

With Speech SDK version 1.43.0, the logging mechanism is extended with more types of loggers: `File logger`, `Memory logger` and `Event logger`.

- `File logger` is the simplest logging solution and suitable for diagnosing most on-device issues when running Speech SDK.
- `Memory logger` is a logging solution that stores log messages in memory. It's suitable for diagnosing issues that occur in a short period of time. For example, if you're running a Speech Recognizer, you might want to dump the memory logger after getting an event indicating recognition was canceled due to some error. The size of the memory buffer is fixed at 2MB and can't be changed. This is a "ring" buffer, that is, new log strings written replace the oldest ones in the buffer.
- `Event logger` is a logging solution that sends log messages to the event handler which is provided by the developer. It's suitable for diagnosing issues when certain new log strings are as soon as available and need for further processing. For example, integrating Speech SDK logs with your existing logging collection system.

The file logger, memory logger, and event logger all have filter mechanism by only logging certain string messages. Also these loggers are process wide constructs. That means that if (for example) you have multiple speech recognizer objects running in parallel, there's one log file containing interleaved logs lines from all recognizers. You can't get a separate file logger for each recognizer. Similarly, there's one memory buffer containing interleaved logs from all recognizers and you can only register one event handler as callback function to receive interleaved logs from all recognizers. You can't get a separate memory logger for each recognizer and you can't register an event handler for each recognizer. However, `File logger`, `memory logger` and `event logger` can coexist in the same process or in the same recognizer.

Samples

C#

```
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Diagnostics.Logging;

class Program
{
    public static void FileLoggerWithoutFilters()
    {
        FileLogger.Start("LogfilePathAndName");

        // Other Speech SDK calls

        FileLogger.Stop();
    }
}
```

```
public static void FileLoggerWithFilters()
{
    string[] filters = { "YourFirstString", "YourSecondString" };
    FileLogger.SetFilters(filters);
    FileLogger.Start("LogfilePathAndName");

    // Other Speech SDK calls

    FileLogger.Stop();
    FileLogger.SetFilters();
}

public static void MemoryLoggerWithoutFilters()
{
    MemoryLogger.Start();

    // Other Speech SDK calls

    // At any time (whether logging is stopped) you can dump the traces in
    memory to a file
    MemoryLogger.Dump("LogfilePathAndName");

    // Or dump to any object that is derived from System.IO.TextWriter. For
    example, System.Console.Out
    MemoryLogger.Dump(System.Console.Out);

    // Or dump to a vector of strings
    List<string> messages = MemoryLogger.Dump().ToList<string>();

    MemoryLogger.Stop();
}

// These variables and method are used by the EvenLogger sample below.
private static readonly object lockObject = new object();
private static List<string> eventMessages = new List<string>();
private static void OnMessageEvent(object sender, string message)
{
    lock (lockObject)
    {
        // Store the message for later processing. Better not processing it in
        the event thread
        eventMessages.Add(message);
    }
}

public static void EventLoggerWithoutFilters()
{
    // Subscribe an event that will get invoked by Speech SDK on every new log
    message
    EventLogger.OnMessage += OnMessageEvent;

    // Other Speech SDK calls

    // Unsubscribe to stop getting events
```

```
        EventLogger.OnMessage -= OnMessageEvent;  
    }  
}
```

Next steps

[Explore our samples on GitHub](#)

How to log audio and transcriptions for speech recognition

You can enable logging for both audio input and recognized speech when using [speech to text](#) or [speech translation](#). For speech translation, only the audio and transcription of the original audio are logged. The translations aren't logged. This article describes how to enable, access, and delete the audio and transcription logs.

Audio and transcription logs can be used as input for [custom speech](#) model training. You might have other use cases.

Warning

Don't depend on audio and transcription logs when the exact record of input audio is required. In the periods of peak load, the service prioritizes hardware resources for transcription tasks. This may result in minor parts of the audio not being logged. Such occasions are rare, but nevertheless possible.

Logging is done asynchronously for both base and custom model endpoints. The Speech service stores audio and transcription logs in its internal storage and not written locally. The logs are retained for 30 days. After this period, the logs are automatically deleted. However you can [delete](#) specific logs or a range of available logs at any time.

You can also store audio and transcription logs within an Azure Storage account you own and control instead of Speech service premises using [Bring your own storage \(BYOS\)](#) technology. See details on how to use BYOS-enabled Speech resource in [this article](#).

Enable audio and transcription logging

Logging is disabled by default. Logging can be enabled [per recognition session](#) or [per custom model endpoint](#).

Enable logging for a single recognition session

You can enable logging for a single recognition session, whether using the default base model or [custom model](#) endpoint.

Warning

For custom model endpoints, the logging setting of your deployed endpoint is prioritized over your session-level setting (SDK or REST API). If logging is enabled for the custom model endpoint, the session-level setting (whether it's set to true or false) is ignored. If logging isn't enabled for the custom model endpoint, the session-level setting determines whether logging is active.

Enable logging for speech to text with the Speech SDK

To enable audio and transcription logging with the Speech SDK, you execute the method `EnableAudioLogging()` of the [SpeechConfig](#) class instance.

C#

```
speechConfig.EnableAudioLogging();
```

To check whether logging is enabled, get the value of the `SpeechServiceConnection_EnableAudioLogging` [property](#):

C#

```
string isAudioLoggingEnabled =
speechConfig.GetProperty(PropertyId.SpeechServiceConnection_EnableAudioLogging);
```

Each [SpeechRecognizer](#) that uses this `speechConfig` has audio and transcription logging enabled.

Enable logging for speech translation with the Speech SDK

For speech translation, only the audio and transcription of the original audio are logged. The translations aren't logged.

To enable audio and transcription logging with the Speech SDK, you execute the method `EnableAudioLogging()` of the [SpeechTranslationConfig](#) class instance.

C#

```
speechTranslationConfig.EnableAudioLogging();
```

To check whether logging is enabled, get the value of the `SpeechServiceConnection_EnableAudioLogging` [property](#):

C#

```
string isAudioLoggingEnabled =  
speechTranslationConfig.GetProperty(PropertyId.SpeechServiceConnection_EnableAudioLo  
gging);
```

Each [TranslationRecognizer](#) that uses this `speechTranslationConfig` has audio and transcription logging enabled.

Enable logging for Speech to text REST API for short audio

If you use [Speech to text REST API for short audio](#) and want to enable audio and transcription logging, you need to use the query parameter and value `storeAudio=true` as a part of your REST request. A sample request looks like this:

HTTP

```
https://eastus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveser  
vices/v1?language=en-US&storeAudio=true
```

Enable audio and transcription logging for a custom model endpoint

This method is applicable for [custom speech](#) endpoints only.

Logging can be enabled or disabled in the persistent custom model endpoint settings. When logging is enabled (turned on) for a custom model endpoint, then you don't need to enable logging at the [recognition session level with the SDK or REST API](#). Even when logging isn't enabled for a custom model endpoint, you can enable logging temporarily at the recognition session level with the SDK or REST API.

Warning

For custom model endpoints, the logging setting of your deployed endpoint is prioritized over your session-level setting (SDK or REST API). If logging is enabled for the custom model endpoint, the session-level setting (whether it's set to true or false) is ignored. If logging isn't enabled for the custom model endpoint, the session-level setting determines whether logging is active.

You can enable audio and transcription logging for a custom model endpoint:

- When you create the endpoint using the Speech Studio, REST API, or Speech CLI. For details about how to enable logging for a custom speech endpoint, see [Deploy a custom](#)

speech model.

- When you update the endpoint ([Endpoints_Update](#)) using the [Speech to text REST API](#). For an example of how to update the logging setting for an endpoint, see [Turn off logging for a custom model endpoint](#). But instead of setting the `contentLoggingEnabled` property to `false`, set it to `true` to enable logging for the endpoint.

Turn off logging for a custom model endpoint

To disable audio and transcription logging for a custom model endpoint, you must update the persistent endpoint logging setting using the [Speech to text REST API](#). There isn't a way to disable logging for an existing custom model endpoint using the Speech Studio.

To turn off logging for a custom endpoint, use the [Endpoints_Update](#) operation of the [Speech to text REST API](#). Construct the request body according to the following instructions:

- Set the `contentLoggingEnabled` property within `properties`. Set this property to `true` to enable logging of the endpoint's traffic. Set this property to `false` to disable logging of the endpoint's traffic.

Make an HTTP PATCH request using the URI as shown in the following example. Replace `YourSpeechResourceKey` with your Speech resource key, replace `YourServiceRegion` with your Speech resource region, replace `YourEndpointId` with your endpoint ID, and set the request body properties as previously described.

Azure CLI

```
curl -v -X PATCH -H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey" -H "Content-Type: application/json" -d '{
  "properties": {
    "contentLoggingEnabled": false
  }
}'
"https://YourServiceRegion.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/YourEndpointId"
```

You should receive a response body in the following format:

JSON

```
{
  "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "model": {
    "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/models/9e240dc1-3d2d-
```

```
4ac9-98ec-1be05ba0e9dd"
},
"links": {
  "logs": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/a07164e8-22d1-4eb7-aa31-bf6bb1097f37/files/logs",
  "restInteractive": "https://eastus.stt.speech.microsoft.com/speech/recognition/interactive/cognitiveservices/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "restConversation": "https://eastus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "restDictation": "https://eastus.stt.speech.microsoft.com/speech/recognition/dictation/cognitiveservices/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "webSocketInteractive": "wss://eastus.stt.speech.microsoft.com/speech/recognition/interactive/cognitiveservices/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "webSocketConversation": "wss://eastus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37",
  "webSocketDictation": "wss://eastus.stt.speech.microsoft.com/speech/recognition/dictation/cognitiveservice/s/v1?cid=a07164e8-22d1-4eb7-aa31-bf6bb1097f37"
},
"project": {
  "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/projects/0198f569-cc11-4099-a0e8-9d55bc3d0c52"
},
"properties": {
  "loggingEnabled": false
},
"lastActionDateTime": "2024-07-15T16:30:12Z",
"status": "Succeeded",
"createdDateTime": "2024-07-15T16:29:36Z",
"locale": "en-US",
"displayName": "My Endpoint",
"description": "My Endpoint Description"
}
```

The response body should reflect the new setting. The name of the logging property in the response (`loggingEnabled`) is different from the name of the logging property that you set in the request (`contentLoggingEnabled`).

Get audio and transcription logs

You can access audio and transcription logs using [Speech to text REST API](#). For custom model endpoints, you can also use [Speech Studio](#). See details in the following sections.

Note

Logging data is kept for 30 days. After this period the logs are automatically deleted.

However you can [delete](#) specific logs or a range of available logs at any time.

Get audio and transcription logs with Speech Studio

This method is applicable for [custom model](#) endpoints only.

To download the endpoint logs:

1. Sign in to the [Speech Studio](#).
2. Select **Custom speech** > Your project name > **Deploy models**.
3. Select the link by endpoint name.
4. Under **Content logging**, select **Download log**.

With this approach, you can download all available log sets at once. There's no way to download selected log sets in Speech Studio.

Get audio and transcription logs with Speech to text REST API

You can download all or a subset of available log sets.

This method is applicable for base and [custom model](#) endpoints. To list and download audio and transcription logs:

- Base models: Use the [Endpoints_ListBaseModelLogs](#) operation of the [Speech to text REST API](#). This operation gets the list of audio and transcription logs that are stored when using the default base model of a given language.
- Custom model endpoints: Use the [Endpoints_ListLogs](#) operation of the [Speech to text REST API](#). This operation gets the list of audio and transcription logs that are stored for a given endpoint.

Get log IDs with Speech to text REST API

In some scenarios, you might need to get IDs of the available logs. For example, you might want to delete a specific log as described [later in this article](#).

To get IDs of the available logs:

- Base models: Use the [Endpoints_ListBaseModelLogs](#) operation of the [Speech to text REST API](#). This operation gets the list of audio and transcription logs that are stored when using

the default base model of a given language.

- Custom model endpoints: Use the [Endpoints_ListLogs](#) operation of the [Speech to text REST API](#). This operation gets the list of audio and transcription logs that are stored for a given endpoint.

Here's a sample output of [Endpoints_ListLogs](#). For simplicity, only one log set is shown:

JSON

```
{  
  "values": [  
    {  
      "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/files/logs/2023-03-13\_163715\_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9\_v2\_json",  
        "name": "163715_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9.v2.json",  
        "kind": "Transcription",  
        "properties": {  
          "size": 79920  
        },  
        "createdDateTime": "2024-07-15T16:29:36Z",  
        "links": {  
          "contentUrl": "<Link to download log file>"  
        }  
      },  
      {  
        "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/files/logs/2023-03-13\_163715\_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9\_wav",  
          "name": "163715_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9.wav",  
          "kind": "Audio",  
          "properties": {  
            "size": 932966  
          },  
          "createdDateTime": "2024-07-15T16:29:36Z",  
          "links": {  
            "contentUrl": "<Link to download log file>"  
          }  
        },  
      ]  
    }  
}
```

The locations of each audio and transcription log file are returned in the response body. See the corresponding `kind` property to determine whether the file includes the audio (`"kind": "Audio"`) or the transcription (`"kind": "Transcription"`).

The log ID for each log file is the last part of the URL in the `"self"` element value. The log ID in the following example is `2023-03-13_163715_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9_v2_json`.

JSON

```
"self":  
"https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/endpoints/xxxxxxxx-  
xxxx-xxxx-xxxx-xxxxxxxxxxxx/files/logs/2023-03-13_163715_0420c53d-e6ac-4857-bce0-  
f39c3f9f5ff9_v2_json"
```

Delete audio and transcription logs

Logging data is kept for 30 days. After this period, the logs are automatically deleted. However you can delete specific logs or a range of available logs at any time.

For any base or [custom model](#) endpoint you can delete all available logs, logs for a given time frame, or a particular log based on its Log ID. The deletion process is done asynchronously and can take minutes, hours, one day, or longer depending on the number of log files.

To delete audio and transcription logs you must use the [Speech to text REST API](#). There isn't a way to delete logs using the Speech Studio.

Delete all logs or logs for a given time frame

To delete all logs or logs for a given time frame:

- Base models: Use the [Endpoints_DeleteBaseModelLogs](#) operation of the [Speech to text REST API](#).
- Custom model endpoints: Use the [Endpoints_DeleteLogs](#) operation of the [Speech to text REST API](#).

Optionally, set the `endDate` of the audio logs deletion (specific day, UTC). Expected format: "yyyy-mm-dd". For instance, "2023-03-15" results in deleting all logs on March 15, 2023 and before.

Delete specific log

To delete a specific log by ID:

- Base models: Use the [Endpoints_DeleteBaseModelLog](#) operation of the [Speech to text REST API](#).
- Custom model endpoints: Use the [Endpoints_DeleteLog](#) operation of the [Speech to text REST API](#).

For details about how to get Log IDs, see a previous section [Get log IDs with Speech to text REST API](#).

Since audio and transcription logs have separate IDs (such as IDs 2023-03-13_163715_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9_v2_json and 2023-03-13_163715_0420c53d-e6ac-4857-bce0-f39c3f9f5ff9_wav from a [previous example in this article](#)), when you want to delete both audio and transcription logs you execute separate [delete by ID](#) requests.

Next steps

- [Speech to text quickstart](#)
 - [Speech translation quickstart](#)
 - [Create and train custom speech models](#)
-

Last updated on 10/31/2025

How to troubleshoot Speech SDK issues

08/07/2025

This article provides information to help you solve issues you might encounter when you use the Speech SDK.

Authentication failed

You might observe one of several authentication errors, depending on the programming environment, API, or SDK. Here are some example errors:

- Did you set the speech resource key and region values?
- AuthenticationFailure
- HTTP 403 Forbidden or HTTP 401 Unauthorized. Connection requests without a valid `Ocp-Apim-Subscription-Key` or `Authorization` header are rejected with a status of 403 or 401.
- ValueError: can't construct SpeechConfig with the given arguments (or a variation of this message). This error could be observed, for example, when you run one of the Speech SDK for Python quickstarts without setting valid credentials.
- Exception with an error code: 0x5. This access denied error could be observed, for example, when you run one of the Speech SDK for C# quickstarts without setting valid credentials.

For baseline authentication troubleshooting tips, see [validate your resource key](#) and [validate an authorization token](#).

Validate your resource key

You can verify that you have a valid resource key by running one of the following commands.

! Note

Replace `YOUR_RESOURCE_KEY` and `YOUR_REGION` with your own resource key and associated region.

PowerShell

```
$FetchTokenHeader = @{
    'Content-type'='application/x-www-form-urlencoded'
    'Content-Length'= '0'
    'Ocp-Apim-Subscription-Key' = 'YOUR_RESOURCE_KEY'
```

```
}

$OAuthToken = Invoke-RestMethod -Method POST -Uri
https://YOUR_REGION.api.cognitive.microsoft.com/sts/v1.0/issueToken -Headers
$FetchTokenHeader
$OAuthToken
```

If you entered a valid resource key, the command returns an authorization token, otherwise an error is returned.

Validate an authorization token

If you're using an authorization token for authentication, you might see an authentication error because:

- The authorization token is invalid
- The authorization token is expired

If you use an authorization token for authentication, run one of the following commands to verify that the authorization token is still valid. Tokens are valid for 10 minutes.

! Note

Replace `YOUR_AUDIO_FILE` with the path to your prerecorded audio file. Replace `YOUR_ACCESS_TOKEN` with the authorization token returned in the preceding step. Replace `YOUR_REGION` with the correct region.

PowerShell

```
$SpeechServiceURI =
'https://YOUR_REGION.stt.speech.microsoft.com/speech/recognition/interactive/cognitiveservices/v1?language=en-US'

# $OAuthToken is the authorization token returned by the token service.
$RecoRequestHeader = @{
    'Authorization' = 'Bearer ' + $OAuthToken
    'Transfer-Encoding' = 'chunked'
    'Content-type' = 'audio/wav; codec=audio/pcm; samplerate=16000'
}

# Read audio into byte array.
$audioBytes = [System.IO.File]::ReadAllBytes("YOUR_AUDIO_FILE")

$RecoResponse = Invoke-RestMethod -Method POST -Uri $SpeechServiceURI -Headers
$RecoRequestHeader -Body $audioBytes
```

```
# Show the result.  
$RecoResponse
```

If you entered a valid authorization token, the command returns the transcription for your audio file, otherwise an error is returned.

InitialSilenceTimeout via RecognitionStatus

This issue usually is observed with [single-shot recognition](#) of a single utterance. For example, the error can be returned under the following circumstances:

- The audio begins with a long stretch of silence. In that case, the service stops the recognition after a few seconds and returns `InitialSilenceTimeout`.
- The audio uses an unsupported codec format, which causes the audio data to be treated as silence.

It's OK to have silence at the beginning of audio, but only when you use [continuous recognition](#).

SPXERR_AUDIO_SYS_LIBRARY_NOT_FOUND

This error can be returned, for example, when multiple versions of Python are installed, or if you're not using a supported version of Python. You can try using a different python interpreter or uninstall all python versions and re-install the latest version of python and the Speech SDK.

HTTP 400 Bad Request

This error usually occurs when the request body contains invalid audio data. Only WAV format is supported. Also, check the request's headers to make sure you specify appropriate values for `Content-Type` and `Content-Length`.

HTTP 408 Request Timeout

The error most likely occurs because no audio data is being sent to the service. Network issues might also cause this error.

Next steps

- [Review the release notes](#)

How to get speech to text session ID and transcription ID

05/25/2025

If you use [speech to text](#) and need to open a support case, you're often asked to provide a *Session ID* or *Transcription ID* of the problematic transcriptions to debug the issue. This article explains how to get these IDs.

ⓘ Note

- *Session ID* is used in [real-time speech to text](#) and [speech translation](#).
- *Transcription ID* is used in [batch transcription](#).

Getting Session ID

[Real-time speech to text](#) and [speech translation](#) use either the [Speech SDK](#) or the [REST API](#) for short audio.

To get the Session ID, when using SDK you need to:

1. Enable application logging.
2. Find the Session ID inside the log.

If you use Speech SDK for JavaScript, get the Session ID as described in [this section](#).

If you use [Speech CLI](#), you can also get the Session ID interactively. See details in [this section](#).

With the [speech to text REST API](#) for short audio, you need to inject the session information in the requests. See details in [this section](#).

Enable logging in the Speech SDK

Enable logging for your application as described in [this article](#).

Get Session ID from the log

Open the log file your application produced and look for `SessionId:`. The number that would follow is the Session ID you need. In the following log excerpt example, `0b734c41faf8430380d493127bd44632` is the Session ID.

```
[874193]: 218ms SPX_DBG_TRACE_VERBOSE: audio_stream_session.cpp:1238  
[0000023981752A40]CSpxAudioStreamSession::FireSessionStartedEvent: Firing  
SessionStarted event: SessionId: 0b734c41faf8430380d493127bd44632
```

Get Session ID using JavaScript

If you use Speech SDK for JavaScript, you get Session ID with the help of `sessionStarted` event from the [Recognizer class](#).

See an example of getting Session ID using JavaScript in [this sample](#). Look for `recognizer.sessionStarted = onSessionStarted;` and then for `function onSessionStarted`.

Get Session ID using Speech CLI

If you use [Speech CLI](#), then you see the Session ID in `SESSION STARTED` and `SESSION STOPPED` console messages.

You can also enable logging for your sessions and get the Session ID from the log file as described in [this section](#). Run the appropriate Speech CLI command to get the information on using logs:

Console

```
spx help recognize log
```

Console

```
spx help translate log
```

Provide Session ID using REST API for short audio

Unlike Speech SDK, [Speech to text REST API for short audio](#) doesn't automatically generate a Session ID. You need to generate it yourself and provide it within the REST request.

Generate a GUID inside your code or using any standard tool. Use the GUID value *without dashes or other dividers*. As an example we use `9f4ffa5113a846eba289aa98b28e766f`.

As a part of your REST request use `x-ConnectionId=<GUID>` expression. For our example, a sample request looks like this:

HTTP

```
https://eastus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?language=en-US&X-ConnectionId=9f4ffa5113a846eba289aa98b28e766f
```

9f4ffa5113a846eba289aa98b28e766f is your Session ID.

⚠ Warning

The value of the parameter X-ConnectionId should be in the format of GUID without dashes or other dividers. All other formats aren't supported and will be discarded by the Service.

Example. If the request contains expressions like these:

- X-ConnectionId=9f4ffa51-13a8-46eb-a289-aa98b28e766f (GUID with dividers)
- X-ConnectionId=Request9f4ffa5113a846eba289aa98b28e766f (non-GUID)
- X-ConnectionId=5948f700d2a811ee (non-GUID)

then the value of X-ConnectionId will not be accepted by the system, and the Session won't be found in the logs.

Getting Transcription ID for Batch transcription

Batch transcription API is part of the [Speech to text REST API](#).

The required Transcription ID is the GUID value contained in the main self element of the Response body returned by requests, like [Transcriptions - Submit](#).

The following is and example response body of a [Transcriptions - Submit](#) request. GUID value 537216f8-0620-4a10-ae2d-00bdb423b36f found in the first self element is the Transcription ID.

JSON

```
{
  "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/transcriptions/537216f8-0620-4a10-ae2d-00bdb423b36f",
  "model": {
    "self": "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/models/base/824bd685-2d45-424d-bb65-c3fe99e32927"
  },
  "links": {
```

```
"files":  
  "https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/transcriptions/53721  
  6f8-0620-4a10-ae2d-00bdb423b36f/files"  
,  
  "properties": {  
    "diarizationEnabled": false,  
    "wordLevelTimestampsEnabled": false,  
    "channels": [  
      0,  
      1  
    ],  
    "punctuationMode": "DictatedAndAutomatic",  
    "profanityFilterMode": "Masked"  
,  
    "lastActionDateTime": "2021-11-19T14:09:51Z",  
    "status": "NotStarted",  
    "createdDateTime": "2021-11-19T14:09:51Z",  
    "locale": "ru-RU",  
    "displayName": "transcriptiontest"  
  }  
}
```

 **Note**

Use the same technique to determine different IDs required for debugging issues related to [custom speech](#), like uploading a dataset using [Datasets Create](#) request.

 **Note**

You can also see all existing transcriptions and their Transcription IDs for a given Speech resource by using [Transcriptions - Get](#) request.

How to track Speech SDK memory usage

08/07/2025

The Speech SDK is based on a native code base projected into multiple programming languages through a series of interoperability layers. Each language-specific projection has idiomatically correct features to manage the object lifecycle. Additionally, the Speech SDK includes memory management tooling to track resource usage with object logging and object limits.

How to read object logs

If [Speech SDK logging is enabled](#), tracking tags are emitted to enable historical object observation. These tags include:

- `TrackHandle` or `StopTracking`
- The object type
- The current number of objects that are tracked the type of the object, and the current number being tracked.

Here's a sample log:

terminal

```
(284): 8604ms SPX_DBG_TRACE_VERBOSE: handle_table.h:90 TrackHandle
type=Microsoft::CognitiveServices::Speech::Impl::ISpxRecognitionResult
handle=0x0x7f688401e1a0, ptr=0x0x7f688401e1a0, total=19
```

Set a warning threshold

You can create a warning threshold, and if that threshold is exceeded (assuming logging is enabled), a warning message is logged. The warning message contains a dump of all objects in existence along with their count. This information can be used to better understand issues.

To enable a warning threshold, it must be specified on a `SpeechConfig` object. This object is checked when a new recognizer is created. In the following examples, let's assume that you created an instance of `SpeechConfig` called `config`:

C#

```
config SetProperty("SPEECH-ObjectCountWarnThreshold", "10000");
```

💡 Tip

The default value for this property is 10,000.

Set an error threshold

Using the Speech SDK, you can set the maximum number of objects allowed at a given time. If this setting is enabled, when the maximum number is hit, attempts to create new recognizer objects fail. Existing objects continue to work.

Here's a sample error:

terminal

```
Runtime error: The maximum object count of 500 has been exceeded.  
The threshold can be adjusted by setting the SPEECH-ObjectCountErrorThreshold  
property on the SpeechConfig object.  
Handle table dump by object type:  
class Microsoft::CognitiveServices::Speech::Impl::ISpxRecognitionResult 0  
class Microsoft::CognitiveServices::Speech::Impl::ISpxRecognizer 0  
class Microsoft::CognitiveServices::Speech::Impl::ISpxAudioConfig 0  
class Microsoft::CognitiveServices::Speech::Impl::ISpxSpeechConfig 0
```

To enable an error threshold, it must be specified on a `SpeechConfig` object. This object is checked when a new recognizer is created. In the following examples, let's assume that you created an instance of `SpeechConfig` called `config`:

C#

```
config SetProperty("SPEECH-ObjectCountErrorThreshold", "10000");
```

💡 Tip

The default value for this property is the platform-specific maximum value for a `size_t` data type. A typical recognition will consume between 7 and 10 internal objects.

Next steps

- [Learn more about the Speech SDK](#)

Generate a REST API client library for the Speech to text REST API

08/07/2025

The Speech service offers a Swagger specification to interact with a handful of REST APIs used to import data, create models, test model accuracy, create custom endpoints, queue up batch transcriptions, and manage subscriptions. Most operations available through the [custom speech area of the Speech Studio](#) can be completed programmatically using these APIs.

! Note

Speech service has several REST APIs for [Speech to text](#) and [Text to speech](#).

However only the [speech to text REST API](#) and [custom voice REST API](#) are documented in the Swagger specification. See the documents referenced in the previous paragraph for the information on all other Speech service REST APIs.

Generating code from the Swagger specification

The [Swagger specification](#) has options that allow you to quickly test for various paths. However, sometimes it's desirable to generate code for all paths, creating a single library of calls that you can base future solutions on. Let's take a look at the process to generate a Python library for the Speech to text REST API version 3.1.

You need to set Swagger to the region of your Speech resource. You can confirm the region in the **Overview** part of your Speech resource settings in Azure portal. The complete list of supported regions is available [here](#).

1. In a browser, go to <https://editor.swagger.io>
2. Select **File**, select **Import URL**,
3. Enter the URL `https://github.com/Azure/azure-rest-api-specs/blob/master/specification/cognitiveservices/data-plane/Speech/SpeechToText/stable/v3.1/speechtotext.json` and select **OK**.
4. Select **Generate Client** and select **python**. The client library downloads to your computer in a `.zip` file.
5. Extract everything from the download. You might use `tar -xf` to extract everything.
6. Install the extracted module into your Python environment:
`pip install path/to/package/python-client`

7. The installed package is named `swagger_client`. Check that the installation succeeded:

```
python -c "import swagger_client"
```

You can use the Python library that you generated with the [Speech service samples on GitHub ↗](#).

Next steps

- [Speech service samples on GitHub ↗](#).
- [Speech to text REST API](#)

Summary

Article • 03/20/2025

[+] Expand table

Members	Descriptions
namespace Microsoft::CognitiveServices::Speech	
namespace Microsoft::CognitiveServices::Speech::Audio	
namespace Microsoft::CognitiveServices::Speech::Diagnostics::Logging	
namespace Microsoft::CognitiveServices::Speech::Dialog	
namespace Microsoft::CognitiveServices::Speech::Intent	
namespace Microsoft::CognitiveServices::Speech::Speaker	
namespace Microsoft::CognitiveServices::Speech::Transcription	
namespace Microsoft::CognitiveServices::Speech::Translation	
class SpeechSynthesisRequest::InputStream	Represents an input stream for speech synthesis request. Note: This class is in preview and may be subject to change in future versions. Added in version 1.37.0.
class Transcription::ConversationTranscriber::PrivatePropertyCollection	
class Transcription::MeetingTranscriber::PrivatePropertyCollection	

Microsoft.CognitiveServices.Speech Namespace

Classes

 [Expand table](#)

AudioDataStream	Provides audio data as a stream. Added in 1.4.0
AutoDetectSourceLanguageConfig	Configures options for automatic detection of languages. Updated in 1.13.0
AutoDetectSourceLanguageResult	Contains languages detected by the Speech service. Added in 1.9.0
CancellationDetails	Contains detailed information about why a result was canceled.
ClassLanguageModel	Represents a list of grammars for dynamic grammar scenarios. Added in 1.7.0
Connection	A proxy class for managing connection to the speech service of the specified Recognizer. Added in 1.2.0
ConnectionEventArgs	Contains payload for Connected/Disconnected events Added in 1.2.0
ConnectionMessage	Represents implementation-specific messages sent to and received from the speech service. For debugging only. Added in 1.10.0
ConnectionMessageEventArgs	Contains payload for MessageReceived events of a Connection instance. Added in 1.10.0
DetailedSpeechRecognitionResult	Contains recognition details including confidence score, recognized text, raw lexical form, normalized form, and normalized form with masked profanity. Changed in 1.7.0
EmbeddedSpeechConfig	Class that defines embedded (offline) speech configuration.
Grammar	Represents base class grammar for customizing speech recognition. Added in 1.5.0
GrammarList	Represents a list of grammars for dynamic grammar scenarios. Added in 1.7.0
GrammarPhrase	Represents a phrase that can be spoken by the user. Added in 1.5.0
HybridSpeechConfig	Class that defines hybrid (cloud and embedded) configurations for speech recognition and speech synthesis.

KeywordRecognitionEvent	Class for the events emitted by the KeywordRecognizer .
Args	
KeywordRecognitionModel	Represents keyword recognition model that can trigger an event when pre-defined keywords are spoken.
KeywordRecognitionResult	Contains the results emitted by the KeywordRecognizer .
KeywordRecognizer	Recognizes a word or short phrase using a keyword model.
NoMatchDetails	Contains detailed information for NoMatch recognition results.
PersonalVoiceSynthesisRequest	Contains detailed information for a personal voice synthesis request. Note: This class is in preview and may be subject to change in future versions. Added in version 1.39.0
PhonemeLevelTimingResult	Phoneme level timing result. Added in 1.14.0
PhraseListGrammar	Identifies known phrases in audio data. Added in 1.5.0
PronunciationAssessmentNBestPhoneme	Pronunciation assessment nbest phoneme result Added in 1.20.0
PropertyCollection	Class to retrieve or set a property value from a property collection.
RecognitionEventArgs	Contains payload for recognition events like Speech Start/End Detected.
RecognitionResult	Contains detailed information about result of a recognition operation.
Recognizer	Base class that mostly contains common event handlers.
SessionEventArgs	Contains payload for SessionStarted and SessionStopped events.
SourceLanguageConfig	Source Language configuration. Added in 1.17.0
SourceLanguageRecognizer	Detects the spoken language on the input audio. Added in version 1.17.0
SpeechConfig	Information about your subscription, including your key and region, endpoint, host, or authorization token.
SpeechRecognitionCanceledEventArgs	Contains payload of speech recognition canceled result events.
SpeechRecognitionEventArgs	Contains payload of speech recognizing/recognized events.
SpeechRecognitionModel	Speech recognition model information.
SpeechRecognitionResult	Contains result of speech recognition.

SpeechRecognitionResult	Extension methods for speech recognition result
Extensions	
SpeechRecognizer	Transcribes speech into text. Speech can arrive via microphone, audio file, or other audio input stream.
SpeechSynthesisBookmarkEventArgs	Contains bookmark event in synthesized speech. Added in 1.16.0
SpeechSynthesisCancellationDetails	Contains detailed information about why a speech synthesis result was canceled. Added in 1.4.0
SpeechSynthesisEventArgs	Contains payload of speech synthesis events. Added in 1.4.0
SpeechSynthesisRequest	Contains detailed information for a speech synthesis request. Note: This class is in preview and may be subject to change in future versions. Added in version 1.37.0
SpeechSynthesisRequestInputStream	Manages the input stream for a speech synthesis request. Added in version 1.37.0
SpeechSynthesisResult	Contains detailed information about result of a speech synthesis operation. Added in 1.4.0
SpeechSynthesisVisemeEventArgs	Contains facial pose events that correspond to time-based offsets in synthesized speech. Added in 1.16.0
SpeechSynthesisWordBoundaryEventArgs	Contains location and length details about words in synthesized speech. Added in 1.7.0
SpeechSynthesizer	Performs speech synthesis to speaker, file, or other audio output streams, and gets synthesized audio as result. Updated in 1.16.0
SpeechTranslationConfig	Speech translation configuration.
SpeechTranslationModel	Speech translation model information.
SyllableLevelTimingResult	Syllable level timing result. Added in 1.20.0
SynthesisVoicesResult	Contains detailed information about the retrieved synthesis voices list. Added in 1.16.0
VoiceInfo	Contains detailed information about the synthesis voice. Updated in 1.17.0
WordLevelTimingResult	For a recognized word in speech audio, contains the offset to the start and the duration, in ticks. 1 tick = 100 ns. Added in 1.7.0

Enums

CancellationErrorCode	Lists error codes possible when CancellationReason is Error . Added in 1.1.0
CancellationReason	Lists the possible reasons a recognition result might be canceled.
NoMatchReason	Lists the possible reasons a recognition result was not recognized.
OutputFormat	Output format.
ProfanityOption	Removes profanity (swearing), or replaces letters of profane words with stars. Added in 1.5.0
PropertyId	Lists speech property IDs.
RecognitionFactorScope	Lists the scope that a recognition factor applies to.
ResultReason	Describes a recognition result.
ServicePropertyChannel	Lists channels used to pass property settings to service. Added in 1.5.0
SpeechSynthesisBoundaryType	Defines the boundary type of speech synthesis boundary event Added in version 1.21.0
SpeechSynthesisOutputFormat	Lists synthesis output audio formats.
SpeechSynthesisRequestInputType	Defines the type of input for a speech synthesis request.
StreamStatus	Lists possible status values of an audio data stream. Added in 1.4.0
SynthesisVoiceGender	Lists synthesis voice gender. Added in version 1.17.0
SynthesisVoiceStatus	Lists synthesis voice status.
SynthesisVoiceType	Lists synthesis voice types.

com.microsoft.cognitiveservices.speech

Package: com.microsoft.cognitiveservices.speech

Maven Artifact: [com.microsoft.cognitiveservices.speech:client-sdk:1.47.0](#)

Classes

 [Expand table](#)

AudioDataStream	Represents audio data stream used for operating audio data as a stream.
AutoDetectSourceLanguageConfig	Represents auto detect source language configuration used for specifying the possible source language candidates Note: close() must be called in order to release underlying resources held by the object.
AutoDetectSourceLanguageResult	Represents the result of auto detecting source languages Added in version 1.8.0
CancellationDetails	Contains detailed information about why a result was canceled.
ClassLanguageModel	Represents a ClassLanguageModel.
Connection	Connection is a proxy class for managing connection to the speech service of the specified Recognizer.
ConnectionEventArgs	Defines payload for connection events like Connected/Disconnected.
ConnectionMessage	ConnectionMessage represents implementation specific messages sent to and received from the speech service.
ConnectionMessageEventArgs	Defines payload for Connection's MessageReceived events.
Diagnostics	Native logging and other diagnostics
EmbeddedSpeechConfig	Class that defines embedded (offline) speech configuration.
Grammar	Represents a generic grammar used to assist in improving speech recognition accuracy.
GrammarList	Allows adding multiple grammars to a SpeechRecognizer to improve the accuracy of speech recognition.
HybridSpeechConfig	Class that defines hybrid (cloud and embedded) configurations for speech recognition and speech synthesis.
KeywordRecognitionEventArgs	Defines content of an keyword recognizing/recognized events.

KeywordRecognitionModel	Represents a keyword recognition model for recognizing when the user says a keyword to initiate further speech recognition.
KeywordRecognitionResult	Defines result of keyword recognition.
KeywordRecognizer	Performs keyword recognition on the speech input.
NoMatchDetails	Contains detailed information for NoMatch recognition results.
PhraseListGrammar	Allows additions of new phrases to improve speech recognition.
PronunciationAssessmentConfig	Represents pronunciation assessment configuration.
PronunciationAssessmentResult	Represents the result of pronunciation assessment.
PropertyCollection	Represents collection of properties and their values.
RecognitionEventArgs	Defines payload for recognition events like Speech Start/End Detected
RecognitionResult	Contains detailed information about result of a recognition operation.
Recognizer	Defines the base class Recognizer which mainly contains common event handlers.
SessionEventArgs	Defines payload for SessionStarted/Stopped events.
SourceLanguageConfig	Represents source language configuration used for specifying recognition source language.
SpeechConfig	Speech configuration.
SpeechRecognitionCanceledEventArgs	Defines payload of speech recognition canceled events.
SpeechRecognitionEventArgs	Defines contents of speech recognizing/recognized event.
SpeechRecognitionModel	Contains detailed speech recognition model information.
SpeechRecognitionResult	Defines result of speech recognition.
SpeechRecognizer	Performs speech recognition from microphone, file, or other audio input streams, and gets transcribed text as result.
SpeechSynthesisBookmarkEventArgs	Defines contents of speech synthesis bookmark event.

SpeechSynthesis	Contains detailed information about why a speech synthesis was canceled.
CancellationDetails	
SpeechSynthesisEvent	Defines contents of speech synthesis related event.
Args	
SpeechSynthesisResult	Contains detailed information about result of a speech synthesis operation.
SpeechSynthesis	Defines contents of speech synthesis viseme event.
VisemeEventArgs	
SpeechSynthesisWord	Defines contents of speech synthesis word boundary event.
BoundaryEventArgs	
SpeechSynthesizer	Performs speech synthesis to speaker, file, or other audio output streams, and gets synthesized audio as result.
SpeechTranslation	Contains detailed speech translation model information.
Model	
SynthesisVoicesResult	Contains detailed information about the retrieved synthesis voices list.
VoiceInfo	Contains detailed information about the synthesis voice information.

Enums

 [Expand table](#)

CancellationErrorCode	Defines error code in case that CancellationReason is Error.
CancellationReason	Defines the possible reasons a recognition result might be canceled.
NoMatchReason	Defines the possible reasons a recognition result might not be recognized.
OutputFormat	Define Speech Recognizer output formats.
ProfanityOption	Define profanity option for response result.
PronunciationAssessmentGradingSystem	Defines the point system for pronunciation score calibration; default value is FivePoint.
PronunciationAssessmentGranularity	Defines the pronunciation evaluation granularity; default value is Phoneme.
PropertyId	Defines property ids.
ResultReason	Defines the possible reasons a recognition result might be generated.

ServicePropertyChannel	Defines channels used to send service properties.
SpeechSynthesisBoundaryType	Defines the boundary type of speech synthesis boundary event.
SpeechSynthesisOutputFormat	Defines the possible speech synthesis output audio format.
StreamStatus	Defines the possible status of audio data stream.
SynthesisVoiceGender	Define synthesis voice gender.
SynthesisVoiceStatus	Defines the status of synthesis voice.
SynthesisVoiceType	Define synthesis voice type.

microsoft-cognitiveservices-speech-sdk package

Classes

 [Expand table](#)

ActivityReceivedEventArgs	Defines contents of received message/events.
EventArgs	
AudioConfig	Represents audio input configuration used for specifying what type of input to use (microphone, file, stream).
AudioInputStream	Represents audio input stream used for custom audio input configurations.
AudioOutputStream	Represents audio output stream used for custom audio output configurations.
AudioStreamFormat	Represents audio stream format used for custom audio input configurations.
AutoDetectSourceLanguageConfig	Language auto detect configuration.
AutoDetectSourceLanguageResult	Output format
AvatarConfig	Defines the talking avatar configuration.
AvatarEventArgs	Defines content for talking avatar events.
AvatarSynthesizer	Defines the avatar synthesizer.
AvatarVideoFormat	Defines the avatar output video format.
AvatarWebRTCConnectionResult	Defines the avatar WebRTC connection result.
BaseAudioPlayer	Base audio player class TODO: Plays only PCM for now.
BotFrameworkConfig	Class that defines configurations for the dialog service connector object for using a Bot Framework backend.

CancellationDetails	Contains detailed information about why a result was canceled.
CancellationDetailsBase	Contains detailed information about why a result was canceled.
Connection	Connection is a proxy class for managing connection to the speech service of the specified Recognizer. By default, a Recognizer autonomously manages connection to service when needed. The Connection class provides additional methods for users to explicitly open or close a connection and to subscribe to connection status changes. The use of Connection is optional, and mainly for scenarios where fine tuning of application behavior based on connection status is needed. Users can optionally call Open() to manually set up a connection in advance before starting recognition on the Recognizer associated with this Connection. If the Recognizer needs to connect or disconnect to service, it will setup or shutdown the connection independently. In this case the Connection will be notified by change of connection status via Connected/Disconnected events. Added in version 1.2.1.
ConnectionEventArgs	Defines payload for connection events like Connected/Disconnected. Added in version 1.2.0
ConnectionMessage	ConnectionMessage represents implementation specific messages sent to and received from the speech service. These messages are provided for debugging purposes and should not be used for production use cases with the Azure Cognitive Services Speech Service. Messages sent to and received from the Speech Service are subject to change without notice. This includes message contents, headers, payloads, ordering, etc. Added in version 1.11.0.
ConnectionMessageEventArgs	
Conversation	
ConversationExpirationEventArgs	Defines content for session events like SessionStarted/Stopped, SoundStarted/Stopped.
ConversationParticipantsChangedEventArgs	Defines content for session events like SessionStarted/Stopped, SoundStarted/Stopped.
ConversationTranscriber	Performs speech recognition with speaker separation from microphone, file, or other audio input streams, and gets transcribed text as result.
ConversationTranscriptionCanceledEventArgs	Defines content of a RecognitionErrorEvent.
ConversationTranscriptionEventArgs	Defines contents of conversation transcribed/transcribing event.

EventArgs	
ConversationTranscriptionResult	Defines result of conversation transcription.
ConversationTranslationCanceledEventArgs	
ConversationTranslationEventArgs	Defines payload for session events like Speech Start/End Detected
ConversationTranslationResult	Translation text result.
ConversationTranslator	Join, leave or connect to a conversation.
Coordinate	Defines a coordinate in 2D space.
CustomCommandsConfig	Class that defines configurations for the dialog service connector object for using a CustomCommands backend.
Diagnostics	Defines diagnostics API for managing console output Added in version 1.21.0
DialogServiceConfig	Class that defines base configurations for dialog service connector
DialogServiceConnector	Dialog Service Connector
KeywordRecognitionModel	Represents a keyword recognition model for recognizing when the user says a keyword to initiate further speech recognition.
Meeting	
MeetingTranscriber	
MeetingTranscriptionCanceledEventArgs	Defines content of a MeetingTranscriptionCanceledEvent.
MeetingTranscriptionEventArgs	Defines contents of meeting transcribed/transcribing event.
NoMatchDetails	Contains detailed information for NoMatch recognition results.
Participant	Represents a participant in a conversation. Added in version 1.4.0

PhraseList	Allows additions of new phrases to improve speech recognition.
Grammar	Phrases added to the recognizer are effective at the start of the next recognition, or the next time the SpeechSDK must reconnect to the speech service.
Pronunciation Assessment Config	Pronunciation assessment configuration.
Pronunciation Assessment Result	Pronunciation assessment results.
Property Collection	Represents collection of properties and their values.
PullAudioInputStream	Represents audio input stream used for custom audio input configurations.
PullAudioInputStreamCallback	An abstract base class that defines callback methods (read() and close()) for custom audio input streams).
PushAudioOutputStream	Represents memory backed push audio output stream used for custom audio output configurations.
PushAudioInputStream	Represents memory backed push audio input stream used for custom audio input configurations.
PushAudioOutputStream	Represents audio output stream used for custom audio output configurations.
PushAudioOutputStreamCallback	An abstract base class that defines callback methods (write() and close()) for custom audio output streams).
RecognitionEventArgs	Defines payload for session events like Speech Start/End Detected
RecognitionResult	Defines result of speech recognition.
Recognizer	Defines the base class Recognizer which mainly contains common event handlers.
ServiceEventArgs	Defines payload for any Service message event Added in version 1.9.0
SessionEventArgs	Defines content for session events like SessionStarted/Stopped, SoundStarted/Stopped.
SourceLanguageConfig	Source Language configuration.

SpeakerAudio	Represents the speaker playback audio destination, which only works in browser.
Destination	Note: the SDK will try to use Media Source Extensions to play audio. Mp3 format has better supports on Microsoft Edge, Chrome and Safari (desktop), so, it's better to specify mp3 format for playback.
SpeechConfig	Speech configuration.
SpeechConfigImpl	
SpeechRecognitionCanceledEventArgs	
Speech Recognition EventArgs	Defines contents of speech recognizing/recognized event.
Speech Recognition Result	Defines result of speech recognition.
Speech Recognizer	Performs speech recognition from microphone, file, or other audio input streams, and gets transcribed text as result.
SpeechSynthesis BookmarkEvent Args	Defines contents of speech synthesis bookmark event.
SpeechSynthesis EventArgs	Defines contents of speech synthesis events.
SpeechSynthesis Result	Defines result of speech synthesis.
SpeechSynthesis VisemeEvent Args	Defines contents of speech synthesis viseme event.
SpeechSynthesis WordBoundary EventArgs	Defines contents of speech synthesis word boundary event.
Speech Synthesizer	Defines the class SpeechSynthesizer for text to speech. Updated in version 1.16.0
Speech Translation Config	Speech translation configuration.
SynthesisResult	Base class for synthesis results
SynthesisVoices Result	Defines result of speech synthesis.

Synthesizer	
Translation	Define payload of speech recognition canceled result events.
Recognition	
CanceledEvent	
Args	
Translation	Translation text result event arguments.
Recognition	
EventArgs	
Translation	Translation text result.
Recognition	
Result	
Translation	Translation recognizer
Recognizer	
Translation	Translation Synthesis event arguments
SynthesisEvent	
Args	
Translation	Defines translation synthesis result, i.e. the voice output of the translated text in the target language.
SynthesisResult	
Translations	Represents collection of parameters and their values.
TurnStatus	Defines contents of received message/events.
ReceivedEvent	
Args	
User	
VoiceInfo	Information about Speech Synthesis voice Added in version 1.20.0.

Interfaces

[\[\]](#) [Expand table](#)

CancellationEventArgs	
ConversationInfo	
IParticipant	Represents a participant in a conversation. Added in version 1.4.0
IPlayer	Represents audio player interface to control the audio playback, such as pause, resume, etc.
IVoiceJson	

[MeetingInfo](#)

[VoiceSignature](#)

Enums

 [Expand table](#)

AudioFormatTag	
CancellationErrorCode	Defines error code in case that CancellationReason is Error. Added in version 1.1.0.
CancellationReason	Defines the possible reasons a recognition result might be canceled.
LanguageIdMode	Language Identification mode
LogLevel	
NoMatchReason	Defines the possible reasons a recognition result might not be recognized.
OutputFormat	Define Speech Recognizer output formats.
ParticipantChangedReason	
ProfanityOption	Profanity option. Added in version 1.7.0.
PronunciationAssessmentGradingSystem	Defines the point system for pronunciation score calibration; default value is FivePoint. Added in version 1.15.0
PronunciationAssessmentGranularity	Defines the pronunciation evaluation granularity; default value is Phoneme. Added in version 1.15.0
PropertyId	Defines speech property ids.
ResultReason	Defines the possible reasons a recognition result might be generated.
ServicePropertyChannel	Defines channels used to pass property settings to service. Added in version 1.7.0.
SpeechSynthesisBoundaryType	Defines the boundary type of speech synthesis boundary event.
SpeechSynthesisOutputFormat	Define speech synthesis audio output formats. SpeechSynthesisOutputFormat Updated in version 1.17.0
SynthesisVoiceGender	Defines the gender of synthesis voices. Added in version 1.20.0.
SynthesisVoiceType	

Speech SDK for Objective-C Reference

Classes

- [SPXAudioConfiguration class](#)
- [SPXAudioDataStream class](#)
- [SPXAudioInputStream class](#)
- [SPXAudioOutputStream class](#)
- [SPXAudioProcessingOptions class](#)
- [SPXAudioStreamFormat class](#)
- [SPXAutoDetectSourceLanguageConfiguration class](#)
- [SPXAutoDetectSourceLanguageResult class](#)
- [SPXCancellationDetails class](#)
- [SPXConnection class](#)
- [SPXConnectionEventArgs class](#)
- [SPXConnectionMessage class](#)
- [SPXConnectionMessageEventArgs class](#)
- [SPXContentAssessmentResult class](#)
- [SPXConversation class](#)
- [SPXConversationExpirationEventArgs class](#)
- [SPXConversationParticipantsChangedEventArgs class](#)
- [SPXConversationTranscriber class](#)
- [SPXConversationTranscriptionCanceledEventArgs class](#)
- [SPXConversationTranscriptionEventArgs class](#)
- [SPXConversationTranscriptionResult class](#)
- [SPXConversationTranslationCanceledEventArgs class](#)
- [SPXConversationTranslationEventArgs class](#)
- [SPXConversationTranslationResult class](#)
- [SPXConversationTranslator class](#)
- [SPXDialogBotFrameworkConfiguration class](#)
- [SPXDialogCustomCommandsConfiguration class](#)
- [SPXDialogServiceConfiguration class](#)
- [SPXDialogServiceConnector class](#)
- [SPXDialogServiceConnectorActivityReceivedEventArgs class](#)
- [SPXDialogServiceConnectorTurnStatusReceivedEventArgs class](#)
- [SPXEmbeddedSpeechConfiguration class](#)
- [SPXEventLogger class](#)
- [SPXFileLogger class](#)
- [SPXGrammar class](#)
- [SPXGrammarPhrase class](#)

- [SPXKeywordRecognitionCanceledEventArgs](#) class
- [SPXKeywordRecognitionEventArgs](#) class
- [SPXKeywordRecognitionModel](#) class
- [SPXKeywordRecognitionResult](#) class
- [SPXKeywordRecognizer](#) class
- [SPXMeeting](#) class
- [SPXMeetingTranscriber](#) class
- [SPXMeetingTranscriptionCanceledEventArgs](#) class
- [SPXMeetingTranscriptionEventArgs](#) class
- [SPXMeetingTranscriptionResult](#) class
- [SPXMemoryLogger](#) class
- [SPXNBestPhoneme](#) class
- [SPXNoMatchDetails](#) class
- [SPXParticipant](#) class
- [SPXPhonemeLevelTimingResult](#) class
- [SPXPhraseListGrammar](#) class
- [SPXPronunciationAssessmentConfiguration](#) class
- [SPXPronunciationAssessmentResult](#) class
- [SPXPullAudioInputStream](#) class
- [SPXPullAudioOutputStream](#) class
- [SPXPushAudioInputStream](#) class
- [SPXPushAudioOutputStream](#) class
- [SPXRecognitionEventArgs](#) class
- [SPXRecognitionResult](#) class
- [SPXRecognizer](#) class
- [SPXSessionEventArgs](#) class
- [SPXSourceLanguageConfiguration](#) class
- [SPXSpeechConfiguration](#) class
- [SPXSpeechRecognitionCanceledEventArgs](#) class
- [SPXSpeechRecognitionEventArgs](#) class
- [SPXSpeechRecognitionModelInfo](#) class
- [SPXSpeechRecognitionResult](#) class
- [SPXSpeechRecognizer](#) class
- [SPXSpeechSynthesisBookmarkEventArgs](#) class
- [SPXSpeechSynthesisCancellationDetails](#) class
- [SPXSpeechSynthesisEventArgs](#) class
- [SPXSpeechSynthesisResult](#) class
- [SPXSpeechSynthesisVisemeEventArgs](#) class
- [SPXSpeechSynthesisWordBoundaryEventArgs](#) class
- [SPXSpeechSynthesizer](#) class
- [SPXSpeechTranslationConfiguration](#) class

- [SPXSpeechTranslationModelInfo](#) class
- [SPXSyllableLevelTimingResult](#) class
- [SPXSynthesisVoicesResult](#) class
- [SPXTimingResult](#) class
- [SPXTranslationRecognitionCanceledEventArgs](#) class
- [SPXTranslationRecognitionEventArgs](#) class
- [SPXTranslationRecognitionResult](#) class
- [SPXTranslationRecognizer](#) class
- [SPXTranslationSynthesisEventArgs](#) class
- [SPXTranslationSynthesisResult](#) class
- [SPXUser](#) class
- [SPXVoiceInfo](#) class
- [SPXWordLevelTimingResult](#) class

Protocols

- [SPXPropertyCollection](#) protocol

Enums

- [SPXAudioProcessingConstants](#) enum
- [SPXAudioStreamContainerFormat](#) enum
- [SPXCancellationErrorCode](#) enum
- [SPXCancellationReason](#) enum
- [SPXLogLevel](#) enum
- [SPXNoMatchReason](#) enum
- [SPXOutputFormat](#) enum
- [SPXParticipantChangedReason](#) enum
- [SPXPronunciationAssessmentGradingSystem](#) enum
- [SPXPronunciationAssessmentGranularity](#) enum
- [SPXPropertyId](#) enum
- [SPXResultReason](#) enum
- [SPXServicePropertyChannel](#) enum
- [SPXSpeechConfigProfanityOption](#) enum
- [SPXSpeechSynthesisBoundaryType](#) enum
- [SPXSpeechSynthesisOutputFormat](#) enum
- [SPXStreamStatus](#) enum
- [SPXSynthesisVoiceGender](#) enum
- [SPXSynthesisVoiceStatus](#) enum
- [SPXSynthesisVoiceType](#) enum

TypeDefs

- [SPXConnectionEventHandler](#) typedef
- [SPXConnectionMessageEventHandler](#) typedef
- [SPXConversationExpirationEventHandler](#) typedef
- [SPXConversationParticipantsChangedEventHandler](#) typedef
- [SPXConversationSessionEventHandler](#) typedef
- [SPXConversationTranscriptionCanceledEventHandler](#) typedef
- [SPXConversationTranscriptionEventHandler](#) typedef
- [SPXConversationTranslationCanceledEventHandler](#) typedef
- [SPXConversationTranslationEventHandler](#) typedef
- [SPXDialogServiceConnectorActivityReceivedEventHandler](#) typedef
- [SPXDialogServiceConnectorAsyncCompletionHandler](#) typedef
- [SPXDialogServiceConnectorCanceledEventHandler](#) typedef
- [SPXDialogServiceConnectorInteractionIdHandler](#) typedef
- [SPXDialogServiceConnectorRecognitionEventHandler](#) typedef
- [SPXDialogServiceConnectorRecognitionResultHandler](#) typedef
- [SPXDialogServiceConnectorTurnStatusReceivedEventHandler](#) typedef
- [SPXEventLoggerHandler](#) typedef
- [SPXKeywordRecognitionCanceledEventHandler](#) typedef
- [SPXKeywordRecognitionEventHandler](#) typedef
- [SPXMeetingTranscriptionCanceledEventHandler](#) typedef
- [SPXMeetingTranscriptionEventHandler](#) typedef
- [SPXPullAudioInputStreamCloseHandler](#) typedef
- [SPXPullAudioInputStreamGetPropertyHandler](#) typedef
- [SPXPullAudioInputStreamReadHandler](#) typedef
- [SPXPushAudioOutputStreamCloseHandler](#) typedef
- [SPXPushAudioOutputStreamWriteHandler](#) typedef
- [SPXRecognitionEventHandler](#) typedef
- [SPXSessionEventHandler](#) typedef
- [SPXSpeechRecognitionAsyncCompletionHandler](#) typedef
- [SPXSpeechRecognitionCanceledEventHandler](#) typedef
- [SPXSpeechRecognitionEventHandler](#) typedef
- [SPXSpeechSynthesisBookmarkEventHandler](#) typedef
- [SPXSpeechSynthesisEventHandler](#) typedef
- [SPXSpeechSynthesisVisemeEventHandler](#) typedef
- [SPXSpeechSynthesisWordBoundaryEventHandler](#) typedef
- [SPXTranslationRecognitionAsyncCompletionHandler](#) typedef
- [SPXTranslationRecognitionCanceledEventHandler](#) typedef
- [SPXTranslationRecognitionEventHandler](#) typedef

- [SPXTranslationSynthesisEventHandler typedef](#)
-

Last updated on 11/11/2025

speech Package

Microsoft Speech SDK for Python

Packages

 [Expand table](#)

diagnostics	Microsoft Speech SDK for Python
-----------------------------	---------------------------------

Modules

 [Expand table](#)

audio	Classes that are concerned with the handling of audio input to the various recognizers, and audio output from the speech synthesizer.
dialog	Classes related to dialog service connector.
enums	
intent	Classes related to intent recognition from speech.
interop	
languageconfig	Classes that are concerned with the handling of language configurations
properties	
speech	Classes related to recognizing text from speech, synthesizing speech from text, and general classes used in the various recognizers.
transcription	Classes related to conversation transcription.
translation	Classes related to translation of speech to other languages.
version	

Classes

 [Expand table](#)

AudioDataStream	Represents audio data stream used for operating audio data as a stream.
---------------------------------	---

	Generates an audio data stream from a speech synthesis result (type <code>SpeechSynthesisResult</code>) or a keyword recognition result (type <code>KeywordRecognitionResult</code>).
AutoDetectSourceLanguageResult	Represents auto detection source language result. The result can be initialized from a speech recognition result.
CancellationDetails	
Connection	<p>Proxy class for managing the connection to the speech service of the specified Recognizer.</p> <p>By default, a Recognizer autonomously manages connection to service when needed. The Connection class provides additional methods for users to explicitly open or close a connection and to subscribe to connection status changes. The use of Connection is optional. It is intended for scenarios where fine tuning of application behavior based on connection status is needed. Users can optionally call open to manually initiate a service connection before starting recognition on the Recognizer associated with this Connection. After starting a recognition, calling open or close might fail. This will not impact the Recognizer or the ongoing recognition. Connection might drop for various reasons, the Recognizer will always try to reinstitute the connection as required to guarantee ongoing operations. In all these cases connected/disconnected events will indicate the change of the connection status.</p>
	<p>ⓘ Note</p> <p>Updated in version 1.17.0.</p>
	Constructor for internal use.
ConnectionEventArgs	Provides data for the ConnectionEvent .
	<p>ⓘ Note</p> <p>Added in version 1.2.0.</p>
	Constructor for internal use.
EventSignal	Clients can connect to the event signal to receive events, or disconnect from the event signal to stop receiving events.
	Constructor for internal use.

KeywordRecognitionEventArgs	<p>Class for keyword recognition event arguments.</p> <p>Constructor for internal use.</p>
KeywordRecognitionModel	Represents a keyword recognition model.
KeywordRecognitionResult	<p>Result of a keyword recognition operation.</p> <p>Constructor for internal use.</p>
KeywordRecognizer	A keyword recognizer.
NoMatchDetails	
PhraseListGrammar	<p>Class that allows runtime addition of phrase hints to aid in speech recognition.</p> <p>Phrases added to the recognizer are effective at the start of the next recognition, or the next time the speech recognizer must reconnect to the speech service.</p> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f0f8ff;"> <p>! Note</p> <p>Added in version 1.5.0.</p> </div>
PronunciationAssessmentConfig	<p>Constructor for internal use.</p> <p>Represents pronunciation assessment configuration</p> <div style="border: 1px solid #ccc; padding: 10px; background-color: #f0f8ff;"> <p>! Note</p> <p>Added in version 1.14.0.</p> </div>
PronunciationAssessmentPhonemeResult	<p>The configuration can be initialized in two ways:</p> <ul style="list-style-type: none"> from parameters: pass reference text, grading system, granularity, enable miscue and scenario id. from json: pass a json string <p>For the parameters details, see https://docs.microsoft.com/azure/cognitive-services/speech-service/rest-speech-to-text#pronunciation-assessment-parameters</p>
PronunciationAssessmentPhonemeResult	Contains phoneme level pronunciation assessment result

	Added in version 1.14.0.
PronunciationAssessmentResult	<p>Represents pronunciation assessment result.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>! Note</p> <p>Added in version 1.14.0.</p> </div> <p>The result can be initialized from a speech recognition result.</p>
PronunciationAssessmentWordResult	<p>Contains word level pronunciation assessment result</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>! Note</p> <p>Added in version 1.14.0.</p> </div>
PropertyCollection	<p>Class to retrieve or set a property value from a property collection.</p>
RecognitionEventArgs	<p>Provides data for the RecognitionEvent.</p> <p>Constructor for internal use.</p>
RecognitionResult	<p>Detailed information about the result of a recognition operation.</p> <p>Constructor for internal use.</p>
Recognizer	<p>Base class for different recognizers</p>
ResultFuture	<p>The result of an asynchronous operation.</p> <p>private constructor</p>
SessionEventArgs	<p>Base class for session event arguments.</p> <p>Constructor for internal use.</p>
SourceLanguageRecognizer	<p>A source language recognizer - standalone language recognizer, can be used for single language or continuous language detection.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>! Note</p> <p>Added in version 1.18.0.</p> </div>

SpeechConfig	<p>Class that defines configurations for speech / intent recognition and speech synthesis.</p> <p>The configuration can be initialized in different ways:</p> <ul style="list-style-type: none"> • from subscription: pass a subscription key and a region • from endpoint: pass an endpoint. Subscription key, AzureKeyCredential, or authorization token are optional. • from host: pass a host address. Subscription key or authorization token are optional. • from authorization token: pass an authorization token and a region
SpeechRecognitionCanceledEventArgs	<p>Class for speech recognition canceled event arguments.</p> <p>Constructor for internal use.</p>
SpeechRecognitionEventArgs	<p>Class for speech recognition event arguments.</p> <p>Constructor for internal use.</p>
SpeechRecognitionResult	<p>Base class for speech recognition results.</p> <p>Constructor for internal use.</p>
SpeechRecognizer	<p>A speech recognizer. If you need to specify source language information, please only specify one of these three parameters, language, source_language_config or auto_detect_source_language_config.</p>
SpeechSynthesisBookmarkEventArgs	<p>Class for speech synthesis bookmark event arguments.</p> <div style="background-color: #e0e0ff; padding: 10px; border-radius: 10px;"> <p>⚠ Note</p> <p>Added in version 1.16.0.</p> </div> <p>Constructor for internal use.</p>
SpeechSynthesisCancellationDetails	<p>Contains detailed information about why a result was canceled.</p>
SpeechSynthesisEventArgs	<p>Class for speech synthesis event arguments.</p> <p>Constructor for internal use.</p>
SpeechSynthesisResult	<p>Result of a speech synthesis operation.</p> <p>Constructor for internal use.</p>

[SpeechSynthesisVisemeEventArgs](#)

Class for speech synthesis viseme event arguments.

ⓘ Note

Added in version 1.16.0.

Constructor for internal use.

[SpeechSynthesisWordBoundaryEventArgs](#)

Class for speech synthesis word boundary event arguments.

ⓘ Note

Updated in version 1.21.0.

Constructor for internal use.

[SpeechSynthesizer](#)

A speech synthesizer.

[SyllableLevelTimingResult](#)

Contains syllable level timing result

ⓘ Note

Added in version 1.20.0.

[SynthesisVoicesResult](#)

Contains detailed information about the retrieved synthesis voices list.

ⓘ Note

Added in version 1.16.0.

Constructor for internal use.

[VoiceInfo](#)

Contains detailed information about the synthesis voice information.

ⓘ Note

Updated in version 1.17.0.

Constructor for internal use.

Enums

 [Expand table](#)

AudioStreamContainerFormat	Defines supported audio stream container format.
AudioStreamWaveFormat	Represents the format specified inside WAV container.
CancellationErrorCode	Defines error code in case that CancellationReason is Error.
CancellationReason	Defines the possible reasons a recognition result might be canceled.
NoMatchReason	Defines the possible reasons a recognition result might not be recognized.
OutputFormat	Output format.
ProfanityOption	Removes profanity (swearing), or replaces letters of profane words with stars.
PronunciationAssessmentGradingSystem	Defines the point system for pronunciation score calibration; default value is FivePoint.
PronunciationAssessmentGranularity	Defines the pronunciation evaluation granularity; default value is Phoneme.
PropertyId	Defines speech property ids.
ResultReason	Specifies the possible reasons a recognition result might be generated.
ServicePropertyChannel	Defines channels used to pass property settings to service.
SpeechSynthesisOutputFormat	Defines the possible speech synthesis output audio formats.
StreamStatus	Defines the possible status of audio data stream.
SynthesisVoiceGender	Defines the gender of synthesis voices
SynthesisVoiceType	Defines the type of synthesis voices

Speech to text REST API

Speech to text REST API is used for [fast transcription](#), [batch transcription](#) and [custom speech](#).

Important

Speech to text REST API version `2024-11-15` is the latest version that's generally available.

- [Speech to text REST API](#) version `2024-05-15-preview` will be retired on a date to be announced.
- Speech to text REST API `v3.0`, `v3.1`, `v3.2`, `3.2-preview.1`, and `3.2-preview.2` will be retired on March 31st, 2026.

For more information about upgrading, see the Speech to text REST API [v3.0 to v3.1](#), [v3.1 to v3.2](#), and [v3.2 to 2024-11-15](#) migration guides.

See the Speech to text REST API 2024-11-15 reference documentation

Use Speech to text REST API to:

- [Fast transcription](#): Transcribe audio files with returning results synchronously and much faster than real-time audio. Use the fast transcription API (`/speechtotext/transcriptions:transcribe`) in the scenarios that you need the transcript of an audio recording as quickly as possible with predictable latency, such as quick audio or video transcription or video translation.
- [Batch transcription](#): Transcribe audio files as a batch from multiple URLs or an Azure container. Use the batch transcription API (`/speechtotext/transcriptions:submit`) in the scenarios that you need to transcribe a large amount of audio in storage, such as a large number of files or a long audio file.
- [Custom speech](#): Upload your own data, test and train a custom model, compare accuracy between models, and deploy a model to a custom endpoint. Copy models to other subscriptions if you want colleagues to have access to a model that you built, or if you want to deploy a model to more than one region.

Speech to text REST API includes such features as:

- Request logs for each endpoint.
- Request the manifest of the models that you create, to set up on-premises containers.
- Upload data from Azure storage accounts by using a shared access signature (SAS) URI.
- Bring your own storage. Use your own storage accounts for logs, transcription files, and other data.

- Some operations support webhook notifications. You can register your webhooks where notifications are sent.

Fast transcription

The following operation groups are applicable for [fast transcription](#).

[] [Expand table](#)

Operation group	Description
Transcriptions	<p>Use Transcriptions - Transcribe to transcribe audio files.</p> <p>When you use fast transcription you send a single file per request. See Create a transcription for examples of how to create a transcription from a single audio file.</p>

Batch transcription

The following operation groups are applicable for [batch transcription](#).

[] [Expand table](#)

Operation group	Description
Models	<p>Use base models or custom models to transcribe audio files.</p> <p>You can use models with custom speech and batch transcription. For example, you can use a model trained with a specific dataset to transcribe audio files. See Train a model and custom speech model lifecycle for examples of how to train and manage custom speech models.</p>
Transcriptions	<p>Use Transcriptions - Submit to transcribe a large amount of audio in storage.</p> <p>When you use batch transcription you send multiple files per request or point to an Azure Blob Storage container with the audio files to transcribe. See Create a transcription for examples of how to create a transcription from multiple audio files.</p>
Web hooks	<p>Use web hooks to receive notifications about creation, processing, completion, and deletion events.</p> <p>You can use web hooks with custom speech and batch transcription. Web hooks apply to datasets, endpoints, evaluations, models, and transcriptions.</p>

Custom speech

The following operation groups are applicable for [custom speech](#).

[] [Expand table](#)

Operation group	Description
Datasets	<p>Use datasets to train and test custom speech models.</p> <p>For example, you can compare the performance of a custom speech trained with a specific dataset to the performance of a base model or custom speech model trained with a different dataset. See Upload training and testing datasets for examples of how to upload datasets.</p>
Endpoints	<p>Deploy custom speech models to endpoints.</p> <p>You must deploy a custom endpoint to use a custom speech model. See Deploy a model for examples of how to manage deployment endpoints.</p>
Evaluations	<p>Use evaluations to compare the performance of different models.</p> <p>For example, you can compare the performance of a custom speech model trained with a specific dataset to the performance of a base model or a custom model trained with a different dataset. See test recognition quality and test accuracy for examples of how to test and evaluate custom speech models.</p>
Models	<p>Use base models or custom models to transcribe audio files.</p> <p>You can use models with custom speech and batch transcription. For example, you can use a model trained with a specific dataset to transcribe audio files. See Train a model and custom speech model lifecycle for examples of how to train and manage custom speech models.</p>
Web hooks	<p>Use web hooks to receive notifications about creation, processing, completion, and deletion events.</p> <p>You can use web hooks with custom speech and batch transcription. Web hooks apply to datasets, endpoints, evaluations, models, and transcriptions.</p>

Related content

- [Create a custom speech project](#)
- [Get familiar with batch transcription](#)

Last updated on 11/08/2025

Speech to text REST API for short audio

05/25/2025

Use cases for the Speech to text REST API for short audio are limited. Use it only in cases where you can't use the [Speech SDK](#) or [fast transcription API](#).

Before you use the Speech to text REST API for short audio, consider the following limitations:

- Requests that use the REST API for short audio and transmit audio directly can contain no more than 60 seconds of audio. For pronunciation assessment, the audio duration should be no more than 30 seconds. The input [audio formats](#) are more limited compared to the [Speech SDK](#).
- The REST API for short audio returns only final results. It doesn't provide partial results.
- [Speech translation](#) isn't supported via REST API for short audio. You need to use the [Speech SDK](#).
- [Batch transcription](#) and [custom speech](#) aren't supported via REST API for short audio. You should always use the [Speech to text REST API](#) for batch transcription and custom speech.

Before you use the Speech to text REST API for short audio, understand that you need to complete a token exchange as part of authentication to access the service. For more information, see [Authentication](#).

Regions and endpoints

The endpoint for the REST API for short audio has this format:

```
https://<REGION_IDENTIFIER>.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1
```

Replace `<REGION_IDENTIFIER>` with the identifier that matches the [region](#) of your Speech resource.

 **Note**

For Azure Government and Microsoft Azure operated by 21Vianet endpoints, see [this article about sovereign clouds](#).

Audio formats

Audio is sent in the body of the HTTP `POST` request. It must be in one of the formats in this table:

[Expand table](#)

Format	Codec	Bit rate	Sample rate
WAV	PCM	256 kbps	16 kHz, mono
OGG	OPUS	256 kbps	16 kHz, mono

! Note

The preceding formats are supported through the REST API for short audio and WebSockets in the Speech service. The [Speech SDK](#) supports the WAV format with PCM codec as well as [other formats](#).

Request headers

This table lists required and optional headers for speech to text requests:

[Expand table](#)

Header	Description	Required or optional
<code>Ocp-Apim-Subscription-Key</code>	Your resource key for the Speech service.	Either this header or <code>Authorization</code> is required.
<code>Authorization</code>	An authorization token preceded by the word <code>Bearer</code> . For more information, see Authentication .	Either this header or <code>Ocp-Apim-Subscription-Key</code> is required.
<code>Pronunciation-Assessment</code>	Specifies the parameters for showing pronunciation scores in recognition results. These scores assess the pronunciation quality of speech input, with indicators like accuracy, fluency, and completeness. This parameter is a Base64-encoded JSON that contains multiple detailed parameters. To learn how to build this header, see Pronunciation assessment parameters .	Optional
<code>Content-type</code>	Describes the format and codec of the provided audio data. Accepted values are <code>audio/wav; codecs=audio/pcm; samplerate=16000</code> and <code>audio/ogg; codecs=opus</code> .	Required

Header	Description	Required or optional
Transfer-Encoding	Specifies that chunked audio data is being sent, rather than a single file. Use this header only if you're chunking audio data.	Optional
Expect	If you're using chunked transfer, send <code>Expect: 100-continue</code> . The Speech service acknowledges the initial request and awaits more data.	Required if you're sending chunked audio data.
Accept	If provided, it must be <code>application/json</code> . The Speech service provides results in JSON. Some request frameworks provide an incompatible default value. It's good practice to always include <code>Accept</code> .	Optional, but recommended.

Query parameters

These parameters might be included in the query string of the REST request.

! Note

You must append the language parameter to the URL to avoid receiving a 4xx HTTP error. For example, the language set to US English via the West US endpoint is:

```
https://westus.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?language=en-US.
```

[] Expand table

Parameter	Description	Required or optional
<code>language</code>	Identifies the spoken language that's being recognized. See Supported languages .	Required
<code>format</code>	Specifies the result format. Accepted values are <code>simple</code> and <code>detailed</code> . Simple results include <code>RecognitionStatus</code> , <code>DisplayText</code> , <code>Offset</code> , and <code>Duration</code> . Detailed responses include four different representations of display text. The default setting is <code>simple</code> .	Optional
<code>profanity</code>	Specifies how to handle profanity in recognition results. Accepted values are: <code>masked</code> , which replaces profanity with asterisks. <code>removed</code> , which removes all profanity from the result. <code>raw</code> , which includes profanity in the result.	Optional

Parameter	Description	Required or optional
	The default setting is <code>masked</code> .	

Pronunciation assessment parameters

This table lists required and optional parameters for pronunciation assessment:

[+] Expand table

Parameter	Description	Required or optional
<code>ReferenceText</code>	The text that the pronunciation is evaluated against.	Required
<code>GradingSystem</code>	The point system for score calibration. The <code>FivePoint</code> system gives a 0-5 floating point score, and <code>HundredMark</code> gives a 0-100 floating point score. Default: <code>FivePoint</code> .	Optional
<code>Granularity</code>	<p>The evaluation granularity. Accepted values are:</p> <ul style="list-style-type: none"> <code>Phoneme</code>, which shows the score on the full-text, word, and phoneme levels. <code>Word</code>, which shows the score on the full-text and word levels. <code>FullText</code>, which shows the score on the full-text level only. <p>The default setting is <code>Phoneme</code>.</p>	Optional
<code>Dimension</code>	<p>Defines the output criteria. Accepted values are:</p> <ul style="list-style-type: none"> <code>Basic</code>, which shows the accuracy score only. <code>Comprehensive</code>, which shows scores on more dimensions (for example, fluency score and completeness score on the full-text level, and error type on the word level). <p>To see definitions of different score dimensions and word error types, see Response properties. The default setting is <code>Basic</code>.</p>	Optional
<code>EnableMiscue</code>	Enables miscue calculation. With this parameter enabled, the pronounced words are compared to the reference text. They are marked with omission or insertion based on the comparison. Accepted values are <code>False</code> and <code>True</code> . The default setting is <code>False</code> .	Optional
<code>EnableProsodyAssessment</code>	Enables prosody assessment for your pronunciation evaluation. This feature assesses aspects like stress, intonation, speaking	Optional

Parameter	Description	Required or optional
	<p>speed, and rhythm. This feature provides insights into the naturalness and expressiveness of your speech.</p> <p>If this property is set to <code>True</code>, the <code>ProsodyScore</code> result value is returned.</p>	
<code>ScenarioId</code>	A GUID that indicates a customized point system.	Optional

Here's example JSON that contains the pronunciation assessment parameters:

JSON

```
{
  "ReferenceText": "Good morning.",
  "GradingSystem": "HundredMark",
  "Granularity": "Word",
  "Dimension": "Comprehensive",
  "EnableProsodyAssessment": "True"
}
```

The following sample code shows how to build the pronunciation assessment parameters into the `Pronunciation-Assessment` header:

C#

```
var pronAssessmentParamsJson = $"{{\"ReferenceText\":\"Good morning.\",\"GradingSystem\":\"HundredMark\", \"Granularity\":\"Word\", \"Dimension\":\"Comprehensive\", \"EnableProsodyAssessment\":\"True\"}}";
var pronAssessmentParamsBytes = Encoding.UTF8.GetBytes(pronAssessmentParamsJson);
var pronAssessmentHeader = Convert.ToString(pronAssessmentParamsBytes);
```

We strongly recommend streaming ([chunked transfer](#)) uploading while you're posting the audio data, which can significantly reduce the latency. To learn how to enable streaming, see the [sample code in various programming languages ↗](#).

!*Note*

For more information, see [pronunciation assessment](#).

Sample request

The following sample includes the host name and required headers. It's important to note that the service also expects audio data, which isn't included in this sample. As mentioned earlier, chunking is recommended but not required.

HTTP

```
POST speech/recognition/conversation/cognitiveservices/v1?language=en-US&format=detailed HTTP/1.1
Accept: application/json;text/xml
Content-Type: audio/wav; codecs=audio/pcm; samplerate=16000
Ocp-Apim-Subscription-Key: YOUR_RESOURCE_KEY
Host: westus.stt.speech.microsoft.com
Transfer-Encoding: chunked
Expect: 100-continue
```

To enable pronunciation assessment, you can add the following header. To learn how to build this header, see [Pronunciation assessment parameters](#).

HTTP

```
Pronunciation-Assessment: eyJSZWZlc...
```

HTTP status codes

The HTTP status code for each response indicates success or common errors.

[] Expand table

HTTP status code	Description	Possible reasons
100 Continue		The initial request is accepted. Proceed with sending the rest of the data. (This code is used with chunked transfer.)
200 OK		The request was successful. The response body is a JSON object.
400 Bad request		The language code wasn't provided, the language isn't supported, or the audio file is invalid (for example).
401 Unauthorized		A resource key or an authorization token is invalid in the specified region, or an endpoint is invalid.
403 Forbidden		A resource key or authorization token is missing.

Sample responses

Here's a typical response for `simple` recognition:

JSON

```
{  
    "RecognitionStatus": "Success",  
    "DisplayText": "Remind me to buy 5 pencils.",  
    "Offset": "1236645672289",  
    "Duration": "1236645672289"  
}
```

Here's a typical response for `detailed` recognition:

JSON

```
{  
    "RecognitionStatus": "Success",  
    "Offset": "1236645672289",  
    "Duration": "1236645672289",  
    "NBest": [  
        {  
            "Confidence": 0.9052885,  
            "Display": "What's the weather like?",  
            "ITN": "what's the weather like",  
            "Lexical": "what's the weather like",  
            "MaskedITN": "what's the weather like"  
        },  
        {  
            "Confidence": 0.92459863,  
            "Display": "what is the weather like",  
            "ITN": "what is the weather like",  
            "Lexical": "what is the weather like",  
            "MaskedITN": "what is the weather like"  
        }  
    ]  
}
```

Here's a typical response for recognition with pronunciation assessment:

JSON

```
{  
    "RecognitionStatus": "Success",  
    "Offset": 700000,  
    "Duration": 8400000,  
    "DisplayText": "Good morning.",  
    "SNR": 38.76819,  
    "NBest": [  
        {  
            "Confidence": 0.98503506,  
            "Lexical": "good morning",  
            "PronunciationAssessment": {  
                "Score": 0.98503506,  
                "Feedback": "Good pronunciation."  
            }  
        }  
    ]  
}
```

```
"ITN": "good morning",
"MaskedITN": "good morning",
"Display": "Good morning.",
"AccuracyScore": 100.0,
"FluencyScore": 100.0,
"ProsodyScore": 87.8,
"CompletenessScore": 100.0,
"PronScore": 95.1,
"Words": [
  {
    "Word": "good",
    "Offset": 700000,
    "Duration": 2600000,
    "Confidence": 0.0,
    "AccuracyScore": 100.0,
    "ErrorType": "None",
    "Feedback": {
      "Prosody": {
        "Break": {
          "ErrorTypes": [
            "None"
          ],
          "BreakLength": 0
        },
        "Intonation": {
          "ErrorTypes": [],
          "Monotone": {
            "Confidence": 0.0,
            "WordPitchSlopeConfidence": 0.0,
            "SyllablePitchDeltaConfidence": 0.91385907
          }
        }
      }
    }
  },
  {
    "Word": "morning",
    "Offset": 3400000,
    "Duration": 5700000,
    "Confidence": 0.0,
    "AccuracyScore": 100.0,
    "ErrorType": "None",
    "Feedback": {
      "Prosody": {
        "Break": {
          "ErrorTypes": [
            "None"
          ],
          "UnexpectedBreak": {
            "Confidence": 3.5294118e-08
          },
          "MissingBreak": {
            "Confidence": 1.0
          }
        },
        "BreakLength": 0
      }
    }
  }
]
```

```
        },
        "Intonation": {
            "ErrorTypes": [],
            "Monotone": {
                "Confidence": 0.0,
                "WordPitchSlopeConfidence": 0.0,
                "SyllablePitchDeltaConfidence": 0.91385907
            }
        }
    }
}
]
}
```

Response properties

Results are provided as JSON. The `simple` format includes the following top-level fields:

 Expand table

Property	Description
RecognitionStatus	Status, such as <code>Success</code> for successful recognition. See the next table.
DisplayText	The recognized text after capitalization, punctuation, inverse text normalization, and profanity masking. Present only on success. Inverse text normalization is conversion of spoken text to shorter forms, such as 200 for "two hundred" or "Dr. Smith" for "doctor smith."
Offset	The time (in 100-nanosecond units) at which the recognized speech begins in the audio stream.
Duration	The duration (in 100-nanosecond units) of the recognized speech in the audio stream.
SNR	The signal-to-noise ratio (SNR) of the recognized speech in the audio stream.

The `RecognitionStatus` field might contain these values:

 Expand table

Status	Description
Success	The recognition was successful, and the <code>DisplayText</code> field is present.

Status	Description
NoMatch	Speech was detected in the audio stream, but no words from the target language were matched. This status usually means that the recognition language is different from the language that the user is speaking.
InitialSilenceTimeout	The start of the audio stream contained only silence, and the service timed out while waiting for speech.
BabbleTimeout	The start of the audio stream contained only noise, and the service timed out while waiting for speech.
Error	The recognition service encountered an internal error and couldn't continue. Try again if possible.

ⓘ Note

If the audio consists only of profanity, and the `profanity` query parameter is set to `remove`, the service does not return a speech result.

The `detailed` format includes more forms of recognized results. When you're using the `detailed` format, `DisplayText` is provided as `Display` for each result in the `NBest` list.

The object in the `NBest` list can include:

[+] Expand table

Property	Description
<code>Confidence</code>	The confidence score of the entry, from 0.0 (no confidence) to 1.0 (full confidence).
<code>Lexical</code>	The lexical form of the recognized text: the actual words recognized.
<code>ITN</code>	The inverse-text-normalized (ITN) or canonical form of the recognized text, with phone numbers, numbers, abbreviations ("doctor smith" to "dr smith"), and other transformations applied.
<code>MaskedITN</code>	The ITN form with profanity masking applied, if requested.
<code>Display</code>	The display form of the recognized text, with punctuation and capitalization added. This parameter is the same as what <code>DisplayText</code> provides when the format is set to <code>simple</code> .
<code>AccuracyScore</code>	Pronunciation accuracy of the speech. Accuracy indicates how closely the phonemes match a native speaker's pronunciation. The accuracy score at the word and full-text levels is aggregated from the accuracy score at the phoneme level.

Property	Description
FluencyScore	Fluency of the provided speech. Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words.
ProsodyScore	Prosody of the given speech. Prosody indicates how natural the given speech is, including stress, intonation, speaking speed, and rhythm.
	To see definitions of prosody assessment results in details, see Result parameters .
CompletenessScore	Completeness of the speech, determined by calculating the ratio of pronounced words to reference text input.
PronScore	Overall score that indicates the pronunciation quality of the provided speech. This score is aggregated from AccuracyScore, FluencyScore, and CompletenessScore with weight.
ErrorType	Value that indicates whether a word is omitted, inserted, or badly pronounced, compared to ReferenceText. Possible values are None (meaning no error on this word), Omission, Insertion, and Mispronunciation.

Chunked transfer

Chunked transfer (`Transfer-Encoding: chunked`) can help reduce recognition latency. It allows the Speech service to begin processing the audio file while it's transmitted. The REST API for short audio doesn't provide partial or interim results.

The following code sample shows how to send audio in chunks. Only the first chunk should contain the audio file's header. `request` is an `HttpWebRequest` object that's connected to the appropriate REST endpoint. `audioFile` is the path to an audio file on disk.

C#

```
var request = (HttpWebRequest)HttpWebRequest.Create(requestUri);
request.SendChunked = true;
request.Accept = @"application/json;text/xml";
request.Method = "POST";
request.ProtocolVersion = HttpVersion.Version11;
request.Host = host;
request.ContentType = @"audio/wav; codecs=audio/pcm; samplerate=16000";
request.Headers["Ocp-Apim-Subscription-Key"] = "YOUR_RESOURCE_KEY";
request.AllowWriteStreamBuffering = false;

using (var fs = new FileStream(audioFile, FileMode.Open, FileAccess.Read))
{
    // Open a request stream and write 1,024-byte chunks in the stream one at a
    // time.
    byte[] buffer = null;
```

```

    int bytesRead = 0;
    using (var requestStream = request.GetRequestStream())
    {
        // Read 1,024 raw bytes from the input audio file.
        buffer = new Byte[checked((uint)Math.Min(1024, (int)fs.Length))];
        while ((bytesRead = fs.Read(buffer, 0, buffer.Length)) != 0)
        {
            requestStream.Write(buffer, 0, bytesRead);
        }

        requestStream.Flush();
    }
}

```

Authentication

Each request requires an authorization header. This table illustrates which headers are supported for each feature:

[] Expand table

Supported authorization header	Speech to text	Text to speech
Ocp-Apim-Subscription-Key	Yes	Yes
Authorization: Bearer	Yes	Yes

When you're using the `Ocp-Apim-Subscription-Key` header, only your resource key must be provided. For example:

HTTP

```
'Ocp-Apim-Subscription-Key': 'YourSpeechResourceKey'
```

When you're using the `Authorization: Bearer` header, you need to make a request to the `issueToken` endpoint. In this request, you exchange your resource key for an access token that's valid for 10 minutes.

Another option is to use Microsoft Entra authentication that also uses the `Authorization: Bearer` header, but with a token issued via Microsoft Entra ID. See [Use Microsoft Entra authentication](#).

How to get an access token

To get an access token, you need to make a request to the `issueToken` endpoint by using `Ocp-Apim-Subscription-Key` and your resource key.

The `issueToken` endpoint has this format:

HTTP

```
https://<REGION_IDENTIFIER>.api.cognitive.microsoft.com/sts/v1.0/issueToken
```

Replace `<REGION_IDENTIFIER>` with the identifier that matches the [region](#) of your Speech resource.

Use the following samples to create your access token request.

HTTP sample

This example is a simple HTTP request to get a token. Replace `YourSpeechResourceKey` with your resource key for the Speech service. If your Speech resource isn't in the West US region, replace the `Host` header with your region's host name.

HTTP

```
POST /sts/v1.0/issueToken HTTP/1.1
Ocp-Apim-Subscription-Key: YourSpeechResourceKey
Host: eastus.api.cognitive.microsoft.com
Content-type: application/x-www-form-urlencoded
Content-Length: 0
```

The body of the response contains the access token in JSON Web Token (JWT) format.

PowerShell sample

This example is a simple PowerShell script to get an access token. Replace `YourSpeechResourceKey` with your resource key for the Speech service. Make sure to use the correct endpoint for the region that matches your Speech resource. This example is currently set to West US.

PowerShell

```
$FetchTokenHeader = @{
    'Content-type'='application/x-www-form-urlencoded';
    'Content-Length'= '0';
    'Ocp-Apim-Subscription-Key' = 'YourSpeechResourceKey'
}
```

```
$OAuthToken = Invoke-RestMethod -Method POST -Uri  
https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken  
-Headers $FetchTokenHeader  
  
# show the token received  
$OAuthToken
```

cURL sample

cURL is a command-line tool available in Linux (and in the Windows Subsystem for Linux). This cURL command illustrates how to get an access token. Replace `YourSpeechResourceKey` with your resource key for the Speech service. Make sure to use the correct endpoint for the region that matches your Speech resource. This example is currently set to West US.

Console

```
curl -v -X POST \  
"https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken" \  
-H "Content-type: application/x-www-form-urlencoded" \  
-H "Content-Length: 0" \  
-H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey"
```

C# sample

This C# class illustrates how to get an access token. Pass your resource key for the Speech service when you instantiate the class. If your Speech resource isn't in the West US region, change the value of `FetchTokenUri` to match the region for your Speech resource.

C#

```
public class Authentication  
{  
    public static readonly string FetchTokenUri =  
        "https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken";  
    private string subscriptionKey;  
    private string token;  
  
    public Authentication(string subscriptionKey)  
    {  
        this.subscriptionKey = subscriptionKey;  
        this.token = FetchTokenAsync(FetchTokenUri, subscriptionKey).Result;  
    }  
  
    public string GetAccessToken()  
    {
```

```

        return this.token;
    }

    private async Task<string> FetchTokenAsync(string fetchUri, string
subscriptionKey)
    {
        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
subscriptionKey);
            UriBuilder uriBuilder = new UriBuilder(fetchUri);

            var result = await client.PostAsync(uriBuilder.Uri.AbsoluteUri, null);
            Console.WriteLine("Token Uri: {0}", uriBuilder.Uri.AbsoluteUri);
            return await result.Content.ReadAsStringAsync();
        }
    }
}

```

Python sample

Python

```

# Request module must be installed.
# Run pip install requests if necessary.
import requests

subscription_key = 'REPLACE_WITH_YOUR_KEY'

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

```

How to use an access token

The access token should be sent to the service as the `Authorization: Bearer <TOKEN>` header. Each access token is valid for 10 minutes. You can get a new token at any time, but to minimize network traffic and latency, we recommend using the same token for nine minutes.

Here's a sample HTTP request to the Speech to text REST API for short audio:

HTTP

```
POST /cognitiveservices/v1 HTTP/1.1
Authorization: Bearer YOUR_ACCESS_TOKEN
Host: westus.stt.speech.microsoft.com
Content-type: application/ssml+xml
Content-Length: 199
Connection: Keep-Alive

// Message body here...
```

Use Microsoft Entra authentication

To use Microsoft Entra authentication with the Speech to text REST API for short audio, you need to create an access token. The steps to obtain the access token consisting of Resource ID and Microsoft Entra access token are the same as when using the Speech SDK. Follow the steps here [Use Microsoft Entra authentication](#)

- ✓ Create an AI Foundry resource for Speech
- ✓ Configure the Speech resource for Microsoft Entra authentication
- ✓ Get a Microsoft Entra access token
- ✓ Get the Speech resource ID

After the resource ID and the Microsoft Entra access token were obtained, the actual access token can be constructed following this format:

HTTP

```
aad#YOUR_RESOURCE_ID#YOUR_MICROSOFT_ENTRA_ACCESS_TOKEN
```

You need to include the "aad#" prefix and the "#" (hash) separator between resource ID and the access token.

Here's a sample HTTP request to the Speech to text REST API for short audio:

HTTP

```
POST /cognitiveservices/v1 HTTP/1.1
Authorization: Bearer YOUR_ACCESS_TOKEN
Host: westus.stt.speech.microsoft.com
Content-type: application/ssml+xml
Content-Length: 199
Connection: Keep-Alive

// Message body here...
```

To learn more about Microsoft Entra access tokens, including token lifetime, visit [Access tokens in the Microsoft identity platform](#).

Related content

- [Fast transcription API](#)
- [Customize speech models](#)
- [Get familiar with batch transcription](#)

Text to speech REST API

The Speech service allows you to [convert text into synthesized speech](#) and [get a list of supported voices](#) for a region by using a REST API. In this article, you learn about authorization options, query options, how to structure a request, and how to interpret a response.

💡 Tip

Use cases for the text to speech REST API are limited. Use it only in cases where you can't use the [Speech SDK](#). For example, with the Speech SDK you can [subscribe to events](#) for more insights about the text to speech processing and results.

The text to speech REST API supports neural text to speech voices in many locales. Each available endpoint is associated with a region. An API key for the endpoint or region that you plan to use is required. Here are links to more information:

- For a complete list of voices, see [Language and voice support for the Speech service](#).
- For information about regional availability, see [Speech service supported regions](#).
- For Azure Government and Microsoft Azure operated by 21Vianet endpoints, see [this article about sovereign clouds](#).

ⓘ Important

Costs vary for standard voices and custom voices. For more information, see [text to speech pricing](#).

Before you use the text to speech REST API, understand that you need to complete a token exchange as part of authentication to access the service. For more information, see [Authentication](#).

Get a list of voices

You can use the `tts.speech.microsoft.com/cognitiveservices/voices/list` endpoint to get a full list of voices for a specific region or endpoint. Prefix the voices list endpoint with a region to get a list of voices for that region. For example, to get a list of voices for the `westus` region, use the <https://westus.tts.speech.microsoft.com/cognitiveservices/voices/list> endpoint. For a list of all supported regions, see the [regions](#) documentation.

ⓘ Note

[Voices and styles in preview](#) are only available in a subset of regions. For the current list of regions that support voices and styles in public preview, see the [Speech service regions table](#).

Request headers

This table lists required and optional headers for text to speech requests:

 Expand table

Header	Description	Required or optional
Ocp-Apim-Subscription-Key	Your Speech resource key.	Either this header or Authorization is required.
Authorization	An authorization token preceded by the word Bearer. For more information, see Authentication .	Either this header or Ocp-Apim-Subscription-Key is required.

Request body

A body isn't required for `GET` requests to this endpoint.

Sample request

This request requires only an authorization header:

HTTP

GET /cognitiveservices/voices/list HTTP/1.1

Host: westus.tts.speech.microsoft.com
Ocp-Apim-Subscription-Key: YOUR_RESOURCE_KEY

Here's an example curl command:

curl

```
curl --location --request GET  
'https://YOUR_RESOURCE_REGION.tts.speech.microsoft.com/cognitiveservices/voices/list' \  
--header 'Ocp-Apim-Subscription-Key: YOUR_RESOURCE_KEY'
```

Sample response

You should receive a response with a JSON body that includes all supported locales, voices, gender, styles, and other details. The `WordsPerMinute` property for each voice can be used to estimate the length of the output speech. This JSON example shows partial results to illustrate the structure of a response:

JSON

```
"Gender": "Female",
"Locale": "en-US",
"LocaleName": "English (United States)",
"StyleList": [
    "assistant",
    "chat",
    "customerservice",
    "newscast",
    "angry",
    "cheerful",
    "sad",
    "excited",
    "friendly",
    "terrified",
    "shouting",
    "unfriendly",
    "whispering",
    "hopeful"
],
"SampleRateHertz": "24000",
"VoiceType": "Neural",
>Status": "GA",
"ExtendedPropertyMap": {
    "IsHighQuality48K": "True"
},
"WordsPerMinute": "152"
},
// Redacted for brevity
{
    "Name": "Microsoft Server Speech Text to Speech Voice (en-US,
JennyMultilingualNeutral)",
    "DisplayName": "Jenny Multilingual",
    "LocalName": "Jenny Multilingual",
    "ShortName": "en-US-JennyMultilingualNeutral",
    "Gender": "Female",
    "Locale": "en-US",
    "LocaleName": "English (United States)",
    "SecondaryLocaleList": [
        "de-DE",
        "en-AU",
        "en-CA",
        "en-GB",
        "es-ES",
        "es-MX",
        "fr-CA",
        "fr-FR",
        "it-IT",
        "ja-JP",
        "ko-KR",
        "pt-BR",
        "zh-CN"
    ],
    "SampleRateHertz": "24000",
    "VoiceType": "Neural",
    "Status": "GA",
    "WordsPerMinute": "190"
},
// Redacted for brevity
{
    "Name": "Microsoft Server Speech Text to Speech Voice (ga-IE, OrlaNeutral)",
```

```

        "DisplayName": "Orla",
        "LocalName": "Orla",
        "ShortName": "ga-IE-OrlaNeural",
        "Gender": "Female",
        "Locale": "ga-IE",
        "LocaleName": "Irish (Ireland)",
        "SampleRateHertz": "24000",
        "VoiceType": "Neural",
        "Status": "GA",
        "WordsPerMinute": "139"
    },
    // Redacted for brevity
{
    "Name": "Microsoft Server Speech Text to Speech Voice (zh-CN, YunxiNeural)",
    "DisplayName": "Yunxi",
    "LocalName": "云希",
    "ShortName": "zh-CN-YunxiNeural",
    "Gender": "Male",
    "Locale": "zh-CN",
    "LocaleName": "Chinese (Mandarin, Simplified)",
    "StyleList": [
        "narration-relaxed",
        "embarrassed",
        "fearful",
        "cheerful",
        "disgruntled",
        "serious",
        "angry",
        "sad",
        "depressed",
        "chat",
        "assistant",
        "newscast"
    ],
    "SampleRateHertz": "24000",
    "VoiceType": "Neural",
    "Status": "GA",
    "RolePlayList": [
        "Narrator",
        "YoungAdultMale",
        "Boy"
    ],
    "WordsPerMinute": "293"
},
    // Redacted for brevity
]

```

HTTP status codes

The HTTP status code for each response indicates success or common errors.

[\[\] Expand table](#)

HTTP status code	Description	Possible reason
200	OK	The request was successful.
400	Bad request	A required parameter is missing, empty, or null. Or, the value passed to either a required or optional parameter is invalid. A common reason is a header that's too long.
401	Unauthorized	The request isn't authorized. Make sure your resource key or token is valid and in the correct region.
429	Too many requests	You exceeded the quota or rate of requests allowed for your resource.
502	Bad gateway	There's a network or server-side problem. This status might also indicate invalid headers.

Convert text to speech

The `cognitiveservices/v1` endpoint allows you to convert text to speech by using [Speech Synthesis Markup Language \(SSML\)](#).

Regions and endpoints

These regions are supported for text to speech through the REST API. Be sure to select the endpoint that matches your Speech resource region.

Standard voices

Use this table to determine *availability of neural voices* by region or endpoint:

[] [Expand table](#)

Region	Endpoint
Australia East	https://australiaeast.tts.speech.microsoft.com/cognitiveservices/v1
Brazil South	https://brazilsouth.tts.speech.microsoft.com/cognitiveservices/v1
Canada Central	https://canadacentral.tts.speech.microsoft.com/cognitiveservices/v1
Canada East	https://canadaeast.tts.speech.microsoft.com/cognitiveservices/v1
Central US	https://centralus.tts.speech.microsoft.com/cognitiveservices/v1
East Asia	https://eastasia.tts.speech.microsoft.com/cognitiveservices/v1
East US	https://eastus.tts.speech.microsoft.com/cognitiveservices/v1

Region	Endpoint
East US 2	https://eastus2.tts.speech.microsoft.com/cognitiveservices/v1
France Central	https://francecentral.tts.speech.microsoft.com/cognitiveservices/v1
Germany West Central	https://germanywestcentral.tts.speech.microsoft.com/cognitiveservices/v1
India Central	https://centralindia.tts.speech.microsoft.com/cognitiveservices/v1
Italy North	https://italynorth.tts.speech.microsoft.com/cognitiveservices/v1
Japan East	https://japaneast.tts.speech.microsoft.com/cognitiveservices/v1
Japan West	https://japanwest.tts.speech.microsoft.com/cognitiveservices/v1
Korea Central	https://koreacentral.tts.speech.microsoft.com/cognitiveservices/v1
North Central US	https://northcentralus.tts.speech.microsoft.com/cognitiveservices/v1
North Europe	https://northeurope.tts.speech.microsoft.com/cognitiveservices/v1
Norway East	https://norwayeast.tts.speech.microsoft.com/cognitiveservices/v1
Qatar Central	https://qatarcentral.tts.speech.microsoft.com/cognitiveservices/v1
South Africa North	https://southafricanorth.tts.speech.microsoft.com/cognitiveservices/v1
South Central US	https://southcentralus.tts.speech.microsoft.com/cognitiveservices/v1
Southeast Asia	https://southeastasia.tts.speech.microsoft.com/cognitiveservices/v1
Sweden Central	https://swedencentral.tts.speech.microsoft.com/cognitiveservices/v1
Switzerland North	https://switzerlandnorth.tts.speech.microsoft.com/cognitiveservices/v1
Switzerland West	https://switzerlandwest.tts.speech.microsoft.com/cognitiveservices/v1
UAE North	https://uaenorth.tts.speech.microsoft.com/cognitiveservices/v1
UK South	https://uksouth.tts.speech.microsoft.com/cognitiveservices/v1
UK West	https://ukwest.tts.speech.microsoft.com/cognitiveservices/v1
US Gov Arizona	https://usgovarizona.tts.speech.azure.us/cognitiveservices/v1
US Gov Virginia	https://usgovvirginia.tts.speech.azure.us/cognitiveservices/v1
West Central US	https://westcentralus.tts.speech.microsoft.com/cognitiveservices/v1
West Europe	https://westeurope.tts.speech.microsoft.com/cognitiveservices/v1
West US	https://westus.tts.speech.microsoft.com/cognitiveservices/v1
West US 2	https://westus2.tts.speech.microsoft.com/cognitiveservices/v1
West US 3	https://westus3.tts.speech.microsoft.com/cognitiveservices/v1

💡 Tip

For the current list of regions that support voices in preview, see the [Speech service regions table](#).

Custom voices

If you created a custom voice, use the endpoint that you created. You can also use the following endpoints. Replace `{deploymentId}` with the deployment ID for your custom voice model.

 Expand table

Region	Training	Deployment	Endpoint
Australia East	Yes	Yes	<code>https://australiaeast.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Brazil South	No	Yes	<code>https://brazilsouth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Canada Central	No	Yes	<code>https://canadacentral.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Central US	No	Yes	<code>https://centralus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
East Asia	No	Yes	<code>https://eastasia.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
East US	Yes	Yes	<code>https://eastus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
East US 2	Yes	Yes	<code>https://eastus2.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
France Central	No	Yes	<code>https://francecentral.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Germany West Central	No	Yes	<code>https://germanywestcentral.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
India Central	Yes	Yes	<code>https://centralindia.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Italy North	No	Yes	<code>https://italynorth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Japan East	Yes	Yes	<code>https://japaneast.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>
Japan West	No	Yes	<code>https://japanwest.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}</code>

Region	Training	Deployment	Endpoint
Korea Central	Yes	Yes	https://koreacentral.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
North Central US	No	Yes	https://northcentralus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
North Europe	Yes	Yes	https://northeurope.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
Norway East	No	Yes	https://norwayeast.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
South Africa North	No	Yes	https://southafricanorth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
South Central US	Yes	Yes	https://southcentralus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
Southeast Asia	Yes	Yes	https://southeastasia.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
Sweden Central	No	Yes	https://swedencentral.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
Switzerland North	No	Yes	https://switzerlandnorth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
Switzerland West	No	Yes	https://switzerlandwest.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
UAE North	No	Yes	https://uaenorth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
UK South	Yes	Yes	https://uksouth.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
West Central US	No	Yes	https://westcentralus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
West Europe	Yes	Yes	https://westeurope.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
West US	Yes	Yes	https://westus.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
West US 2	Yes	Yes	https://westus2.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}
West US 3	No	Yes	https://westus3.voice.speech.microsoft.com/cognitiveservices/v1?deploymentId={deploymentId}

ⓘ Note

The preceding regions are available for standard voice model hosting and real-time synthesis. Custom voice training is only available in some regions. But you can easily [copy a custom voice model](#) from these regions to other regions in the preceding list.

Long Audio API

The Long Audio API is available in multiple regions with unique endpoints:

[+] Expand table

Region	Endpoint
Australia East	https://australiaeast.customvoice.api.speech.microsoft.com
East US	https://eastus.customvoice.api.speech.microsoft.com
India Central	https://centralindia.customvoice.api.speech.microsoft.com
South Central US	https://southcentralus.customvoice.api.speech.microsoft.com
Southeast Asia	https://southeastasia.customvoice.api.speech.microsoft.com
UK South	https://uksouth.customvoice.api.speech.microsoft.com
West Europe	https://westeurope.customvoice.api.speech.microsoft.com

Request headers

This table lists required and optional headers for text to speech requests:

[+] Expand table

Header	Description	Required or optional
Authorization	An authorization token preceded by the word <code>Bearer</code> . For more information, see Authentication .	Required
Content-Type	Specifies the content type for the provided text. Accepted value: <code>application/ssml+xml</code> .	Required
X-Microsoft-OutputFormat	Specifies the audio output format. For a complete list of accepted values, see Audio outputs .	Required
User-Agent	The application name. The provided value must be fewer than 255 characters.	Required

Request body

If you're using a custom voice, the body of a request can be sent as plain text (ASCII or UTF-8). Otherwise, the body of each `POST` request is sent as [SSML](#). SSML allows you to choose the voice and language of the synthesized speech that the text to speech feature returns. For a complete list of supported voices, see [Language and voice support for the Speech service](#).

Sample request

This HTTP request uses SSML to specify the voice and language. If the body length is long, and the resulting audio exceeds 10 minutes, it's truncated to 10 minutes. In other words, the audio length can't exceed 10 minutes.

HTTP

```
POST /cognitiveservices/v1 HTTP/1.1

X-Microsoft-OutputFormat: riff-24khz-16bit-mono-pcm
Content-Type: application/ssml+xml
Host: westus.tts.speech.microsoft.com
Content-Length: <Length>
Authorization: Bearer [Base64 access_token]
User-Agent: <Your application name>

<speak version='1.0' xml:lang='en-US'><voice xml:lang='en-US' xml:gender='Male'
name='en-US-ChristopherNeural'>
    I'm excited to try text to speech!
</voice></speak>
```

* For the Content-Length, you should use your own content length. In most cases, this value is calculated automatically.

HTTP status codes

The HTTP status code for each response indicates success or common errors:

[\[+\] Expand table](#)

HTTP status code	Description	Possible reason
200	OK	The request was successful. The response body is an audio file.
400	Bad request	A required parameter is missing, empty, or null. Or, the value passed to either a required or optional parameter is invalid. A common reason is a header that's too long.
401	Unauthorized	The request isn't authorized. Make sure your Speech resource key or token is valid and in the correct region.
415	Unsupported media type	It's possible that the wrong <code>Content-Type</code> value was provided. <code>Content-Type</code> should be set to <code>application/ssml+xml</code> .

HTTP status code	Description	Possible reason
429	Too many requests	You exceeded the quota or rate of requests allowed for your resource.
502	Bad gateway	There's a network or server-side problem. This status might also indicate invalid headers.
503	Service Unavailable	There's a server-side problem for various reasons.

If the HTTP status is `200 OK`, the body of the response contains an audio file in the requested format. This file can be played as it's transferred, saved to a buffer, or saved to a file.

Audio outputs

The supported streaming and nonstreaming audio formats are sent in each request as the `X-Microsoft-OutputFormat` header. Each format incorporates a bit rate and encoding type. The Speech service supports 48-kHz, 24-kHz, 16-kHz, and 8-kHz audio outputs. Each standard voice model is available at 24kHz and high-fidelity 48kHz.

Streaming

```
amr-wb-16000hz
audio-16khz-16bit-32kbps-mono-opus
audio-16khz-32kbitrate-mono-mp3
audio-16khz-64kbitrate-mono-mp3
audio-16khz-128kbitrate-mono-mp3
audio-24khz-16bit-24kbps-mono-opus
audio-24khz-16bit-48kbps-mono-opus
audio-24khz-48kbitrate-mono-mp3
audio-24khz-96kbitrate-mono-mp3
audio-24khz-160kbitrate-mono-mp3
audio-48khz-96kbitrate-mono-mp3
audio-48khz-192kbitrate-mono-mp3
g722-16khz-64kbps
ogg-16khz-16bit-mono-opus
ogg-24khz-16bit-mono-opus
ogg-48khz-16bit-mono-opus
raw-8khz-8bit-mono-alaw
raw-8khz-8bit-mono-mulaw
raw-8khz-16bit-mono-pcm
raw-16khz-16bit-mono-pcm
raw-16khz-16bit-mono-truesilk
raw-22050hz-16bit-mono-pcm
raw-24khz-16bit-mono-pcm
raw-24khz-16bit-mono-truesilk
raw-44100hz-16bit-mono-pcm
```

```
raw-48khz-16bit-mono-pcm  
webm-16khz-16bit-mono-opus  
webm-24khz-16bit-24kbps-mono-opus  
webm-24khz-16bit-mono-opus
```

ⓘ Note

If you select 48kHz output format, the high-fidelity voice model with 48kHz will be invoked accordingly. The sample rates other than 24kHz and 48kHz can be obtained through upsampling or downsampling when synthesizing, for example, 44.1kHz is downsampled from 48kHz.

If your selected voice and output format have different bit rates, the audio is resampled as necessary. You can decode the `ogg-24khz-16bit-mono-opus` format by using the [Opus codec](#).

Authentication

Each request requires an authorization header. This table illustrates which headers are supported for each feature:

[+] Expand table

Supported authorization header	Speech to text	Text to speech
<code>Ocp-Apim-Subscription-Key</code>	Yes	Yes
<code>Authorization: Bearer</code>	Yes	Yes

When you're using the `Ocp-Apim-Subscription-Key` header, only your resource key must be provided. For example:

HTTP

```
'Ocp-Apim-Subscription-Key': 'YourSpeechResourceKey'
```

When you're using the `Authorization: Bearer` header, you need to make a request to the `issueToken` endpoint. In this request, you exchange your resource key for an access token that's valid for 10 minutes.

Another option is to use Microsoft Entra authentication that also uses the `Authorization: Bearer` header, but with a token issued via Microsoft Entra ID. See [Use Microsoft Entra authentication](#).

How to get an access token

To get an access token, you need to make a request to the `issueToken` endpoint by using `Ocp-Apim-Subscription-Key` and your resource key.

The `issueToken` endpoint has this format:

HTTP

```
https://<REGION_IDENTIFIER>.api.cognitive.microsoft.com/sts/v1.0/issueToken
```

Replace <REGION_IDENTIFIER> with the identifier that matches the [region](#) of your Speech resource.

Use the following samples to create your access token request.

HTTP sample

This example is a simple HTTP request to get a token. Replace `YourSpeechResourceKey` with your resource key for the Speech service. If your Speech resource isn't in the West US region, replace the `Host` header with your region's host name.

HTTP

```
POST /sts/v1.0/issueToken HTTP/1.1
Ocp-Apim-Subscription-Key: YourSpeechResourceKey
Host: eastus.api.cognitive.microsoft.com
Content-type: application/x-www-form-urlencoded
Content-Length: 0
```

The body of the response contains the access token in JSON Web Token (JWT) format.

PowerShell sample

This example is a simple PowerShell script to get an access token. Replace `YourSpeechResourceKey` with your resource key for the Speech service. Make sure to use the correct endpoint for the region that matches your Speech resource. This example is currently set to West US.

PowerShell

```
$FetchTokenHeader = @{
    'Content-type'='application/x-www-form-urlencoded';
    'Content-Length'= '0';
    'Ocp-Apim-Subscription-Key' = 'YourSpeechResourceKey'
}

$OAuthToken = Invoke-RestMethod -Method POST -Uri
https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken
-Headers $FetchTokenHeader

# show the token received
$OAuthToken
```

cURL sample

cURL is a command-line tool available in Linux (and in the Windows Subsystem for Linux). This cURL command illustrates how to get an access token. Replace `YourSpeechResourceKey` with your resource key

for the Speech service. Make sure to use the correct endpoint for the region that matches your Speech resource. This example is currently set to West US.

Console

```
curl -v -X POST \
"https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken" \
-H "Content-type: application/x-www-form-urlencoded" \
-H "Content-Length: 0" \
-H "Ocp-Apim-Subscription-Key: YourSpeechResourceKey"
```

C# sample

This C# class illustrates how to get an access token. Pass your resource key for the Speech service when you instantiate the class. If your Speech resource isn't in the West US region, change the value of `FetchTokenUri` to match the region for your Speech resource.

C#

```
public class Authentication
{
    public static readonly string FetchTokenUri =
        "https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken";
    private string subscriptionKey;
    private string token;

    public Authentication(string subscriptionKey)
    {
        this.subscriptionKey = subscriptionKey;
        this.token = FetchTokenAsync(FetchTokenUri, subscriptionKey).Result;
    }

    public string GetAccessToken()
    {
        return this.token;
    }

    private async Task<string> FetchTokenAsync(string fetchUri, string subscriptionKey)
    {
        using (var client = new HttpClient())
        {
            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);
            UriBuilder uriBuilder = new UriBuilder(fetchUri);

            var result = await client.PostAsync(uriBuilder.Uri.AbsoluteUri, null);
            Console.WriteLine("Token Uri: {0}", uriBuilder.Uri.AbsoluteUri);
            return await result.Content.ReadAsStringAsync();
        }
    }
}
```

Python sample

Python

```
# Request module must be installed.  
# Run pip install requests if necessary.  
import requests  
  
subscription_key = 'REPLACE_WITH_YOUR_KEY'  
  
def get_token(subscription_key):  
    fetch_token_url = 'https://eastus.api.cognitive.microsoft.com/sts/v1.0/issueToken'  
    headers = {  
        'Ocp-Apim-Subscription-Key': subscription_key  
    }  
    response = requests.post(fetch_token_url, headers=headers)  
    access_token = str(response.text)  
    print(access_token)
```

How to use an access token

The access token should be sent to the service as the `Authorization: Bearer <TOKEN>` header. Each access token is valid for 10 minutes. You can get a new token at any time, but to minimize network traffic and latency, we recommend using the same token for nine minutes.

Here's a sample HTTP request to the Speech to text REST API for short audio:

HTTP

```
POST /cognitiveservices/v1 HTTP/1.1  
Authorization: Bearer YOUR_ACCESS_TOKEN  
Host: westus.stt.speech.microsoft.com  
Content-type: application/ssml+xml  
Content-Length: 199  
Connection: Keep-Alive  
  
// Message body here...
```

Use Microsoft Entra authentication

To use Microsoft Entra authentication with the Speech to text REST API for short audio, you need to create an access token. The steps to obtain the access token consisting of Resource ID and Microsoft Entra access token are the same as when using the Speech SDK. Follow the steps here [Use Microsoft Entra authentication](#)

- ✓ Create an AI Foundry resource for Speech
- ✓ Configure the Speech resource for Microsoft Entra authentication
- ✓ Get a Microsoft Entra access token
- ✓ Get the Speech resource ID

After the resource ID and the Microsoft Entra access token were obtained, the actual access token can be constructed following this format:

HTTP

```
aad#YOUR_RESOURCE_ID#YOUR_MICROSOFT_ENTRA_ACCESS_TOKEN
```

You need to include the "aad#" prefix and the "#" (hash) separator between resource ID and the access token.

Here's a sample HTTP request to the Speech to text REST API for short audio:

HTTP

```
POST /cognitiveservices/v1 HTTP/1.1
Authorization: Bearer YOUR_ACCESS_TOKEN
Host: westus.stt.speech.microsoft.com
Content-type: application/ssml+xml
Content-Length: 199
Connection: Keep-Alive

// Message body here...
```

To learn more about Microsoft Entra access tokens, including token lifetime, visit [Access tokens in the Microsoft identity platform](#).

Next steps

- [Create a free Azure account ↗](#)
- [Get started with custom voice](#)
- [Batch synthesis](#)

Last updated on 10/21/2025

Data Plane

Article • 06/12/2024

REST Operation Groups

 Expand table

Operation Group
Base Models
Consents
Endpoints
Models
Operations
Personal Voices
Projects
Training Sets

Data Plane

Article • 05/07/2024

REST Operation Groups

 Expand table

Operation Group
Batch Syntheses
Operations

Data Plane

Article • 08/12/2024

REST Operation Groups

 Expand table

Operation Group
Batch Syntheses
Operations

How to use the online transcription editor

09/12/2025

ⓘ Important

Online transcription editor in Azure AI Speech will be retired on December 15, 2025. You won't be able to use the online transcription editor after this date. To avoid the loss of any edits, export any data that is in the editor to a training and testing dataset before the retirement on December 15, 2025.

This change doesn't affect other Azure AI Speech capabilities such as [speech to text](#) (including no change to speaker diarization), [text to speech](#), and [speech translation](#).

The online transcription editor allows you to create or edit audio + human-labeled transcriptions for custom speech. The main use cases of the editor are as follows:

- You only have audio data, but want to build accurate audio + human-labeled datasets from scratch to use in model training.
- You already have audio + human-labeled datasets, but there are errors or defects in the transcription. The editor allows you to quickly modify the transcriptions to get best training accuracy.

The only requirement to use the transcription editor is to have audio data uploaded, with or without corresponding transcriptions.

You can find the **Editor** tab next to the **Training and testing dataset** tab on the main **Speech datasets** page.

The screenshot shows the 'Speech datasets' page in the Azure AI Speech Studio. On the left, there's a sidebar with options: 'Custom Speech', 'Contoso project' (selected), 'English (United States)', 'Speech datasets' (selected), 'Train custom models', 'Test models', and 'Deploy models'. The main area has a breadcrumb navigation: 'Speech Studio > Custom Speech > Contoso project > Editor'. Below the breadcrumb is a section titled 'Training and testing dataset' with a sub-section 'Editor' (underlined). A note says 'Import your uploaded datasets here to quickly modify, annotate, or reassemble the files, and export back for further training and testing purposes.' Below this are buttons for 'Import data', 'Export to Training and testing dataset', 'Download', 'Edit', and 'Delete'. A table lists datasets with columns: Name, Description, Type, Last edited, Quantity, and Status. Two rows are shown: 'From audio' (From audio only, Audio + transcript, 5/9/2022 10:46 PM, 00:59s, Succeeded) and 'From audio and transcript' (Import with transcript, Audio + transcript, 5/9/2022 6:30 PM, 00:42s, Succeeded).

Name	Description	Type	Last edited	Quantity	Status
From audio	From audio only	Audio + transcript	5/9/2022 10:46 PM	00:59s	Succeeded
From audio and transcript	Import with transcript	Audio + transcript	5/9/2022 6:30 PM	00:42s	Succeeded

Datasets in the **Training and testing dataset** tab can't be updated. You can import a copy of a training or testing dataset to the **Editor** tab, add or edit human-labeled transcriptions to match the audio, and then export the edited dataset to the **Training and testing dataset** tab. Also note that you can't use a dataset that's in the Editor to train or test a model.

Import datasets to the Editor

To import a dataset to the Editor, follow these steps:

1. Sign in to the [Speech Studio](#).
2. Select **Custom speech** > Your project name > **Speech datasets** > **Editor**.
3. Select **Import data**
4. Select datasets. You can select audio data only, audio + human-labeled data, or both. For audio-only data, you can use the default models to automatically generate machine transcription after importing to the editor.
5. Enter a name and description for the new dataset, and then select **Next**.
6. Review your settings, and then select **Import and close** to kick off the import process.

After data is successfully imported, you can select datasets and start editing.

! Note

You can also select a dataset from the main **Speech datasets** page and export them to the Editor. Select a dataset and then select **Export to Editor**.

Edit transcription to match audio

Once a dataset is imported to the Editor, you can start editing the dataset. You can add or edit human-labeled transcriptions to match the audio as you hear it. You don't edit any audio data.

To edit a dataset's transcription in the Editor, follow these steps:

1. Sign in to the [Speech Studio](#).
2. Select **Custom speech** > Your project name > **Speech datasets** > **Editor**.
3. Select the link to a dataset by name.
4. From the **Audio + text files** table, select the link to an audio file by name.
5. After you make edits, select **Save**.

If there are multiple files in the dataset, you can select **Previous** and **Next** to move from file to file. Edit and save changes to each file as you go.

The detail page lists all the segments in each audio file, and you can select the desired utterance. For each utterance, you can play and compare the audio with the corresponding transcription. Edit the transcriptions if you find any insertion, deletion, or substitution errors. For more information about word error types, see [Test model quantitatively](#).

Export datasets from the Editor

Datasets in the Editor can be exported to the **Training and testing dataset** tab, where they can be used to train or test a model.

To export datasets from the Editor, follow these steps:

1. Sign in to the [Speech Studio](#).
2. Select **Custom speech** > Your project name > **Speech datasets** > **Editor**.
3. Select the link to a dataset by name.
4. Select one or more rows from the **Audio + text files** table.
5. Select **Export** to export all of the selected files as one new dataset.

The files are exported as a new dataset, and don't affect or replace other training or testing datasets.

Next steps

- [Test recognition quality](#)
- [Train your model](#)
- [Deploy your model](#)

Migrate code from version 2024-11-15 to version 2025-10-15

Use the speech to text REST API for [fast transcription](#), [batch transcription](#), and [custom speech](#). This article describes changes from version 2024-11-15 to version 2025-10-15.

Important

Speech to text REST API version 2025-10-15 is the latest version that's generally available.

- [Speech to text REST API](#) version 2024-05-15-preview will be retired on a date to be announced.
- Speech to text REST API v3.0, v3.1, v3.2, 3.2-preview.1, and 3.2-preview.2 will be retired on March 31, 2026.

For more information about upgrading, see the Speech to text REST API [v3.0 to v3.1](#), [v3.1 to v3.2](#), and [v3.2 to 2024-11-15](#) migration guides.

To summarize the changes in this version:

- The Transcribe API has new features enhanced mode, and phrase list.
- The Projects API returns (absent in version 2024-11-15), and has some changes.

Transcription API changes

Request structure

- New endpoint:

```
POST <your_endpoint>/speechtotext/transcriptions:transcribe?api-version=2025-10-15
```

- Headers and form data:
 - Content-Type: multipart/form-data
 - Ocp-Apim-Subscription-Key: \$KEY
 - Form fields: definition, audio

Example:

Bash

```
curl --request POST \
--url '<your_endpoint>/speechtotext/transcriptions:transcribe?api-version=2025-10-15' \
--header 'Content-Type: multipart/form-data' \
--header 'Ocp-Apim-Subscription-Key: $KEY' \
--form 'definition=$DEFINITION' \
--form 'audio=@C:\workspace\audios\test.wav'
```

Definition object updates

- **Removed:**
 - "models" dictionary (no longer in request definition)
- **Added:**
 - "disfluencyRemoval" (boolean): Removes filler words (such as "um" and "uh")
 - "phraseList": Now supports `biasingWeight` for recognition bias tuning
 - "enhancedMode" object includes:
 - `enabled` (boolean)
 - `task` (such as `"translate"`)
 - `targetLanguage` (such as `"ko"`)
 - `prompt` (array of instructions or lexical boosts)

Example:

JSON

```
{
  "locales": ["en-US"],
  "profanityFilterMode": "Masked",
  "diarization": {
    "enabled": true,
    "maxSpeakers": 6
  },
  "channels": [0],
  "disfluencyRemoval": true,
  "enhancedMode": {
    "enabled": true,
    "task": "translate",
    "targetLanguage": "ko",
    "prompt": [
      "Provide lexical output",
      "Boost the terms: CONTOSO, AAZZ; Replace '50cents' to '50-Cents'"
    ]
  },
  "phraseList": {
    "phrases": ["Kenichi Kumatani", "John McDonough", "Bhiksha Raj"],
    "biasingWeight": 1.6
  }
}
```

```
}
```

Result structure

- **Channel-based output:**
 - Results are organized per channel
- **Phrase segmentation:**
 - Each phrase includes channel, start and end time, speaker, text, and word-level confidence

Projects API changes

New features

- **Foundry project name:**
 - New property: `foundryProjectName` in Create, Get, Update, List APIs
- **Project creation:**
 - Projects are created through Azure Resource Manager (ARM) conventions
 - `locale` is now required for custom speech projects

Example:

```
POST {endpoint}/speechtotext/projects?api-version=2025-10-15
Headers:
  Ocp-Apim-Subscription-Key: <YOUR_SUBSCRIPTION_KEY>
  Content-Type: application/json
Body:
{
  "locale": "en-US",
  "displayName": "My speech project",
  "foundryProjectName": "MyFoundrySpeechProject"
}
```

Project listing and filtering

- **Filter by Foundry project name:**

```
GET {endpoint}/speechtotext/projects?filter=foundryProjectName eq  
'MyFoundrySpeechProject'&api-version=2025-10-15
```

Next steps

- [Speech to text REST API](#)
- [Speech to text REST API 2025-10-15 reference documentation](#)

Last updated on 11/08/2025

Migrate code from v3.2 to version 2024-11-15

08/07/2025

The Speech to text REST API is used for [fast transcription](#), [batch transcription](#), and [custom speech](#). This article describes changes from version 3.2 to version 2024-11-15.

Important

Speech to text REST API version `2024-11-15` is the latest version that's generally available.

- [Speech to text REST API](#) version `2024-05-15-preview` will be retired on a date to be announced.
- Speech to text REST API `v3.0`, `v3.1`, `v3.2`, `3.2-preview.1`, and `3.2-preview.2` will be retired on March 31st, 2026.

For more information about upgrading, see the Speech to text REST API [v3.0 to v3.1](#), [v3.1 to v3.2](#), and [v3.2 to 2024-11-15](#) migration guides.

Base path

Custom speech API switched from a path based versioning scheme to a query parameter based scheme in alignment with general Azure API versioning schemes. This required changes to the used base path. Update path from `/speechtotext/v3.2` to `/speechtotext` and append API version with `?api-version=2024-11-15` to all requests.

Datasets

The `email` property and the connected email notification process is removed from the API.

The `duration` property in dataset responses is renamed from `duration` to `durationMilliseconds` and are now a plain number instead of an ISO8601 formatted string (P1D2H3M4S...) to further simplify processing.

The query parameter `sasValidityInSeconds` is renamed to `sasLifetimeMinutes` for getting files. Usage is only allowed for an account with BYOS disabled. For BYOS enabled accounts, SAS URLs aren't returned.

The `project` property is removed in creation requests.

Models

Removed the `text` property in a model creation request. The alternative is to create a dataset with the text content and create a dataset first, which then is later on used for model creation.

The `email` property and the connected email notification process is removed from the API.

The query parameter `sasValidityInSeconds` is renamed to `sasLifetimeMinutes` for getting files. Usage is only allowed for an account with BYOS (bring your own storage) disabled. For BYOS enabled accounts, SAS URLs aren't returned.

The `GET models/id/manifest` operation now always requires a nonzero SAS lifetime. The corresponding `sasValidityInSeconds` property is renamed to `sasLifetimeMinutes`.

The `project` property is removed in creation requests.

Evaluations

The query parameter `sasValidityInSeconds` is renamed to `sasLifetimeMinutes` for getting files. Usage is only allowed for an account with BYOS disabled. For BYOS enabled accounts, SAS URLs aren't returned.

The `project` property is removed in creation requests

The `email` property and the connected email notification process is removed from the API.

Endpoints

The API to retrieve and delete log files of endpoint logs is removed. Custom speech now supports BYOS (bring your own storage). Only accounts with BYOS enabled can enable logging on model endpoints. This offers full manageability of log files on customer storage instead of a proxy API.

Removed support for `timeToLive` in endpoint creations.

Removed the `text` property in an endpoint creation request. The alternative is to create a dataset with the text content and create a dataset first, which then is later on used for model creation. This model can then be used to create an endpoint.

Endpoint links now only return endpoint of websocket connection, used for SDK.

The `project` property is removed in creation requests.

The `email` property and the connected email notification process is removed from the API.

Transcriptions

Removed the top-level `diarizationEnabled` property of a transcription. The diarization configuration is simplified to `"diarization": {"maxSpeakers": 2,"enabled": true}`. The `maxSpeakers` property is optional and defaults to 2. The `enabled` property is required for diarization.

Transcription creation: `timeToLive` renamed to `timeToLiveHours` including a format change from ISO8601 formatted string to a simple int (number of hours).

The `duration` property in transcription responses is renamed from `duration` to `durationMilliseconds` and are now a plain number instead of an ISO8601 formatted string (P1D2H3M4S...) to further simplify processing. Transcription result files have this property added for consistency with API.

The query parameter `sasValidityInSeconds` is renamed to `sasLifetimeMinutes` for getting files. Usage is only allowed for an account with BYOS disabled. For BYOS enabled accounts, SAS URLs aren't returned.

The `project` property is removed in creation requests.

The `email` property and the connected email notification process is removed from the API.

Projects

The projects API is removed.

Next steps

- [Speech to text REST API](#)
- [Speech to text REST API 2024-11-15 reference documentation](#)

Migrate code from v3.1 to v3.2 of the REST API

The Speech to text REST API is used for [fast transcription](#), [batch transcription](#), and [custom speech](#). This article describes changes from version 3.1 to 3.2.

Important

Speech to text REST API version `2024-11-15` is the latest version that's generally available.

- [Speech to text REST API](#) version `2024-05-15-preview` will be retired on a date to be announced.
- Speech to text REST API `v3.0`, `v3.1`, `v3.2`, `3.2-preview.1`, and `3.2-preview.2` will be retired on March 31st, 2026.

For more information about upgrading, see the Speech to text REST API [v3.0 to v3.1](#), [v3.1 to v3.2](#), and [v3.2 to 2024-11-15](#) migration guides.

Base path

You must update the base path in your code from `/speechtotext/v3.1` to `/speechtotext/v3.2`.

For example, to get base models in the `eastus` region, use

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/models/base` instead of

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/models/base`.

For more information, see [Operation IDs](#) later in this guide.

Batch transcription

Important

New pricing is in effect for batch transcription via Speech to text REST API v3.2. For more information, see the [pricing guide](#).

Backwards compatibility limitations

Don't use Speech to text REST API v3.0 or v3.1 to retrieve a transcription created via Speech to text REST API v3.2. You might see an error message such as: "The API version can't be used to

access this transcription. Use API version v3.2 or higher."

Language identification mode

The `LanguageIdentificationMode` is added to `LanguageIdentificationProperties` as sibling of `candidateLocales` and `speechModelMapping`. The modes available for language identification are `Continuous` or `Single`. Continuous language identification is the default. For more information, see [Language identification](#).

Whisper models

Azure AI Speech now supports OpenAI's Whisper model via Speech to text REST API v3.2. To learn more, check out the [Create a batch transcription](#) guide.

(!) Note

Azure OpenAI in Azure AI Foundry Models also supports OpenAI's Whisper model for speech to text with a synchronous REST API. To learn more, check out the [quickstart](#). Check out [What is the Whisper model?](#) to learn more about when to use Azure AI Speech vs. Azure OpenAI in Azure AI Foundry Models.

Custom speech

(i) Important

You'll be charged for custom speech model training if the base model was created on October 1, 2023 and later. You're not charged for training if the base model was created prior to October 2023. For more information, see [Azure AI Speech pricing ↗](#).

To programmatically determine whether a model was created before or after October 1, 2023, use the `chargedForAdaptation` property that's [new in version 3.2](#).

Custom display text formatting

To support model adaptation with [custom display text formatting](#) data, the `Datasets_Create` operation supports the `OutputFormatting` data kind. For more information, see [upload datasets](#).

Added a definition for `OutputFormatType` with `Lexical` and `Display` enum values.

JSON

```
"OutputFormatType": {  
    "title": "OutputFormatType",  
    "enum": [  
        "Lexical",  
        "Display"  
    ],  
    "type": "string",  
    "x-ms-enum": {  
        "name": "OutputFormatType",  
        "modelAsString": true,  
        "values": [  
            {  
                "value": "Lexical",  
                "description": "Model provides the transcription output without  
formatting."  
            },  
            {  
                "value": "Display",  
                "description": "Model supports display formatting transcriptions  
output or endpoints."  
            }  
        ]  
    }  
},
```

The `OutputFormattingData` enum value is added to `FileKind` (type of input data).

The `supportedOutputFormat` property is added to `BaseModelFeatures`. This property is within the `BaseModel` definition.

JSON

```
"BaseModelFeatures": {  
    "title": "BaseModelFeatures",  
    "description": "Features supported by the model.",  
    "type": "object",  
    "allOf": [  
        {  
            "$ref": "#/definitions/SharedModelFeatures"  
        }  
    ],  
    "properties": {  
        "supportsAdaptationsWith": {  
            "description": "Supported dataset kinds to adapt the model.",  
            "type": "array",  
            "items": {  
                "$ref": "#/definitions/DatasetKind"  
            },  
            "readOnly": true  
        },  
        "supportedOutputFormat": {  
            "type": "string",  
            "x-ms-enum": {  
                "name": "OutputFormatType",  
                "modelAsString": true,  
                "values": [  
                    {  
                        "value": "Lexical",  
                        "description": "Model provides the transcription output without  
formatting."  
                    },  
                    {  
                        "value": "Display",  
                        "description": "Model supports display formatting transcriptions  
output or endpoints."  
                    }  
                ]  
            }  
        }  
    }  
},
```

```
        "description": "Supported output formats.",
        "type": "array",
        "items": {
            "$ref": "#/definitions/OutputFormatType"
        },
        "readOnly": true
    }
}
},
```

Charge for adaptation

The `chargeForAdaptation` property is added to `BaseModelProperties`. This property is within the `BaseModel` definition.

Important

You'll be charged for custom speech model training if the base model was created on October 1, 2023 and later. You're not charged for training if the base model was created prior to October 2023. For more information, see [Azure AI Speech pricing](#).

If the value of `chargeForAdaptation` is `true`, you're charged for training the model. If the value is `false`, you're charged for training the model. Use the `chargeForAdaptation` property instead of the created date to programmatically determine whether you're charged for training a model.

JSON

```
"BaseModelProperties": {
    "title": "BaseModelProperties",
    "type": "object",
    "properties": {
        "deprecationDates": {
            "$ref": "#/definitions/BaseModelDeprecationDates"
        },
        "features": {
            "$ref": "#/definitions/BaseModelFeatures"
        },
        "chargeForAdaptation": {
            "description": "A value indicating whether model adaptation is charged.",
            "type": "boolean",
            "readOnly": true
        }
    }
},
```

Text normalization

The `textNormalizationKind` property is added to `DatasetProperties`.

Entity definition for `TextNormalizationKind`: The kind of text normalization.

- Default: Default text normalization (for example, 'two to three' replaces '2 to 3' in en-US).
- None: No text normalization is applied to the input text. This value is an override option that should only be used when text is normalized before the upload.

Evaluation properties

Added token count and token error properties to the `EvaluationProperties` properties:

- `correctTokenCount1`: The number of correctly recognized tokens by model1.
- `tokenCount1`: The number of processed tokens by model1.
- `tokenDeletionCount1`: The number of recognized tokens by model1 that are deletions.
- `tokenErrorRate1`: The token error rate of recognition with model1.
- `tokenInsertionCount1`: The number of recognized tokens by model1 that are insertions.
- `tokenSubstitutionCount1`: The number of recognized words by model1 that are substitutions.
- `correctTokenCount2`: The number of correctly recognized tokens by model2.
- `tokenCount2`: The number of processed tokens by model2.
- `tokenDeletionCount2`: The number of recognized tokens by model2 that are deletions.
- `tokenErrorRate2`: The token error rate of recognition with model2.
- `tokenInsertionCount2`: The number of recognized tokens by model2 that are insertions.
- `tokenSubstitutionCount2`: The number of recognized words by model2 that are substitutions.

Model copy

The following changes are for the scenario where you copy a model.

- Added the new [Models_Copy](#) operation. Here's the schema in the new copy operation:
`"$ref": "#/definitions/ModelCopyAuthorization"`
- Deprecated the [Models_CopyTo](#) operation. Here's the schema in the deprecated copy operation: `"$ref": "#/definitions/ModelCopy"`
- Added the new [Models_AuthorizeCopy](#) operation that returns `"$ref": "#/definitions/ModelCopyAuthorization"`. This returned entity can be used in the new [Models_Copy](#) operation.

Added a new entity definition for `ModelCopyAuthorization`:

JSON

```
"ModelCopyAuthorization": {
    "title": "ModelCopyAuthorization",
    "required": [
        "expirationDateTime",
        "id",
        "sourceResourceId",
        "targetResourceEndpoint",
        "targetResourceId",
        "targetResourceRegion"
    ],
    "type": "object",
    "properties": {
        "targetResourceRegion": {
            "description": "The region (aka location) of the target speech resource (e.g., westus2).",
            "minLength": 1,
            "type": "string"
        },
        "targetResourceId": {
            "description": "The Azure Resource ID of the target speech resource.",
            "minLength": 1,
            "type": "string"
        },
        "targetResourceEndpoint": {
            "description": "The endpoint (base url) of the target resource (with custom domain name when it is used).",
            "minLength": 1,
            "type": "string"
        },
        "sourceResourceId": {
            "description": "The Azure Resource ID of the source speech resource.",
            "minLength": 1,
            "type": "string"
        },
        "expirationDateTime": {
            "format": "date-time",
            "description": "The expiration date of this copy authorization.",
            "type": "string"
        },
        "id": {
            "description": "The ID of this copy authorization.",
            "minLength": 1,
            "type": "string"
        }
    }
},
```

Added a new entity definition for `ModelCopyAuthorizationDefinition`:

JSON

```
"ModelCopyAuthorizationDefinition": {  
    "title": "ModelCopyAuthorizationDefinition",  
    "required": [  
        "sourceResourceId"  
    ],  
    "type": "object",  
    "properties": {  
        "sourceResourceId": {  
            "description": "The Azure Resource ID of the source speech resource.",  
            "minLength": 1,  
            "type": "string"  
        }  
    }  
},
```

CustomModelLinks copy properties

Added a new `copy` property.

- `copyTo` URI: The location of the obsolete model copy action. See the [Models_CopyTo](#) operation for more details.
- `copy` URI: The location of the model copy action. See the [Models_Copy](#) operation for more details.

JSON

```
"CustomModelLinks": {  
    "title": "CustomModelLinks",  
    "type": "object",  
    "properties": {  
        "copyTo": {  
            "format": "uri",  
            "description": "The location to the obsolete model copy action. See operation \\\"Models_CopyTo\\\" for more details.",  
            "type": "string",  
            "readOnly": true  
        },  
        "copy": {  
            "format": "uri",  
            "description": "The location to the model copy action. See operation \\\"Models_Copy\\\" for more details.",  
            "type": "string",  
            "readOnly": true  
        },  
        "files": {  
            "format": "uri",  
            "description": "The location to get all files of this entity. See operation \\\"Models_ListFiles\\\" for more details.",  
            "type": "string",  
            "readOnly": true  
        }  
    }  
},
```

```
        "readOnly": true
    },
    "manifest": {
        "format": "uri",
        "description": "The location to get a manifest for this model to be used in  
the on-prem container. See operation \\\"Models_GetCustomModelManifest\\\" for more  
details.",
        "type": "string",
        "readOnly": true
    }
},
"readOnly": true
},
```

Operation IDs

You must update the base path in your code from `/speechtotext/v3.1` to `/speechtotext/v3.2`.

For example, to get base models in the `eastus` region, use

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.2/models/base` instead of

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/models/base`.

Next steps

- [Speech to text REST API](#)
- [Speech to text REST API v3.2 reference](#)

Last updated on 10/28/2025

Migrate code from v3.0 to v3.1 of the REST API

The Speech to text REST API is used for [fast transcription](#), [batch transcription](#), and [custom speech](#). Changes from version 3.0 to 3.1 are described in the sections below.

Important

Speech to text REST API version `2024-11-15` is the latest version that's generally available.

- [Speech to text REST API](#) version `2024-05-15-preview` will be retired on a date to be announced.
- Speech to text REST API `v3.0`, `v3.1`, `v3.2`, `3.2-preview.1`, and `3.2-preview.2` will be retired on March 31st, 2026.

For more information about upgrading, see the Speech to text REST API [v3.0 to v3.1](#), [v3.1 to v3.2](#), and [v3.2 to 2024-11-15](#) migration guides.

Base path

You must update the base path in your code from `/speechtotext/v3.0` to `/speechtotext/v3.1`.

For example, to get base models in the `eastus` region, use

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/models/base` instead of

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.0/models/base`.

Note these other changes:

- The `/models/{id}/copyto` operation (includes '/') in version 3.0 is replaced by the `/models/{id}:copyto` operation (includes ':') in version 3.1.
- The `/webhooks/{id}/ping` operation (includes '/') in version 3.0 is replaced by the `/webhooks/{id}:ping` operation (includes ':') in version 3.1.
- The `/webhooks/{id}/test` operation (includes '/') in version 3.0 is replaced by the `/webhooks/{id}:test` operation (includes ':') in version 3.1.

For more information, see [Operation IDs](#) later in this guide.

Batch transcription

Note

Don't use Speech to text REST API v3.0 to retrieve a transcription created via Speech to text REST API v3.1. You'll see an error message such as the following: "The API version cannot be used to access this transcription. Please use API version v3.1 or higher."

In the [Transcriptions_Create](#) operation the following three properties are added:

- The `displayFormWordLevelTimestampsEnabled` property can be used to enable the reporting of word-level timestamps on the display form of the transcription results. The results are returned in the `displayWords` property of the transcription file.
- The `diarization` property can be used to specify hints for the minimum and maximum number of speaker labels to generate when performing optional diarization (speaker separation). With this feature, the service is now able to generate speaker labels for more than two speakers. To use this property, you must also set the `diarizationEnabled` property to `true`. With the v3.1 API, we have increased the number of speakers that can be identified through diarization from the two speakers supported by the v3.0 API. It's recommended to keep the number of speakers under 30 for better performance.
- The `languageIdentification` property can be used to specify settings for language identification on the input prior to transcription. Up to 10 candidate locales are supported for language identification. The returned transcription includes a new `locale` property for the recognized language or the locale that you provided.

The `filter` property is added to the [Transcriptions_List](#), [Transcriptions_ListFiles](#), and [Projects_ListTranscriptions](#) operations. The `filter` expression can be used to select a subset of the available resources. You can filter by `displayName`, `description`, `createdDateTime`, `lastActionDateTime`, `status`, and `locale`. For example: `filter=createdDateTime gt 2022-02-01T11:00:00Z`

If you use webhook to receive notifications about transcription status, note that the webhooks created via V3.0 API can't receive notifications for V3.1 transcription requests. You need to create a new webhook endpoint via V3.1 API in order to receive notifications for V3.1 transcription requests.

Custom speech

Datasets

The following operations are added for uploading and managing multiple data blocks for a dataset:

- [Datasets_UploadBlock](#) - Upload a block of data for the dataset. The maximum size of the block is 8MiB.
- [Datasets_GetBlocks](#) - Get the list of uploaded blocks for this dataset.
- [Datasets_CommitBlocks](#) - Commit blocklist to complete the upload of the dataset.

To support model adaptation with [structured text in markdown](#) data, the [Datasets_Create](#) operation now supports the **LanguageMarkdown** data kind. For more information, see [upload datasets](#).

Models

The [Models_ListBaseModels](#) and [Models_GetBaseModel](#) operations return information on the type of adaptation supported by each base model.

JSON

```
"features": {  
    "supportsAdaptationsWith": [  
        "Acoustic",  
        "Language",  
        "LanguageMarkdown",  
        "Pronunciation"  
    ]  
}
```

The [Models_Create](#) operation has a new `customModelWeightPercent` property where you can specify the weight used when the Custom Language Model (trained from plain or structured text data) is combined with the Base Language Model. Valid values are integers between 1 and 100. The default value is currently 30.

The `filter` property is added to the following operations:

- [Datasets_List](#)
- [Datasets_ListFiles](#)
- [Endpoints_List](#)
- [Evaluations_List](#)
- [Evaluations_ListFiles](#)
- [Models_ListBaseModels](#)
- [Models_ListCustomModels](#)
- [Projects_List](#)
- [Projects_ListDatasets](#)

- [Projects_ListEndpoints](#)
- [Projects_ListEvaluations](#)
- [Projects_ListModels](#)

The `filter` expression can be used to select a subset of the available resources. You can filter by `displayName`, `description`, `createdDateTime`, `lastActionDateTime`, `status`, `locale`, and `kind`. For example: `filter=locale eq 'en-US'`

Added the [Models_ListFiles](#) operation to get the files of the model identified by the given ID.

Added the [Models_GetFile](#) operation to get one specific file (identified with `fileId`) from a model (identified with ID). This lets you retrieve a **ModelReport** file that provides information on the data processed during training.

Operation IDs

You must update the base path in your code from `/speechtotext/v3.0` to `/speechtotext/v3.1`.

For example, to get base models in the `eastus` region, use

`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.1/models/base` instead of
`https://eastus.api.cognitive.microsoft.com/speechtotext/v3.0/models/base`.

The name of each `operationId` in version 3.1 is prefixed with the object name. For example, the `operationId` for "Create Model" changed from [CreateModel](#) in version 3.0 to [Models_Create](#) in version 3.1.

The `/models/{id}/copyto` operation (includes '/') in version 3.0 is replaced by the `/models/{id}:copyto` operation (includes ':') in version 3.1.

The `/webhooks/{id}/ping` operation (includes '/') in version 3.0 is replaced by the `/webhooks/{id}:ping` operation (includes ':') in version 3.1.

The `/webhooks/{id}/test` operation (includes '/') in version 3.0 is replaced by the `/webhooks/{id}:test` operation (includes ':') in version 3.1.

Next steps

- [Speech to text REST API](#)
- [Speech to text REST API v3.1 reference](#)
- [Speech to text REST API v3.0 reference](#)

Last updated on 10/28/2025

Migrate code from Long Audio API to Batch synthesis API

08/07/2025

The [Batch synthesis API](#) provides asynchronous synthesis of long-form text to speech. This article describes the benefits of upgrading from Long Audio API to Batch synthesis API, and details about how to do so.

ⓘ Important

[Batch synthesis API](#) is generally available. the Long Audio API is retiring on April 1st, 2027.

Base path and version

Update the endpoint from `https://YourSpeechRegion.customvoice.api.speech.microsoft.com` to `https://YourSpeechRegion.api.cognitive.microsoft.com` or you can use custom domain instead: `https://{{customDomainName}}.cognitiveservices.azure.com/`.

Update the base path in your code from `/texttospeech/v3.0/longaudiosynthesis` to `/texttospeech/batchsyntheses`.

Update the version from base path to query string `/texttospeech/v3.0/longaudiosynthesis` to `?api-version=2024-04-01`.

For example, to list synthesis jobs for your Speech resource in the `eastus` region, use

`https://eastus.api.cognitive.microsoft.com/texttospeech/batchsyntheses?api-version=2024-04-01` instead of

`https://eastus.customvoice.api.speech.microsoft.com/api/texttospeech/v3.0/longaudiosynthesis`.

Regions and endpoints

Batch synthesis API is available in more [Speech regions](#).

The Long Audio API is limited to the following regions:

 Expand table

Region	Endpoint
Australia East	https://australiaeast.customvoice.api.speech.microsoft.com
East US	https://eastus.customvoice.api.speech.microsoft.com
India Central	https://centralindia.customvoice.api.speech.microsoft.com
South Central US	https://southcentralus.customvoice.api.speech.microsoft.com
Southeast Asia	https://southeastasia.customvoice.api.speech.microsoft.com
UK South	https://uksouth.customvoice.api.speech.microsoft.com
West Europe	https://westeurope.customvoice.api.speech.microsoft.com

Voices list

Batch synthesis API supports all [text to speech voices and styles](#).

The Long Audio API is limited to the set of voices returned by a GET request to <https://<endpoint>/api/texttospeech/v3.0/longaudiosynthesis/voices>.

Text inputs

Batch synthesis text inputs are sent in a JSON payload of up to 2 megabytes.

Long Audio API text inputs are uploaded from a file that meets the following requirements:

- One plain text (.txt) or SSML text (.txt) file encoded as [UTF-8 with Byte Order Mark \(BOM\)](#). Don't use compressed files such as ZIP. If you have more than one input file, you must submit multiple requests.
- Contains more than 400 characters for plain text or 400 [billable characters](#) for SSML text, and less than 10,000 paragraphs. For plain text, each paragraph is separated by a new line. For SSML text, each SSML piece is considered a paragraph. Separate SSML pieces by different paragraphs.

With Batch synthesis API, you can use any of the [supported SSML elements](#), including the `audio`, `mstts:backgroundaudio`, and `lexicon` elements. The long audio API doesn't support the `audio`, `mstts:backgroundaudio`, and `lexicon` elements.

Audio output formats

Batch synthesis API supports all [text to speech audio output formats](#).

The Long Audio API is limited to the following set of audio output formats. The sample rate for long audio voices is 24kHz, not 48kHz. Other sample rates can be obtained through upsampling or downsampling when synthesizing.

- riff-8khz-16bit-mono-pcm
- riff-16khz-16bit-mono-pcm
- riff-24khz-16bit-mono-pcm
- riff-48khz-16bit-mono-pcm
- audio-16khz-32kbitrate-mono-mp3
- audio-16khz-64kbitrate-mono-mp3
- audio-16khz-128kbitrate-mono-mp3
- audio-24khz-48kbitrate-mono-mp3
- audio-24khz-96kbitrate-mono-mp3
- audio-24khz-160kbitrate-mono-mp3

Getting results

With batch synthesis API, use the URL from the `outputs.result` property of the HTTP GET batch synthesis response. The `results` are in a ZIP file that contains the audio (such as `0001.wav`), summary, and debug details.

Long Audio API text inputs and results are returned via two separate content URLs as shown in the following example. The one with `"kind": "LongAudioSynthesisScript"` is the input script submitted. The other one with `"kind": "LongAudioSynthesisResult"` is the result of this request. Both ZIP files can be downloaded from the URL in their `links.contentUrl` property.

Cleaning up resources

Batch synthesis API supports up to 300 batch synthesis jobs that don't have a status of "Succeeded" or "Failed". The Speech service keeps each synthesis history for up to 31 days, or the duration of the request `timeToLiveInHours` property, whichever comes sooner. The date and time of automatic deletion (for synthesis jobs with a status of "Succeeded" or "Failed") is equal to the `lastActionDateTime + timeToLiveInHours` properties.

The Long Audio API is limited to 20,000 requests for each Azure subscription account. The Speech service doesn't remove job history automatically. You must remove the previous job run history before making new requests that would otherwise exceed the limit.

Next steps

- Batch synthesis API
- Speech Synthesis Markup Language (SSML)
- Text to speech quickstart

Migrate to custom voice REST API (Preview)

08/07/2025

! Note

This feature is currently in public preview. This preview is provided without a service-level agreement, and is not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

The custom voice REST API is a new version of the text to speech REST API. You can deploy and use custom voice models with the API.

In this article, you learn how to migrate code from the v3 text to speech REST API to the custom voice REST API.

This article retains information about the v3 text to speech REST API for reference. But you should use the custom voice REST API for new development.

Required changes

Suspend endpoint: Use the custom voice API [suspend endpoint](#) operation to suspend an endpoint. The v3 text to speech REST API [suspend endpoint operation](#) is retiring.

Resume endpoint: Use the custom voice API [resume endpoint](#) operation to suspend an endpoint. The v3 text to speech REST API [resume endpoint operation](#) is retiring.

Reference documentation for v3 text to speech REST API (retiring)

Suspend and resume endpoint via REST API

This section shows you how to [get](#), [suspend](#), or [resume](#) a custom voice endpoint via REST API.

Get endpoint

Get the endpoint by endpoint ID. The operation returns details about an endpoint such as model ID, project ID, and status.

For example, you might want to track the status progression for [suspend](#) or [resume](#) operations. Use the `status` property in the response payload to determine the status of the endpoint.

The possible `status` property values are:

[[Expand table](#)

Status	Description
<code>NotStarted</code>	The endpoint isn't deployed and it's not available for speech synthesis.
<code>Running</code>	The endpoint is in the process of being deployed or resumed, and it's not available for speech synthesis.
<code>Succeeded</code>	The endpoint is active and available for speech synthesis. The endpoint is deployed or the resume operation succeeded.
<code>Failed</code>	The endpoint deployment or suspend operation failed. The endpoint can only be viewed or deleted in Speech Studio .
<code>Disabling</code>	The endpoint is in the process of being suspended, and it's not available for speech synthesis.
<code>Disabled</code>	The endpoint is inactive, and it's not available for speech synthesis. The suspend operation succeeded or the resume operation failed.

Tip

If the status is `Failed` or `Disabled`, check `properties.error` for a detailed error message. However, there won't be error details if the status is `Disabled` due to a successful suspend operation.

Get endpoint example

For information about endpoint ID, region, and Speech resource key parameters, see [request parameters](#).

HTTP example:

HTTP

```
GET api/texttospeech/v3.0/endpoints/<YourEndpointId> HTTP/1.1
Ocp-Apim-Subscription-Key: YourResourceKey
```

Host: <YourResourceRegion>.customvoice.api.speech.microsoft.com

cURL example:

Console

```
curl -v -X GET  
"https://<YourResourceRegion>.customvoice.api.speech.microsoft.com/api/texttospee  
ch/v3.0/endpoints/<YourEndpointId>" -H "Ocp-Apim-Subscription-Key: <YourResourceKey  
>"
```

Response header example:

Status code: 200 OK

Response body example:

JSON

```
{  
  "model": {  
    "id": "a92aa4b5-30f5-40db-820c-d2d57353de44"  
  },  
  "project": {  
    "id": "ffc87aba-9f5f-4bfa-9923-b98186591a79"  
  },  
  "properties": {},  
  "status": "Succeeded",  
  "lastActionDateTime": "2019-01-07T11:36:07Z",  
  "id": "e7ffdf12-17c7-4421-9428-a7235931a653",  
  "createdDateTime": "2019-01-07T11:34:12Z",  
  "locale": "en-US",  
  "name": "Voice endpoint",  
  "description": "Example for voice endpoint"  
}
```

Suspend endpoint

You can suspend an endpoint to limit spend and conserve resources that aren't in use. You aren't charged while the endpoint is suspended. When you resume an endpoint, you can use the same endpoint URL in your application to synthesize speech.

You suspend an endpoint with its unique deployment ID. The endpoint status must be **Succeeded** before you can suspend it.

Use the [get endpoint](#) operation to poll and track the status progression from `Succeeded`, to `Disabling`, and finally to `Disabled`.

Suspend endpoint example

For information about endpoint ID, region, and Speech resource key parameters, see [request parameters](#).

HTTP example:

HTTP

```
POST api/texttospeech/v3.0/endpoints/<YourEndpointId>/suspend HTTP/1.1
Ocp-Apim-Subscription-Key: YourResourceKey
Host: <YourResourceRegion>.customvoice.api.speech.microsoft.com
Content-Type: application/json
Content-Length: 0
```

cURL example:

Console

```
curl -v -X POST
"https://<YourResourceRegion>.customvoice.api.speech.microsoft.com/api/texttospeech/v3.0/endpoints/<YourEndpointId>/suspend" -H "Ocp-Apim-Subscription-Key:<YourResourceKey>" -H "content-type: application/json" -H "content-length: 0"
```

Response header example:

Status code: 202 Accepted

For more information, see [response headers](#).

Resume endpoint

When you resume an endpoint, you can use the same endpoint URL that you used before it was suspended.

You resume an endpoint with its unique deployment ID. The endpoint status must be `Disabled` before you can resume it.

Use the [get endpoint](#) operation to poll and track the status progression from `Disabled`, to `Running`, and finally to `Succeeded`. If the resume operation failed, the endpoint status is `Disabled`.

Resume endpoint example

For information about endpoint ID, region, and Speech resource key parameters, see [request parameters](#).

HTTP example:

HTTP

```
POST api/texttospeech/v3.0/endpoints/<YourEndpointId>/resume HTTP/1.1
Ocp-Apim-Subscription-Key: YourResourceKey
Host: <YourResourceRegion>.customvoice.api.speech.microsoft.com
Content-Type: application/json
Content-Length: 0
```

cURL example:

Console

```
curl -v -X POST
"https://<YourResourceRegion>.customvoice.api.speech.microsoft.com/api/texttospeech/v3.0/endpoints/<YourEndpointId>/resume" -H "Ocp-Apim-Subscription-Key: <YourResourceKey >" -H "content-type: application/json" -H "content-length: 0"
```

Response header example:

Status code: 202 Accepted

For more information, see [response headers](#).

Parameters and response codes

Request parameters

You use these request parameters with calls to the REST API.

[+] Expand table

Name	Location	Required	Type	Description
YourResourceRegion	Path	True	string	The Azure region the endpoint is associated with.
YourEndpointId	Path	True	string	The identifier of the endpoint.
Ocp-Apim-Subscription-Key	Header	True	string	The Speech resource key the endpoint is associated with.

Response headers

Status code: 202 Accepted

[Expand table](#)

Name	Type	Description
Location	string	The location of the endpoint that can be used as the full URL to get endpoint.
Retry-After	string	The total seconds of recommended interval to retry to get endpoint status.

HTTP status codes

The HTTP status code for each response indicates success or common errors.

[Expand table](#)

HTTP status code	Description	Possible reason
200	OK	The request was successful.
202	Accepted	The request was accepted and is being processed.
400	Bad Request	The value of a parameter is invalid, or a required parameter is missing, empty, or null. One common issue is a header that is too long.
401	Unauthorized	The request isn't authorized.
429	Too Many Requests	You exceeded the quota or rate of requests allowed for your Speech resource.
502	Bad Gateway	Network or server-side issue. This status code might indicate invalid headers.

Next steps

- Deploy a professional voice

Migrate from retired intent recognition

Intent recognition in Azure AI Speech was retired on September 30, 2025. Applications can no longer use intent recognition via Azure AI Speech. However, you can still perform intent recognition using Azure AI Language Service or Azure OpenAI.

This change doesn't affect other Azure AI Speech capabilities such as [speech to text](#) (including no change to speaker diarization), [text to speech](#), and [speech translation](#).

Azure AI Speech previously exposed the `IntentRecognizer` object family in the Speech SDK.

These APIs depended on a Language Understanding Intelligent Service (LUIS) application or simple pattern matching constructs. With the retirement:

- `IntentRecognizer`, pattern matching intents/entities, and related parameters are no longer available.
- Existing applications must remove direct Speech SDK intent logic and adopt a two-step approach (speech to text, then intent classification) or a single prompt-based approach.

Choose an alternative

[] Expand table

Requirement	Recommended service	Why
Structured intent and entity extraction with labeled training data	Azure AI Language Service Conversational Language Understanding (CLU)	Purpose-built for multi-intent classification and entity extraction; supports versions, testing, and analytics.
Few-shot or zero-shot dynamic intent determination	Azure OpenAI	Use GPT models with example prompts; rapidly adapt without schema changes.
Combine transcription with generative reasoning (summaries + intents)	Azure OpenAI + Speech	Transcribe audio then enrich with GPT outputs for complex reasoning.
Multilingual speech input flowed into consistent intent schema	Speech (STT) + CLU	Speech handles transcription; CLU handles normalization and classification.

Migration steps

1. Replace any Speech SDK `IntentRecognizer` usage with `SpeechRecognizer` or `ConversationTranscriber` to obtain text.
2. For structured intent/entity needs, create a CLU project and deploy a model. Send transcribed utterances to the CLU prediction API.
3. For flexible or rapid scenarios, craft a prompt for an Azure OpenAI model including representative user utterances and expected JSON intent output.
4. Remove dependencies on `LanguageUnderstandingModel` and any LUIS application IDs or endpoints from configuration.
5. Eliminate pattern matching code referencing `PatternMatchingIntent` or `PatternMatchingEntity` types.
6. Validate accuracy by comparing historic `IntentRecognizer` outputs to CLU classification results or OpenAI completions, adjusting training data or prompts as needed.
7. Update monitoring: shift any existing intent latency/accuracy dashboards to new sources (CLU evaluation logs or OpenAI prompt result tracking).

Sample architecture

1. Speech to text transcribes audio into text with real-time or batch mode.
2. Text is sent to CLU or Azure OpenAI depending on your intent strategy.
3. Response is normalized into a common JSON shape (for example: `{ "intent": "BookFlight", "entities": { "Destination": "Seattle" } }`).
4. Business logic routes the normalized output to downstream services (booking, knowledge base, workflow engine).

Result format considerations

[] Expand table

Aspect	CLU	Azure OpenAI
Schema stability	High (defined intents/entities)	Flexible (prompt-defined)
Versioning	Built-in model versions	Manual prompt versioning
Training effort	Requires labeled dataset	Few-shot examples in prompt
Edge cases	Requires more labeled data	Add examples or instructions
Latency	Prediction API call	Completion API call (similar)

Frequently asked questions

Do I need to re-label data? If you used LUIS, you need to export and reimport data into CLU, then retrain. Mapping is often direct (intents, entities). Pattern matching intents might require manual conversion to examples.

Can I combine CLU and Azure OpenAI? Yes. Use CLU for deterministic classification and OpenAI for summarization or fallback classification when confidence is low.

Is speaker diarization affected? No. Diarization features continue; you just process each speaker segment through CLU or OpenAI after transcription.

Related links

- [Speech to text](#)
- [Conversational Language Understanding overview](#)
- [Azure OpenAI overview](#)

ⓘ **Note:** The author created this article with assistance from AI. [Learn more](#)

Last updated on 11/01/2025

Use cases for Speech to text

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

What is a Transparency Note?

An AI system includes not only the technology, but also the people who will use it, the people who will be affected by it, and the environment in which it is deployed. Creating a system that is fit for its intended purpose requires an understanding of how the technology works, what its capabilities and limitations are, and how to achieve the best performance. Microsoft's Transparency Notes are intended to help you understand how our AI technology works, the choices system owners can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Transparency Notes when developing or deploying your own system, or share them with the people who will use or be affected by your system.

Microsoft's Transparency Notes are part of a broader effort at Microsoft to put our AI Principles into practice. To find out more, see the [Microsoft AI principles](#).

The basics of speech to text

Speech to text, also known as automatic speech recognition (ASR), is a feature under the Azure AI Speech service, which is a part of Azure AI services. [Speech to text](#) converts spoken audio into text. Speech to text in Azure supports more than 140 locales for input. For the latest list of supported locales, see [Language and voice support for the Speech service](#).

Key terms

 Expand table

Term	Definition
Audio input	The streamed audio data or audio file that's used as an input for the speech to text feature. Audio input can contain not only voice, but also silence and non-speech noise. Speech to text generates text for the voice parts of audio input.

Term	Definition
Utterance	A component of audio input that contains human voice. One utterance can consist of a single word or multiple words, such as a phrase.
Transcription	The text output of the speech to text feature. This automatically generated text output leverages <i>speech models</i> and is sometimes called machine transcription or automated speech recognition (ASR). Transcription in this context is fully automated and therefore different from human transcription, which is text that is generated by human transcribers.
Speech model	An automatically generated, machine-learned numerical representation of an utterance that is used to infer a transcription from an audio input. Speech models are trained on voice data that includes various speech styles, languages, accents, dialects, and intonations, and on acoustic variations that are generated by using different types of recording devices. A speech model numerically represents both acoustic and linguistic features, which are used to predict what text should be associated with the utterance.
Real-time API	An API that accepts requests with audio input, and returns a response in real time with transcription within the same network connection.
Language Detection API	A type of real-time API that detects what language is spoken in an audio input. A language is inferred based on voice sound in the audio input.
Speech Translation API	Another type of real-time API that generates transcriptions of a given audio input then translates them into a language specified by the user. This is a cascaded service of Speech Services and Text Translator.
Batch API	A service that is used to send audio input to be transcribed at a later time. You specify the location of audio files and other parameters, such as the language of recognition. The service loads the audio input asynchronously and transcribes it. When transcription is complete, text files are loaded back to a location that you specify.
Diarization	Diarization answers the question of who spoke and when. It differentiates speakers in an audio input based on their voice characteristics. Both real-time and batch APIs support diarization and are capable of differentiating speakers' voices on monochannel recordings. Diarization is combined with speech to text functionality to provide transcription outputs that contain a speaker entry for each transcribed segment. The transcription output is tagged as GUEST1, GUEST2, GUEST3, etc. based on the number of speakers in the audio conversation.
Word error rate(WER)	Word error rate (WER) is the industry standard to measure speech to text accuracy. WER counts the number of incorrect words that are identified during recognition. Then it divides by the total number of words that are provided in the correct transcript (often created by human labeling).
Token error rate(TER)	Token error rate (TER) is a measure of the correctness of the final recognition of words, capitalization, punctuation, and so on, compared to tokens that are provided in the correct transcript (often created by human labeling).

Term	Definition
Runtime latency	In speech to text, latency is the time between the speech audio input and the transcription result output.
Word diarization error rate (WDER)	Word diarization error rate (WDER) counts the number of errors on the words assigned to the wrong speaker compared to the ground truth. A lower WDER rate indicates better quality.

Capabilities

System behavior

Below we are listing the main ways to call our speech to text service.

Real-time Speech to text API

This is a common API call via the Speech SDK or REST API to send an audio input and receive a text transcription in real time. The speech system uses a speech model to recognize what is spoken in an input audio. During real-time speech to text, the system takes an audio stream as input and continuously determines the most likely sequence of words that produced the audio that's observed so far. The model is trained on a large amount of diverse audio across typical usage scenarios and a wide range of speakers. For example, this feature is often used for voice-enabled queries or dictation within an organization's service or application.

Batch transcription API

Batch transcription is another type of API call. It's typically used to send prerecorded audio inputs and to receive transcribed text asynchronously (that is, at a later time). To use this API, you can specify locations for multiple audio files. The speech to text technology reads the audio input from the file and generates transcription text files that are returned to the storage location that you specify. This feature is used to support larger transcription jobs in which it is not necessary to provide end users with the transcription content in real time. An example is transcribing call center recordings to gain insights into customers and call center agent performance.

When you use batch transcription, you can choose to use the Whisper model instead of the default Azure AI speech to text model. To determine whether the Whisper model is appropriate for your use case, you can compare how the output between these models differs in the batch. Try it out in [Speech Studio](#), and then perform deeper evaluations by using the test

capabilities through custom speech. Note that the Whisper model is also available thorough Azure OpenAI.

Speech translation API

This API converts audio input to text, and then translates it into another language. The translated transcription output can be returned in text format, or you can choose to have the text synthesized into audible speech by using [text to speech](#). For more information, see [What is Azure AI Translator?](#)

Sub-features and options

The APIs above can optionally use the following sub-features:

- **Model customization:** Azure AI Speech enables developers to customize the speech to text models in order to improve the recognition accuracy for a specific scenario. There are two ways to customize speech to text:
 - At runtime through the use of the [phrase list](#) feature
 - Ahead of time through the use of [custom speech](#)
- **Language detection:** Unlike in a default API call, in which a language or locale for an audio input must be specified in advance, with language detection, you can specify multiple locales and let the service detect which language should be used to recognize a specific part of the audio.
- **Diarization:** This feature is disabled by default. If you choose to enable this feature, the service differentiates different speakers' utterances. The resulting transcription text contains a "speaker" property that indicates GUEST1, GUEST2, GUEST3, and so on, which denotes which speaker is speaking in an audio file.

Use cases

Speech to text can offer different ways for users to interact with applications and devices. Instead of typing words on a keyboard or using their hands for touchscreen interactions, speech to text technology allows users to operate applications and devices by voice and through dictation.

- **Smart assistants:** Companies that are developing smart assistants on appliances, cars, and homes can use speech to text to enable natural interface search queries or to trigger certain features by voice. This is called *_command-and-_control*.
- **Chat bots:** Companies can build chat bot applications, in which users can use voice-enabled queries or commands to interact with bots.

- **Voice typing:** Apps can allow users to use voice to dictate long-form text. Voice typing can be used to enter text for messaging, emails, and documents.
- **Voice commanding:** Users can trigger certain actions by voice (command-and-control). Two common examples are entering query text by voice and selecting a menu item by voice.
- **Voice translation:** You can use the speech translation features of speech to text technology to communicate by voice with other users who speak different languages. Speech translation enables voice-to-voice communication across multiple languages. See the latest list of supported locales in [Language and voice support for the Speech service](#).
- **Call center transcriptions:** Companies often record conversations with their users in scenarios like customer support calls. Audio recordings can be sent to the batch API for transcription.
- **Mixed-language dictation:** Users can use speech to text technology to dictate in multiple languages. Using language detection, a dictation application can automatically detect spoken languages and transcribe appropriately without requiring a user to specify which language they speak.
- **Live conversation transcription:** When speakers are all in the same room using a single-microphone setup, do live transcription about which speaker (Guest1, Guest2, Guest3, and so on) makes each statement.
- **Conversation transcription of prerecorded audio:** After the recording of audio with multiple speakers you can use our service to get the transcription about which speaker (Guest1, Guest2, Guest3, and so on) makes each statement.

Considerations when choosing other use cases

The speech to text API offers convenient options for developing voice-enabled applications, but it is very important to consider the context in which you will integrate the API. You must ensure that you comply with all laws and regulations that apply to your application. This includes understanding your obligations under privacy and communication laws, including national and regional privacy, eavesdropping, and wiretap laws that apply to your jurisdiction. Collect and process only audio that is within the reasonable expectations of your users. This includes ensuring that you have all necessary and appropriate consents from users for you to collect, process, and store their audio data.

Many applications are designed and intended to be used by a specific individual user for voice-enabled queries, commands, or dictation. However, the microphone for your application might pick up sound or voice from non-primary users. To avoid unintentionally capturing the voices of non-primary users, you should consider the following information:

- **Microphone considerations:** Often, you cannot control who might speak near the input device that sends audio input to the speech to text cloud service. You should encourage

your users to take extra care when they use voice-enabled features and applications in a public or open environment where other people's voices might be easily captured.

- **Use speech to text only in experiences and features that are within the reasonable expectations of your users:** Audio data that contains a person speaking is personal information. Speech to text is not intended to be used for covert audio surveillance purposes, in a manner that violates legal requirements, or in applications and devices in public spaces or locations where users might have a reasonable expectation of privacy. Use the Speech service only to collect and process audio in ways that are within the reasonable expectations of your users. This includes ensuring that you have all necessary and appropriate consents from users to collect, process, and store their audio data.
- **Azure AI Speech service and integration of the Whisper model:** The Whisper model enhances the Azure AI Speech service with advanced features like multilingual recognition and readability. The Speech service also enriches the performance of the Whisper model by enabling larger-scale batch transcriptions and speaker diarization. Whether to use the default Speech service speech to text model or Whisper model depends on the specific use case. We recommend that you take advantage of the batch try out and custom speech experiences in Speech Studio to evaluate both options to find the best fit for your business needs.
- **Conversation transcription on prerecorded events:** The system will perform better if all speakers are in the same acoustic environment (for example, the conversation takes place in a room in which people speak into a common microphone).
- **Conversation transcription:** Although there is no limitation on the numbers of speakers in the conversation, the system performs better when the number of speakers is under 30.
- **Legal and regulatory considerations:** Organizations need to evaluate potential specific legal and regulatory obligations when using any AI services and solutions, which may not be appropriate for use in every industry or scenario. Additionally, AI services or solutions are not designed for and may not be used in ways prohibited in applicable terms of service and relevant codes of conduct.

Unsupported uses

- **Conversation transcription with speaker recognition:** The Speech service is not designed to provide diarization with speaker recognition, and it cannot be used to identify individuals. In other words, speakers will be presented as Guest1, Guest2, Guest3, and so on, in the transcription. These will be randomly assigned and may not be used to identify individual speakers in the conversation. For each conversation transcription, the assignment of Guest1, Guest2, Guest3, and so on, will be random.

To prevent any potential for misuse of Speech service for identification purposes, you are responsible for ensuring that you use the service, including the diarization feature, only for

supported uses, and that you have a proper legal basis and any required consents in place for all uses of the service.

Limitations

Speech to text recognizes what's spoken in an audio input, and then generates transcription outputs. This requires proper setup for the expected languages used in the audio input and spoken styles. Non-optimal settings might lead to lower accuracy.

Technical limitations, operational factors, and ranges

Language of accuracy

The industry standard to measure speech to text accuracy is [word error rate \(WER\)](#). To understand the detailed WER calculation, see [Test the accuracy of a custom speech model](#).

Transcription accuracy and system limitations

Speech to text uses a unified speech recognition machine learning model to transcribe what is spoken in a wide range of contexts and topic domains, including command-and-control, dictation, and conversations. You don't need to consider using different models for your application or feature scenarios.

However, you need to specify a language or locale for each audio input. The language or locale must match the actual language that's spoken in an input voice. For more information, see the list of [supported locales](#).

Many factors can lead to a lower accuracy in transcription:

- **Acoustic quality:** Speech to text-enabled applications and devices might use a wide variety of microphone types and specifications. Unified speech models have been created based on various voice audio device scenarios, such as telephones, mobile phones, and speaker devices. However, voice quality might be degraded by the way a user speaks into a microphone, even if they use a high-quality microphone. For example, if a speaker is located far from the microphone, the input quality would be too low. A speaker who is too close to the microphone could also cause audio quality deterioration. Both cases can adversely affect the accuracy of speech to text.
- **Non-speech noise:** If an input audio contains a certain level of noise, accuracy is affected. Noise might come from the audio devices that are used to make a recording, or audio input itself might contain noise, such as background or environmental noise.

- **Overlapped speech:** There might be multiple speakers within range of an audio input device, and they might speak at the same time. Also, other speakers might speak in the background while the main user is speaking.
- **Vocabularies:** The speech to text model has been trained on a wide variety of words in many domains. However, users might speak organization-specific terms and jargon that aren't in a standard vocabulary. If a word that doesn't exist in a model appears in the audio, the result is an error in transcription.
- **Accents:** Even within one locale, such as in English - United States (en-US), many people have different accents. Very specific accents might also lead to an error in transcription.
- **Mismatched locales:** Users might not speak the languages that you expect. If you specified English - United States (en-US) for an audio input, but a speaker spoke in Swedish, for example, accuracy would be reduced.
- **Insertion errors:** At times, the speech to text models can produce insertion errors in the presence of noise or soft background speech. This is limited when you use the Speech service, but it's slightly more frequent when you use the Whisper model, as stated in the OpenAI [model card ↗](#).

Because of these acoustic and linguistic variations, you should expect a certain level of inaccuracy in the output text when you design an application.

System performance

System performance is measured by these key factors (from the user's point of view):

- Word error rate (WER)
- Token error rate (TER)
- Runtime latency

A model is considered better only when it shows significant improvements (such as a 5% relative WER improvement) in all scenarios (like transcription of conversation speech, call center transcription, dictation, and voice assistant) while being in line with the resource usage and response latency goals.

For diarization, we measure the quality by using word diarization error rate (WDER). The lower the WDER, the better the quality of diarization.

Best practices for improving system performance

As described earlier, acoustic conditions like background noise, side speech, distance to microphone, and speaking styles and characteristics can adversely affect the accuracy of what is recognized.

For better speech experiences, consider the following application or service design principles:

- **Design UIs to match input locales:** Mismatched locales reduce accuracy. The Speech SDK supports [automatic language detection](#), but it detects only one out of four locales that are specified at runtime. You still need to know the locale that your users will speak in. Your UI should clearly indicate which languages the users can speak in via a dropdown that lists the languages that are supported. For more information, see the [supported locales](#).
- **Allow users to try again:** Misrecognition might occur due to a temporary issue, such as unclear or fast speech or a long pause. If your application expects specific transcriptions, such as predefined action commands like "Yes" and "No" and it did not get any of them, users should be able to try again. A typical method is to tell users, "Sorry, I didn't get that. Please try again."
- **Confirm before you take an action by voice:** Just as with keyboard-based, click-based, or tap-based UIs, if an audio input can trigger an action, users should be given an opportunity to confirm the action, especially by displaying or playing back what was recognized or transcribed. A typical example is sending a text message by voice. An app repeats what was recognized and asks for confirmation: "You said, 'Thank you.' Send it or change it?"
- **Add custom vocabularies:** The general speech recognition model that's provided by speech to text covers a broad vocabulary. However, scenario-specific jargon and named entities (for example, people names and product names) might be underrepresented. What words and phrases are likely to be spoken can vary significantly depending on the scenario. If you can anticipate which words and phrases will be spoken (for instance, when a user selects an item from a list), you might want to use the phrasal list grammar. For more information, see "Improving recognition accuracy" in [Get started with speech to text](#).
- **Use custom speech:** If speech to text accuracy in your application scenarios remains low, you might want to consider customizing the model for your acoustic and linguistic variations. You can create your own models by training them by using your own voice audio data or text data. For details, see [custom speech](#).

Evaluation of speech to text

A speech to text model is evaluated through testing. The goal of testing is to confirm that the model performs well across each of the key scenarios and in prevalent audio conditions, and that we are achieving our fairness goals across demographic factors.

Evaluation methods

For model evaluation, test datasets are used. Both a regression test and a model performance test are run before each model deployment. Key metrics for regression tests are WER, TER, WDER (if diarization is enabled when doing speech to text), and latency at the 90th percentile.

Evaluation results

We strive to ship all model updates regression-free (that is, the updated model should only improve the current production model). Each candidate is compared directly to the current production model. To consider a model for deployment, we must see at least a 5% relative WER improvement over the current production model.

Speech to text models are trained and tuned by using voice audio that has variations, including:

- Microphones and device specifications
- Speech environment
- Speech scenarios
- Languages and accents of speakers
- Age and gender of speakers
- Ethnic background of speakers

For diarization, additional data variations are used:

- Length of time each speaker speaks
- Number of speakers
- Emotional speech that alters pitch and tone

The resulting speech to text system transcribes the user's spoken words into text, which then can be used by a dialog system with natural language understanding or for analytics like summarization or sentiment.

Fairness considerations

At Microsoft, we strive to empower every person on the planet to achieve more. An essential part of this goal is working to create technologies and products that are fair and inclusive. Fairness is a multidimensional, sociotechnical topic, and it affects many different aspects of our product development. Learn more about [the Microsoft approach to fairness](#).

One dimension we need to consider is how well the system performs for different groups of people. Research has shown that without conscious effort focused on improving performance for all groups, it is often possible for the performance of an AI system to vary across groups based on factors such as race, ethnicity, region, gender, and age.

Each version of the speech to text model is tested and evaluated against various test sets to make sure that the model can perform without a large gap in each of the evaluation criteria. More granular fairness results are coming soon.

Evaluating and integrating speech to text for your use

The performance of speech to text will vary depending on the real-world uses and conditions that you implement. To ensure optimal performance in your scenario, you should conduct your own evaluations of the solutions you implement by using speech to text.

A test voice dataset should consist of actual voice inputs that were collected in your applications in production. You should randomly sample data to reflect real user variations over a certain period of time. Also, the test dataset should be refreshed periodically to reflect changes in the variations.

Guidance for integration and responsible use with speech to text

As Microsoft works to help customers responsibly develop and deploy solutions by using speech to text, we are taking a principled approach to upholding personal agency and dignity by considering the AI systems' fairness, reliability & safety, privacy & security, inclusiveness, transparency, and human accountability. These considerations reflect our commitment to developing Responsible AI.

When getting ready to deploy AI-powered products or features, the following activities help to set you up for success:

- **Understand what it can do:** Fully assess the capabilities of speech to text to understand its capabilities and limitations. Understand how it will perform in your particular scenario and context by thoroughly testing it with real life conditions and data.
- **Respect an individual's right to privacy:** Only collect data and information from individuals for lawful and justifiable purposes. Only use data and information that you have consent to use for this purpose.
- **Legal review:** Obtain appropriate legal advice to review your solution, particularly if you will use it in sensitive or high-risk applications. Understand what restrictions you might need to work within and your responsibility to resolve any issues that might come up in the future. Do not provide any legal advice or guidance.
- **Human-in-the-loop:** Keep a human-in-the-loop and include human oversight as a consistent pattern area to explore. This means ensuring constant human oversight of the

AI-powered product or feature and maintaining the role of humans in decision making. Ensure that you can have real-time human intervention in the solution to prevent harm. This enables you to manage situations when the AI model does not perform as required.

- **Security:** Ensure that your solution is secure and has adequate controls to preserve the integrity of your content and prevent unauthorized access.
- **Build trust with affected stakeholders:** Communicate the expected benefits and potential risks to affected stakeholders. Help people understand why the data is needed and how the use of the data will lead to their benefit. Describe data handling in an understandable way.
- **Customer feedback loop:** Provide a feedback channel that allows users and individuals to report issues with the service once it's been deployed. After you've deployed an AI-powered product or feature, it requires ongoing monitoring and improvement. Be ready to implement any feedback and suggestions for improvement. Establish channels to collect questions and concerns from affected stakeholders (people who might be directly or indirectly affected by the system, including employees, visitors, and the general public).
- **Feedback:** Seek out feedback from a diverse sampling of the community during the development and evaluation process (for example, from historically marginalized groups, people with disabilities, and service workers). See: [Community jury](#).
- **User study:** Any consent or disclosure recommendations should be framed in a user study. Evaluate the first and continuous-use experience with a representative sample of the community to validate that the design choices lead to effective disclosure. Conduct user research with 10-20 community members (affected stakeholders) to evaluate their comprehension of the information and to determine if their expectations are met.

Recommendations for preserving privacy

A successful privacy approach empowers individuals with information and provides controls and protection to preserve their privacy.

Consent to process and store audio input: Be sure to have all necessary permissions from your end users before you use the speech to text-enabled features in your applications or devices. Also ensure that you have permission for Microsoft to process this data as your third-party cloud service processor. Note that the real-time API does not separately store any of the audio input and transcription output data. However, you can design your application or device to retain end-user data, such as transcription text. You have an option to turn on local data logging via the Speech SDK (see [Enable logging in the Speech SDK](#)).

Next steps

- [Data, privacy, and security for speech to text](#)

Data and Privacy for Speech to text

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Note

This article is provided for informational purposes only and not for the purpose of providing legal advice. We strongly recommend seeking specialist legal advice when implementing Speech Services.

This article provides some high-level details regarding how speech to text processes data provided by customers. Note that audio data of humans speaking and the related text transcripts may be considered personal data and/or sensitive data under various privacy regulations and laws because it contains not only the voice of humans, but the content of the audio may also contain personal information depending on the context within which the audio was collected. Audio data and the related text transcripts may also be regulated under various communications laws or other law and regulations. As an important reminder, you are responsible for the implementation of this technology and are required to obtain all necessary permissions for processing of the data, as well as any licenses, permissions or other proprietary rights required for the content you input into the speech to text service. It is your responsibility to comply with all applicable laws and regulations in your jurisdiction.

What data does speech to text process?

Speech to text processes the following types of data:

- **Audio input or voice audio:** All speech to text features accept voice audio as an input that is streamed through the Speech SDK/REST API into the service endpoint. In batch transcription, audio input will be sent to a storage location instructed by the customer, and the Speech service accesses and processes the audio input for the purposes of providing the transcription services requested. See more information about how to specify storage in [How to use batch transcription](#).
- **Input transcription text:** In the pronunciation assessment, transcribed text is sent together with an input voice audio as "correct" text. Pronunciations are assessed based on the input transcriptions.

- **Transcription for speech translation:** When the speech translation feature is used, transcribed text that speech to text generated is translated into a specified language through the [Translator service](#).

The text translation service is used only to convert text from one language to another. No input/output data is retained by Speech service after the completion of a translation request. See [What is the Translator service](#) for more information about the text translation service.

If users need transcribed/translated text in an audio format, the feature sends the output text to [text to speech](#). Again, no data is persisted in the text to speech data processing.

How does speech to text process data?

Real-time speech to text

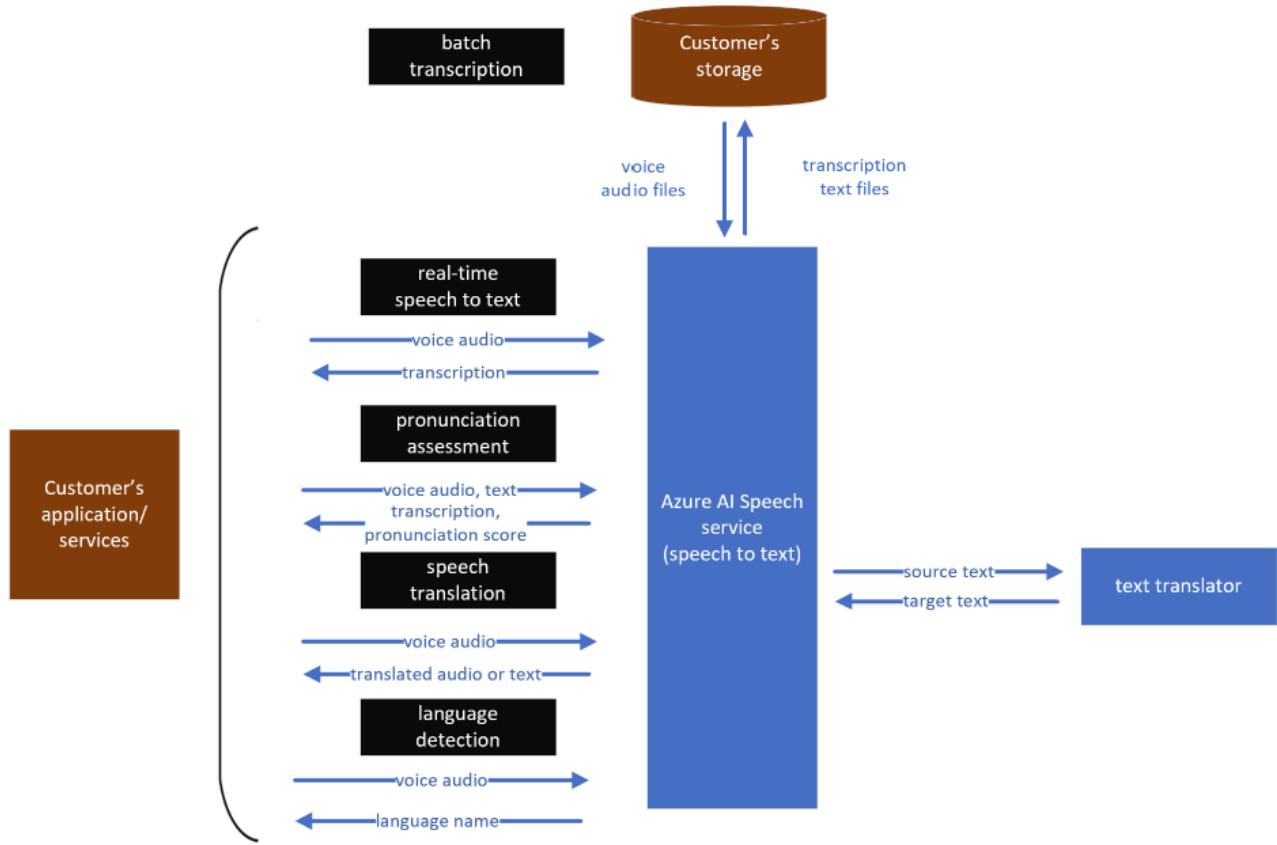
When a client application sends audio input to speech to text, the speech recognition engine parses audio and converts it to text. Relying upon its acoustic and linguistic or language understanding features, speech to text selects candidate words and phrases that may be uttered in the audio input. The transcription output represents the best inference or prediction in text format of what was spoken in the audio input.

For real-time speech to text, audio input is processed only on the Azure's server memory, and no data is stored at rest. All data in-transit are encrypted for protection. See [Trusted Cloud: security, privacy, compliance, resiliency, and IP](#) for more information about Azure-wide security and privacy protection.

Batch transcription

In batch transcription, customers specify their chosen storage location of both audio input and output transcription text files for Speech service to access, process, and provide the transcription output. The customer controls the storage of this data, including the retention of such data. Customers may set a retention time for generated transcription text files by using a parameter called "timeToLive". See [Batch Transcription -- Configuration Properties](#) for more detail.

See the data flows for each Speech to text feature:



Speaker diarization/separation

This feature is available for both real-time and batch API. When customers enable the speaker separation (diarization) option (disabled by default), the speech to text engine analyzes and extracts unique voice characteristics signals from the audio input to differentiate the audio between speakers. These voice characteristics signals are used and temporarily retained for the sole purpose of annotating the transcription output with markers next to text for Speaker 1 (Guest-1) or Speaker 2 (Guest-2). Upon completion of the process, all signal data used to separate the speakers is discarded. The speaker separation feature supports the separation of two or more speakers in a single audio file. Speaker Separation does not support speaker identity recognition enrollment or the ability to track unique speakers across multiple audio files.

Language detection

Language detection is similar to speech recognition except that the model calculates probabilities of mapping between phonemes and languages. Each language has specific phonemes and phoneme combinations, which characterize the language. The language detection model identifies the characteristics in phonemes to calculate likelihood of languages used in an input voice.

Speech translation

When speech translation is used, first, an audio input is used to generate machine-transcribed text with speech to text. Then the machine-transcribed text is sent to the text translation service to convert the text (in the source language) to another language. If customers need translated text in an audio format, this feature can send the translated text to [text to speech](#). Customers have the option to produce translated text only or translated voice output.

Speech containers

With speech containers, customers deploy Speech services APIs to their own environment through Docker containers. Since all speech components run on customers' controlled environment, audio data inputs and transcription outputs are processed within customers' container and is not sent to the cloud based Speech service. See [Install and run Docker containers for the Speech service APIs](#) for more information.

Security for customers' data in speech container

The security of customer data is a shared responsibility. Details on the security model of Azure AI containers, like the speech container can be found in [Azure AI Services container security](#).

You are responsible for securing and maintaining the equipment and infrastructure required to operate speech containers located on your premises, such as your edge device and network.

To learn more about Microsoft's privacy and security commitments visit [the Microsoft Trust Center](#).

Data storage and retention

No data trace

When doing real-time speech to text, pronunciation assessment and speech translation, Microsoft does not retain or store the data provided by customers. In batch transcription, customers specify their own storage locations to send the audio input. Generated transcription text may be stored either in customer's own storage or Microsoft storage if no storage is specified. If output transcriptions are stored in Microsoft storage, customers may delete the data either by calling a deletion API or setting the timeToLive parameter to automatically delete the data in a specified time. See more details in [How to use batch transcription - Speech service - Azure AI services](#).

To learn more about Microsoft's privacy and security commitments visit the Microsoft [Trust Center](#).

Transparency note: text to speech

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

What is a Transparency Note?

An AI system includes not only the technology, but also the people who will use it, the people who will be affected by it, and the environment in which it is deployed. Creating a system that is fit for its intended purpose requires an understanding of how technology works, what its capabilities and limitations are, and how to achieve the best performance. Microsoft's Transparency Notes are intended to help you understand how our AI technology works, the choices system owners can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Transparency Notes when developing or deploying your own system or share them with the people who will use or be affected by your system.

Microsoft's Transparency Notes are part of a broader effort at Microsoft to put our AI Principles into practice. To find out more, see [the Microsoft AI principles](#).

The basics of text to speech

Introduction

[Text to speech](#), part of Azure AI Speech, is a versatile tool that can convert written text into natural-sounding speech audio. The feature takes input in the form of text and generates high-quality speech audio output that can be played on devices. For the speech audio output, text to speech offers a range of prebuilt neural voices or, for Limited Access customers, the option to create a custom neural voice for your product or brand.

Text to speech also has visual capabilities. Using text to speech avatar, customers can input text and create a synthetic video of an avatar speaking. Both prebuilt text to speech avatars and custom text to speech avatars are available, which can be used with both prebuilt neural voice and custom neural voice, though some features are only available for Limited Access customers.

In a text to speech system, customers can turn written information into audible speech and improve accessibility for users. Whether listening to documents or enhancing user experiences with synthesized speech, text to speech transforms text into natural-sounding spoken words.

Key terms

[] [Expand table](#)

Term	Definition
Real-time speech synthesis	Use the Speech SDK or REST API to convert text to speech by using prebuilt neural voice , prebuilt text to speech avatar , custom neural voice , and custom text to speech avatar .
Voice model	In a text to speech system, a voice model refers to a machine learning-based model or algorithm that generates synthetic speech from written text. This model is trained to convert text input into spoken language output, mimicking the characteristics of a human voice, including pitch, tone, and pronunciation.
Prosody	Prosody refers to the modulation of speech elements such as pitch, duration, volume, and pauses to infuse synthetic voices with a natural and expressive quality, conveying emotional nuances and contextual meaning, thereby reducing the robotic quality of the generated speech, and making it more engaging and comprehensible to listeners.
Speech Synthesis Markup Language ("SSML")	Speech Synthesis Markup Language (SSML) is an XML-based markup language that's used to customize text to speech outputs. With SSML, you can adjust pitch, add pauses, improve pronunciation, change speaking rate, adjust volume, and attribute multiple voices to a single document. You can use SSML to define your own lexicons or switch to different speaking styles.
Asynchronous synthesis of long audio	Use the batch synthesis API (preview) to asynchronously synthesize text to speech files longer than 10 minutes (for example, audio books or lectures). Unlike synthesis performed via the Speech SDK or Speech to text REST API, responses aren't returned in real-time. The expectation is that requests are sent asynchronously, responses are polled for, and synthesized audio is downloaded when the service makes it available.
Visemes	Visemes are the key poses in observed speech, including the position of the lips, jaw, and tongue in producing a particular phoneme. Visemes have a strong correlation with voices and phonemes.

Prebuilt neural voice

Introduction

Prebuilt neural voice provides a wide range of voices, offering over 400 options in more than 140 languages and locales. These text to speech voices enable you to quickly integrate read-aloud functionality into your applications for enhanced accessibility.

Key terms

 Expand table

Term	Definition
Prebuilt neural voice	Microsoft offers a set of prebuilt neural voices, which use deep neural networks to overcome the limits of traditional speech synthesis with regard to stress and intonation in spoken language. Prosody prediction and voice synthesis happen simultaneously, which results in more fluid and natural-sounding outputs. Each prebuilt neural voice model is available at 24kHz and high-fidelity 48kHz, and the output can be upsampled or downsampled to other formats.

Capabilities

System behavior

Text to speech

Text to speech converts text into natural-sounding speech.

Below are the main options for calling the text to speech service.

Real-time text to speech API

This is a common API call via the [Speech SDK](#) or [REST API](#) to send a text input and receive an audio output in real time. The Speech system uses a text to speech voice model to convert the text into human-like synthetic speech. The output audio can be saved as a file or be played back to an output device such as a speaker (learn more about [how to synthesize speech from text](#)). Users can also use [SSML](#) to fine-tune the text to speech output.

Text to speech models are trained on large amounts of diverse audio across typical usage scenarios and a wide range of speakers. For example, the text to speech service is often used for voice-enabled chat bots or for audio content creation.

Batch synthesis API

Batch synthesis is another type of API call. It's typically used to send large text files and to receive audio outputs asynchronously (that is, at a later time). To use this API, you can specify locations for multiple text files. The text to speech technology reads the text input from the file and generates audio files that are returned to the storage location that you specify. This feature is used to support larger speech synthesis jobs in which it is not necessary to provide end users with the audio output in real time. An example is to create audio books.

Text to speech – custom neural voice

Custom neural voice is a [text to speech](#) feature that allows Limited Access customers to create a one-of-a-kind customized synthetic voice for their applications by providing their own audio data of customer's selected voice talents.

With custom neural voice, you can record your voice talent by having them read Microsoft-provided scripts in the Speech Studio and quickly create a synthetic voice that sounds like your voice talent using a lite project (Preview). A lite project is ideal for a quick trial or a proof of concept.

With a pro project, you can upload studio-recorded high-quality voice data of your selected voice talent and create a realistic-sounding voice. Pro supports highly natural voice training that even more closely resembles your voice talent's voice and can be adapted to speak in multiple emotions and across languages, without the need for additional emotion-specific or language-specific training data.

Once a custom neural voice is created, you can deploy the voice model with a unique endpoint and use the model to generate synthetic speech with the real-time synthesis API or the batch synthesis API described above.

For more information on custom neural voice, see [Overview of custom neural voice](#).

Personal voice

The personal voice feature allows Limited Access customers to create a voice model from a short human voice sample. The feature can create a voice model based on the prompt in as little as a few seconds. This feature is typically used to power personalized voice experiences for business customers' applications. Personal voice models can create realistic-sounding voices that can speak in close to 100 languages.

[Watermarks](#) are added to custom neural voices created with the personal voice feature. Watermarks allow users to identify whether speech is synthesized using Azure AI Speech, and specifically, which voice was used. Eligible customers can use Azure AI Speech watermark

detection capabilities. To request to add watermark detection to your applications please contact [mstts\[at\]microsoft.com](mailto:mstts[at]microsoft.com).

For more information on personal voice, see [personal voice](#).

Text to speech avatar

Text to speech avatar converts text into a digital video of a photorealistic human (either a prebuilt avatar or a custom avatar) speaking with a natural-sounding voice powered by text to speech features like prebuilt neural voice or custom neural voice. The text to speech avatar video can be synthesized asynchronously or in real time. Developers can build applications integrated with text to speech avatar through an API, or use a content creation tool on Speech Studio to create video content without coding.

With text to speech avatar's advanced neural network models, the feature empowers users to deliver life-like and high-quality synthetic talking avatar videos for various applications.

Text to speech avatar adopts Coalition for Content Provenance and Authenticity (C2PA) Standard to provide audiences with clearer insights into the source and history of video content created by avatars. This standard offers transparent information about AI-generation of video content. For more details on the integration of C2PA with text to speech avatars, refer to [Content Credentials in Azure Text to Speech Avatar](#).

In addition, avatar outputs are automatically watermarked. Watermarks allow approved users to identify whether a video is synthesized using the avatar feature of Azure AI Speech. To request watermark detection, please contact [avatarvoice\[at\]microsoft.com](mailto:avatarvoice[at]microsoft.com).

Video translation (preview)

Video translation can efficiently localize your video content to cater to diverse audiences around the globe. Video translation will automatically extract dialogue audio, transcribe, translate and dub the content with prebuilt or personal voice to the target language, with accurate subtitles for better accessibility. Multi-speaker features will help identify the number of individuals speaking and recommend suitable voices. Content editing with human in the loop allows for precise alignment with customer preference. Enhanced translation quality ensures precise audio and video alignment with GPT integration. Video translation enables authentic and personalized dubbing experiences with personal voice.

Use cases

Text to speech offers a variety of features catering to a wide range of intended uses across industries and domains. All text to speech features including video translation are subject to the terms and conditions applicable to customers' Azure subscription, including the Azure Acceptable Use Policy and the [Code of conduct for Azure AI Speech text to speech](#).

In addition, custom text to speech features like custom neural voice, personal voice, and custom text to speech avatar are limited to the approved use cases as outlined in the specific scenarios described below:

Intended uses for Custom Neural Voice Pro and Custom Neural Voice Lite

The following are the approved use cases for Custom Neural Voice Pro and Custom Neural Voice Lite:

- **Educational or interactive learning:** To create a fictional brand or character voice for reading or speaking educational materials, online learning, interactive lesson plans, simulation learning, or guided museum tours.
- **Media: Entertainment:** To create a fictional brand or character voice for reading or speaking entertainment content for video games, movies, TV, recorded music, podcasts, audio books, or augmented or virtual reality.
- **Media: Marketing:** To create a fictional brand or character voice for reading or speaking marketing and product or service media, product introductions, business promotion, or advertisements.
- **Self-authored content:** To create a voice for reading content authored by the voice talent.
- **Accessibility Features:** For use in audio description systems and narration, including any fictional brand or character voice, or to facilitate communication by people with speech impairments.
- **Interactive Voice Response (IVR) Systems:** To create voices, including any fictional brand or character voice, for call center operations, telephony systems, or responses for phone interactions.
- **Public Service and Informational Announcements:** To create a fictional brand or character voice for communicating public service information, including announcements for public venues, or for informational broadcasts such as traffic, weather, event information, and schedules. This use case is not intended for journalistic or news content.
- **Translation and Localization:** For use in translation applications for translating conversations in different languages or translating audio media.
- **Virtual Assistant or Chatbot:** To create a fictional brand or character voice for smart assistants in or for virtual web assistants, appliances, cars, home appliances, toys, control of IoT devices, navigation systems, reading out personal messages, virtual companions, or customer service scenarios.

Intended uses for personal voice

The personal voice API (see [Personal voice](#) for more information) is available in Limited Access preview. Only customers who meet Limited Access eligibility criteria can integrate the personal voice API with their applications. These eligible customers are permitted to use personal voices for the following use cases only:

- **Applications:** For use in applications where voice output is constrained and defined by customers, and where the voice does not read user-generated or open-ended content. Voice model usage must remain within the application and output must not be publishable or shareable from the application. Some examples of applications that fit this description are voice assistants in smart devices and customizing a character voice in gaming.
- **Media, films, and TV:** To dub for films, TV, video, and audio for entertainment scenarios only, where customers maintain sole control over the creation of, access to, and use of the voice models and their output.
- **Business content:** To create audio and video content for business scenarios to communicate product information, marketing materials, business promotional content, and internal business communications.
- **Special use, bundled with video translation:** To synthesize voices for each speaker in a video. Customers can also edit and generate lip-synced audio content in target languages. Customers are not required to submit to Microsoft additional [audio consent](#) for video content in this scenario, but customers must maintain sole control over the creation, access to, and use of the voice models and their outputs.

All other uses of custom neural voice, including Custom Neural Voice Pro, Custom Neural Voice Lite, and personal voice, are prohibited. In addition, custom neural voice is a Limited Access service, and registration is required for access to this service. To learn more about Microsoft's Limited Access policy, refer to [Limited Access features for Azure AI services](#). Certain features are only available to Microsoft managed customers and partners, and only for certain use cases approved by Microsoft at the time of registration.

Prebuilt neural voice may also be used for the custom neural voice use cases above, as well as additional use cases selected by customers and consistent with the Azure Acceptable Use Policy and the [Code of conduct for Azure AI Speech text to speech](#). No registration or pre-approval is required for additional use cases for prebuilt neural voice that meet all applicable terms and conditions.

Intended use cases for video translation (preview)

Video translation could be used for films, TV, and other visual (including but not limited to video or animation) and audio applications, where customers maintain sole control over the

creation of, access to, and use of the voice models and their output. Personal voice and lip syncing are subject to the Limited Access framework, and eligible customers may use these capabilities with Video translation. The following are the approved use cases for Video translation service:

- **Education & learning:** To translate audio in educational visuals, online courses, training modules, simulation-based learning, or guided museum tour visuals for multilingual learners.
- **Media: Entertainment:** To translate audio in films, movies, TV shows, documentaries, video games, mini-series, short-play and AR/VR content for global audiences, ensuring seamless storytelling across languages.
- **Media: Marketing:** To translate audio in promotional visuals, product demos, advertisements, and branding campaigns to resonate with international markets and cultures.
- **Self-Authored Content:** To translate audio in vlogs, short-form visuals, influencer content, travel guides, destination promotional videos, social media visuals, and cultural highlight reels making them accessible and engaging.
- **Corporate Training and Communication:** To translate audio in internal communication visuals, employee onboarding materials, compliance training, and global corporate announcements for international teams.
- **E-commerce & Product Demonstrations:** To translate audio in product unboxing visuals, tutorials, customer testimonials, and explainer visuals to cater to international shoppers.
- **Public Service and Informational Announcements:** To translate audio in public awareness visuals, event schedules, safety announcements, and government informational broadcasts for multilingual accessibility.
- **Accessibility Features:** To broaden the accessibility of video content through multilingual audio and subtitles.
- **News and Journalistic Content:** To translate audio in news segments, interviews, press releases, and breaking news reports for diverse linguistic audiences. Customers looking to translate news sources will require additional review.

Intended uses for custom text to speech avatar and prebuilt text to speech avatar

The following are the approved use cases for custom text to speech avatar:

- **Virtual Assistant or Chatbot:** To create virtual assistants, virtual companions, virtual sales assistants, or for customer service applications.
- **Content generation for enterprise contexts:** For use to communicate product information, marketing materials, business promotional content, and internal business

communications. Examples include character avatars or digital twins of a business leader to promote a brand.

- **Educational or interactive learning:** To create a fictional brand or character avatar for presenting educational materials, online learning, interactive lesson plans, simulation learning, or guided museum tours.
- **Media: Entertainment:** To present updates, share knowledge, create interactive media, or make talking head videos for entertainment scenarios such as videos, gaming, and augmented or virtual reality.
- **Accessibility Features:** For use to facilitate communication by people with speech impairments.
- **Self-authored content:** To create an avatar for reading content authored by the avatar talent.
- **Public Service and Informational Announcements:** To create a fictional brand or character image for communicating public service information, including announcements for public venues, or for informational broadcasts such as traffic, weather, event information, and schedules. This use case is not intended for journalistic or news content.
- **Translation and Localization:** For use in translation applications for translating conversations in different languages or translating audio media in video format.

All other uses of custom text to speech avatar are prohibited. In addition, custom text to speech avatar is a Limited Access service, and registration is required for access to this feature. To learn more about Microsoft's Limited Access policy visit aka.ms/limitedaccesscogservices. Certain features are only available to Microsoft managed customers and partners, and only for certain use cases approved by Microsoft at the time of registration.

Prebuilt text to speech avatar may also be used for the custom avatar use cases above, as well as additional use cases selected by customers and consistent with Azure Acceptable Use Policy and the [Code of conduct for Azure AI Speech text to speech](#). No registration or pre-approval is required for additional use cases for prebuilt text to speech avatar that meet all applicable terms and conditions.

Considerations when choosing use cases

We encourage customers to use text to speech features in their innovative solutions or applications. All text to speech features must adhere to the Azure Acceptable Use Policy and the [Code of conduct for Azure AI Speech text to speech](#). In addition, custom neural voice and custom text to speech avatars may only be used for the use cases approved through the [Limited Access registration form](#). Additionally, here are some considerations when choosing a use case for any text to speech feature:

- **Ensure use case alignment:** Ensure that the intended use of any text to speech feature aligns with the capabilities and intended purpose of the text to speech feature.

- **Responsible AI considerations:** Prioritize responsible AI practices by avoiding the creation of misleading or harmful content. Adhere to privacy, data protection, and legal regulations when using text to speech features.
- **Review the code of conduct:** Microsoft has established a code of conduct that prohibits certain uses of all text to speech features. Ensure compliance with the code of conduct when selecting a use case for text to speech services.
- **Exercise editorial control:** Carefully consider using synthetic voices with content that lacks proper editorial control, as synthetic voices can sound human-like and amplify the effect of incorrect or misleading content.
- **Disclosure:** Disclose the synthetic nature of voices, images, and/or videos to users such that users are not likely to be deceived or duped—or able to prank others—into believing they are interacting with a real person.
- **Legal and regulatory considerations:** Organizations need to evaluate potential specific legal and regulatory obligations when using any AI services and solutions, which may not be appropriate for use in every industry or scenario. Additionally, AI services or solutions are not designed for and may not be used in ways prohibited in applicable terms of service and relevant codes of conduct.

By adhering to these considerations, users can leverage both prebuilt and custom neural voice responsibly.

Limitations

The limitations of text to speech should be considered at the intersection of technology and the human, social, and organizational factors that influence its usage and impact. While text to speech offers advanced speech synthesis capabilities, there are certain limitations to be aware of in deploying it responsibly to minimize potential errors.

Technical limitations, operational factors, and ranges

Technical limitations to consider when using text to speech include the accuracy of pronunciation and intonation. While text to speech is designed to generate natural-sounding speech, it may encounter difficulties with certain words, names, or uncommon phrases. Users should be aware that there can be instances where the system may mispronounce or emphasize words incorrectly, especially when dealing with niche or domain-specific vocabulary.

It is important to note that certain populations may be more negatively impacted by these technical limitations. For example, individuals with hearing impairments who rely heavily on synthesized speech may face challenges in understanding unclear or distorted speech output. Similarly, users with cognitive or language-related disabilities may find it difficult to comprehend speech with unnatural intonation or mispronounced words.

- **Linguistic limitations:** While we carefully curate and prepare training data to minimize biases, especially related to gender, ethnicity, or regional accents, and while text to speech supports multiple languages and accents, there may be variations in the quality and availability of voices across different languages. Customers should be aware of potential limitations in pronunciation accuracy, intonation, and linguistic nuances specific to certain languages or dialects.
- **Context and emotion:** Text to speech may have limitations in accurately conveying contextual information and emotions. Customers should be mindful of the system's inability to understand the emotional nuances or subtle cues present in the input text. Considerations should be made to provide additional context or utilize other methods to convey emotions effectively.
- **Availability:** Microsoft will provide customers with 12 months' notice before removing any prebuilt neural voices from our catalog, unless security, legal, or system performance considerations require an expedited removal. This does not apply to previews.

Each application is different, and our base model may not match your context or cover all scenarios required for your use case. We encourage developers to thoroughly evaluate the quality of text to speech synthetic voice and video with real-world data that reflects your use case, including testing with users from different demographic groups and with different speech characteristics. Please see the [Quality of the voice model trained](#) section for best practices for building high quality voice models.

In addition to ensuring performance, it is important to consider how to minimize risks of stereotyping and erasure that may result from synthetic voices and avatar. For example, if you are creating a custom neural voice for a smart voice assistant, carefully consider what voice is appropriate to create, and seek diverse perspectives from individuals from a variety of backgrounds. When building and evaluating your system, always seek diverse input.

Fairness considerations

At Microsoft, we strive to empower every person on the planet to do more. An essential part of this goal is working to create technologies and products that are fair and inclusive. Fairness is a multi-dimensional, socio-technical topic and impacts many different aspects of our product development. You can learn more about Microsoft's approach to fairness [here](#).

One important dimension to consider when using AI systems, including text to speech, is how well the system performs for different groups of people. Research has shown that without

conscious effort focused on improving performance for all groups, AI systems can exhibit varying levels of performance across different demographic factors such as race, ethnicity, gender, and age.

As part of our evaluation of Azure AI text to speech, we have conducted an analysis to assess potential fairness harms. We have examined the system's performance across different demographic groups, aiming to identify any disparities or differences that may exist and could potentially impact fairness.

In some cases, there may be remaining performance disparities. It is important to note that these disparities may exceed the target, and we are actively working to address and minimize any potential biases or performance gaps, carefully consider the demographic group choice of the actor, and seek diverse perspectives from a variety of backgrounds.

Regarding representational harms, such as stereotyping, demeaning, or erasing outputs, we acknowledge the risks associated with these issues. While our evaluation process aims to mitigate such risks, we encourage users to consider their specific use cases carefully and implement additional mitigations as appropriate. Having a human in the loop can provide an extra layer of oversight to address any potential biases or unintended consequences. The use of blocklists or allowlists can also help ensure that the synthesized speech aligns with desired standards and avoids any harmful or inappropriate content.

We are committed to continuously improving our fairness evaluations to gain a deeper understanding of the system's performance across various demographic groups and potential fairness concerns. The evaluation process is ongoing, and we are actively working to enhance fairness and inclusivity, and mitigate any identified disparities. We understand the importance of addressing fairness considerations and strive to ensure that text to speech delivers reliable and equitable synthesized speech outputs.

Please note that this information represents what we know so far about fairness evaluations, and we remain dedicated to refining our evaluation methodologies and addressing any fairness concerns that may arise.

System performance

Performance for the text to speech system refers to how accurately and naturally it can convert written text into synthesized speech. This is measured using various metrics to evaluate the quality and effectiveness of the generated audio output. Some common performance metrics used include:

- **Mean opinion score (MOS):** A rating system where judges provide a score that represents the overall quality of synthesized speech and avatar video. A higher MOS indicates better

quality.

- **MOS gap:** The difference between the MOS score of human recordings and the generated audio tracks/videos. A smaller MOS Gap indicates a closer resemblance to human speech/human likeness.
- **Similarity MOS (SMOS):** Measures the similarity of the generated audio tracks/videos to the human recordings. A higher SMOS signifies better similarity.
- **Intelligibility:** The percentage of correctly intelligible words in synthesized speech.

Even with state-of-the-art models, AI systems like text to speech can produce errors. For example, the system may produce synthesized speech with subtle unnatural intonations or pronunciation errors, leading to a less-than-ideal user experience, or the system may misinterpret text or struggle with unusual linguistic constructs, resulting in unnatural or unintelligible speech.

Best practices for improving system performance

To improve system performance and adapt system behavior in text to speech, there are several best practices that can be followed. These practices involve adjusting various components and parameters to optimize the tradeoffs and meet specific use case requirements. However, it is important to consider the potential impact on different populations to ensure fairness and inclusivity.

Prebuilt neural voice

Using SSML (Speech Synthesis Markup Language) is considered a best practice to enhance text to speech output quality. SSML allows users to exert greater control over synthesized speech, enabling the customization of pronunciation, intonation, emphasis, and other prosodic features. By incorporating SSML tags into the text, users can add pauses, adjust speech rate, specify phonetic pronunciations, and control pitch and volume, among other parameters. This level of fine-tuning helps create more natural and expressive speech, making the text to speech output sound more human-like and engaging. All the SSML markups can be passed directly to the API. We also provide an online tool, Audio Content Creation, that allows customers to fine-tune using an intuitive user interface.

If your use case involves specialized vocabulary or domain-specific content, consider using the custom lexicon feature to improve the system's ability to accurately pronounce and convey domain-specific terms or phrases.

Evaluation of text to speech

Evaluation methods

Some commonly used metrics for evaluating text to speech overall system performance include:

- Mean opinion score (MOS) gap with human recording: usually used to compare the quality of the text to speech voice model against a human recording. The quality of a voice model created by custom neural voice compared to that of a human recording is expected to be close, with a gap of no more than 0.5 in the MOS score.
- For custom neural voice, you also can use Similarity MOS (SMOS) to measure how similar the custom voice sounds compared to the original human recordings. With SMOS studies, judges are asked to listen to a set of paired audio tracks, one generated using the custom voice, the other from the original human recordings in the training data, and rate if the two audio tracks in each pair are spoken by the same person, using a five-point scale (1 being the lowest, 5 the highest). The average score is reported as the SMOS score. We recommend that a good custom neural voice should achieve an SMOS higher than 4.0.
- Besides measuring naturalness with MOS and SMOS, you can also assess the intelligibility of the voice model by checking the pronunciation accuracy of the generated speech. This is done by having judges listen to a set of testing samples, determining whether they can understand the meaning and indicate any words that were unintelligible to them. Intelligibility rate is calculated using the percentage of the correctly intelligible words among the total number of words tested (i.e., the number of intelligible words/the total number of words tested * 100%). Normally a usable text to speech engine needs to reach a score of > 98% for intelligibility.

Evaluation results

Text to speech consistently delivers high-quality and natural-sounding synthesized speech, meeting the requirements of diverse industries and domains. Our evaluations include extensive testing of the system's training and test data, ensuring that it represents the intended uses and operational factors encountered in real-world scenarios, as well as testing samples of synthesized speech outputs.

The evaluation results have influenced decisions about the constraints in the system's design, such as the maximum case size and the minimum amount of training data required. By analyzing the performance of the system across different data sets, settings, and parameters, appropriate constraints have been set to optimize the system's behavior, reliability, and safety.

While the evaluation covers a wide range of use cases, it is important to note that the results are generalizable to some extent across use cases that were not directly part of the evaluation. The robustness and performance of the system provides confidence in its ability to handle various scenarios, including those that may not have been explicitly tested.

Here are some recommended tests and score ranges based on our experience:

[+] Expand table

Measurement	Definition	How it is calculated	Recommended text size	Recommended score
MOS	Mean opinion score of the quality of the audio tracks	Average of the rating scores of each judge on each audio	> 30 generated audio tracks	> 4.0 (normally requires the MOS of the human recording is higher than 4.5)
MOS gap	The MOS score difference between human recordings and the generated audio tracks	The MOS score on the human recordings minus the MOS score on the generated audio tracks	> 10 human recordings, > 30 generated audio tracks, > 20 judges on each audio	< 0.5
SMOS	The similarity of the generated audio tracks to the human recordings	Average of the rating scores of the similarity level on each pair of audio tracks	> 40 pairs, > 20 judges on each pair	> 4.0, > 3.5 (secondary language)
Intelligibility	The pronunciation accuracy of the generated speech at the word level	Percentage of the correctly intelligible words among the total number of words tested	> 60 generated audio tracks, > 10 judges on each audio	> 98%

Evaluating and integrating text to speech for your use

Below are some best practices to help you responsibly integrate text to speech features into your use cases.

Disclose when the voice is synthetic

Disclosing that a voice is computer generated not only minimizes the risk of harmful outcomes from deception but also increases trust in the organization delivering the voice. Learn more about [how to disclose](#).

Microsoft requires its customers to disclose the synthetic nature of text to speech voices to its users.

- Make sure to provide adequate disclosure to audiences, especially when using the voice of a well-known person. People make judgments about information based in part on the person who delivers it, whether they do so consciously or unconsciously. For example, a disclosure could be verbally shared at the start of a broadcast. For more information, visit [disclosure patterns](#).
- Consider proper disclosure to parents or other parties with use cases that are designed for or may be used in situations involving minors and children. If your use case is intended for minors or children, you'll need to ensure that your disclosure is clear and transparent so that parents or legal guardians can understand the role of synthetic media and make an informed decision on behalf of minors or children about whether to use the experience.

Disclose when the avatar video is synthetic

Disclosing that an avatar speaking video is computer generated not only minimizes the risk of harmful outcomes from deception but also increases trust in the organization delivering the video. Learn more about [how to disclose](#).

Microsoft requires its customers to disclose the synthetic nature of text to speech avatars to its users.

- Make sure to provide adequate disclosure to audiences, especially when using the image (and voice) of a well-known person. People make judgments about information based in part on the person who delivers it, whether they do so consciously or unconsciously. **For example, a disclosure could be made with a watermark, such as, "The voice and image in this video are AI-generated," in text or verbally shared at the start of a video.** For more information visit [disclosure patterns](#).
- Consider proper disclosure to parents or other parties with use cases that are designed for or may be used in situations involving minors and children. If your use case is intended for minors or children, you'll need to ensure that your disclosure is clear and transparent so that parents or legal guardians can understand the role of synthetic media and make an informed decision on behalf of minors or children about whether to use the experience.

Select appropriate voice types for your scenario

Carefully consider the context of use and the potential harms associated with using text to speech voices or avatars. For example, high-fidelity synthetic voices may not be appropriate in

high-risk scenarios, such as for personal messaging, financial transactions, or complex situations that require human adaptability or empathy.

Users may also have different expectations for voice types and avatar expressions or gestures, depending on the context. For example, when listening to sensitive news read by a synthetic voice, some users prefer a more empathetic and human-like tone, while others prefer a neutral voice. Consider testing your application to better understand user preferences.

Be transparent about capabilities and limitations

Users are more likely to have higher expectations when interacting with high-fidelity synthetic voice agents. When system capabilities don't meet those expectations, trust can suffer, and may result in unpleasant, or even harmful experiences.

Provide optional human support

In ambiguous, transactional scenarios (for example, a call support center), users don't always trust a computer agent to appropriately respond to their requests. Human support may be necessary in these situations, regardless of the realistic quality of the voice or capability of the system.

Considerations for voice talent

When customers work with voice talent to create custom neural voice, the guidelines below apply.

- Voice talent should have control over their voice model (how and where it will be used) and be compensated for its use. Microsoft requires custom neural voice customers to obtain explicit written permission from voice talent to create a synthetic voice and to ensure that the customer's agreement with each individual contemplates the duration, use, and any content limitations. **If you are creating a synthetic voice of a well-known person, you should provide a way for the voice talent to edit or approve the content of the output you plan to generate with the voice model.**
- Some voice talent may be unaware of potential malicious uses of technology and should be educated by system owners about the capabilities of the technology. Microsoft requires customers to share Microsoft's [Disclosure for voice and avatar talent](#) with voice talent directly or through the voice talent's authorized representative to describe how synthetic voices are developed and operate in conjunction with text to speech services.

Considerations for avatar talent

When customers work with avatar talent to create custom avatars, the guidelines below apply.

- Avatar talent should have control over their avatar model (how and where it will be used) and be compensated for its use. Microsoft requires custom avatar customers to obtain explicit written permission from their avatar talent to create a synthetic text to speech avatar and ensure that the customer's agreement with each individual contemplates the duration, use, and any content limitations. **If you are creating a custom avatar of a well-known person, you should provide a way for the avatar talent to edit or approve the content of the output you plan to generate with the voice model.**
- Some avatar talent may be unaware of potential malicious uses of technology and should be educated by system owners about the capabilities of the technology. Microsoft requires customers to share Microsoft's [Disclosure for voice and avatar talent](#) with avatar talent directly or through the avatar talent's authorized representative to describe how synthetic avatar video is developed and operates in conjunction with text to speech services.

Considerations for people with speech disorders

When working with individuals with speech disorders to create or deploy synthetic voice technology, the following guidelines apply.

Provide guidelines for contracts with talent in accessibility scenarios

Customers should develop guidelines for establishing contracts with individuals who use synthetic voices for assistance in speaking. Customers should consider specifying in their contracts with individuals the duration of use, ownership transfer and/or license criteria, procedures for deleting the voice model, and how to prevent unauthorized access.

Account for inconsistencies in speech patterns

For individuals with speech disorders who record their own voice fonts, inconsistencies in their speech pattern (slurring or inability to pronounce certain words) may complicate the recording process. In these cases, synthetic voice technology and recording sessions should be designed with appropriate accommodations determined by the customer (for example, provide breaks or additional recording sessions).

Allow modification over time

Individuals with speech disorders may wish to update their synthetic voice to reflect changes due to aging or other factors. Individuals may also have stylistic preferences that change over time, and may want to make changes to pitch, accent, or other voice characteristics.

Learn more about responsible AI

- [Microsoft AI principles ↗](#)
- [Microsoft responsible AI resources ↗](#)
- [Microsoft Azure Learning courses on responsible AI](#)

Learn more about Azure Speech

- [Limited access to Azure Speech Service - Azure AI services | Microsoft Learn](#)
- [Code of conduct for Azure Speech Service | Microsoft Learn](#)
- [Data, privacy, and security for Azure Speech Service - Azure AI services | Microsoft Learn](#)

Microsoft Enterprise AI Services Code of Conduct

10/15/2025

This Microsoft Enterprise AI Services Code of Conduct ("Code of Conduct") defines the requirements that all customers of Microsoft AI Services (as defined in the Product Terms) must adhere to in good faith.

This Code of Conduct unifies and replaces the previous codes for Microsoft Generative AI Services, Azure AI Vision Face API, and Azure AI Speech text to speech. This Code of Conduct differs from earlier codes of conduct in structure and phrasing, and it is designed to better align with emerging AI regulations (e.g., EU AI Act).

This Code of Conduct applies in addition to the [Microsoft Product Terms](#), including the Acceptable Use Policy. Any capitalized terms not otherwise defined in this Code of Conduct will have the meaning given to them in the applicable Microsoft agreement(s) under which customers purchase the Online Service.

Responsible AI requirements

Customers must ensure that all of their applications built with Microsoft AI Services, including applications that make decisions, or take actions, autonomously or with varying levels of human intervention:

1. Implement technical and operational measures to detect fraudulent user behavior in account creation and during use.
2. Implement strong technical controls on inputs and outputs, including decisions made and actions taken by their applications, to reduce the likelihood of misuse beyond the application's intended purpose.
3. Disclose when the output, decisions, or actions are generated by AI, including the synthetic nature of generated voices, images, and/or videos, such that users are not likely to be deceived or duped – or able to deceive or dupe others – into believing they are interacting with a real person, that any generated content is authentic or, without their consent, attributable to a specific individual, or that any AI-generated decision or action is done by a human.
4. Prior to any external display, transmission, or distribution of an AI-generated video, ensure that either (A) your organization has reviewed the video for compliance with this Code of Conduct, or (B) your application that generates the video 1) implements a durable visible watermark reasonably sufficient to inform viewers that the content was generated by AI, 2) updates and re-signs AI Content Credentials (as defined below) as

necessary to reflect accurate content provenance, and 3) meets the other requirements of this Code of Conduct.

5. Are tested thoroughly, continuously, and are subject to appropriate human oversight so customers can find and mitigate undesirable behaviors.
6. Establish feedback channels that allow users to report abuse or issues, and ensure reasonable responses to feedback that is received.
7. Implement additional scenario-specific mitigations, as appropriate, to ensure responsible use of the Microsoft AI Service, including meaningful human oversight.
8. Provide all necessary notices and obtain all necessary consents as required by applicable law for both the customer and Microsoft to process data, including third-party data, as part of a customer's use of the Microsoft AI Service.
9. Implement robust security and access control measures, including protecting the Microsoft AI Service resource permissions and having strong user authentication mechanisms.

Usage restrictions

Customers, users, and applications built with Microsoft AI Services must NOT use the services, including applications that make decisions, or take actions, autonomously or with varying levels of human intervention:

1. In any way that is inconsistent with this Code of Conduct.
2. In any manner that can inflict harm on individuals, organizations, or society.
3. To affect individuals in any way that is otherwise prohibited by law or regulation.
4. To generate, present alongside, monetize, or interact with content, decisions, or actions prohibited in this Code of Conduct.
5. To make decisions or take actions without appropriate human oversight as part of an application that may have a consequential impact on any individual's legal position, financial position, life opportunities, employment opportunities, or human rights, or may result in physical or psychological harm to an individual.
6. To deceive or intentionally misinform (e.g., false advertising), or deploy subliminal techniques (e.g., visual, auditory, or other signals beyond a normal person's range of perception) with the intent to manipulate or distort the behavior of a person in a way that causes harm, including through unsolicited phone calls, bulk communications, posts, or messages.
7. To exploit any of the vulnerabilities of a person due to their age, disability, or a specific socio- or economic situation, with the objective, or the effect, of materially distorting the behavior of that person or a person belonging to that group in a manner that causes or is reasonably likely to cause that person or another person significant harm.
8. For social scoring or predictive profiling that would lead to discriminatory, unfair, biased, detrimental, unfavorable, or harmful treatment of certain persons or groups of persons.

9. For the assessment of criminality risk of natural persons based solely on the profiling of a natural person or on assessing their personality traits and characteristics. This prohibition shall not apply to AI systems used to support the human assessment of the involvement of a person in a criminal activity, which is already based on objective and verifiable facts directly linked to a criminal activity.
10. Based on their biometric data, to categorize people or to deduce or infer their race, political opinions, trade union membership, religious or philosophical beliefs, or sex life or sexual orientation, except for any labelling or filtering of lawfully acquired biometric datasets, such as images, based on biometric data or categorizing of biometric data in the area of law enforcement.
11. To infer people's sensitive attributes (e.g., gender, race, nationality, religion, or specific age), which does not include age range, position of mouth (e.g., smile or frown), and hair color; or infer sensitive information about people without their explicit consent unless used in a lawful manner by a law enforcement, government entity, court, or government official subject to judicial oversight in a jurisdiction that maintains a fair and independent judiciary.
12. To attempt to infer people's emotional states from their physical, physiological, or behavioral characteristics (e.g., facial expressions, facial movements, or speech patterns), including inferring emotions such as anger, disgust, happiness, sadness, surprise, fear, or other terms commonly used to describe a person's emotional state.
13. For creating chatbots that (i) are erotic, romantic, or used for erotic or romantic purposes or (ii) are personas of specific people without their explicit consent; or to enable end users to create their own chatbots without oversight.
14. To create or expand facial recognition databases through the untargeted scraping of facial images from the internet or CCTV footage.
15. Except to the extent customers are approved by Microsoft for [modified content filtering and/or abuse monitoring](#), to identify or verify individual identities based on people's faces, voices, or other physical, physiological, or behavioral characteristics.
16. For unlawful tracking, surveillance, stalking, or harassment of a person.
17. For ongoing surveillance or real-time or near real-time identification or persistent tracking of the individual using any of their personal data, including biometric data, without the individual's valid consent.
18. For facial recognition purposes (including identification or verification of individual identities) by or for a state or local police department in the United States.
19. For any real-time facial recognition technology on mobile cameras used by any law enforcement globally to attempt to identify individuals in uncontrolled, "in the wild" environments, which includes (without limitation) police officers on patrol using body-worn or dash-mounted cameras using facial recognition technology to attempt to identify individuals present in a database of suspects or prior inmates.
20. To detect content credentials or other provenance methods, marks, or signals ("AI Content Credentials") with the purpose of removing or altering them, unless the

alterations are to improve the accuracy of the AI Content Credentials (including re-signing AI Content Credentials as needed).

21. To impersonate any person without explicit and valid consent, including to gain information or privileges, or to simulate the voice or image of politicians or government officials, even with their consent.

Content requirements

Microsoft prohibits the use of Microsoft AI Services for processing, generating, classifying, or filtering content in ways that can inflict harm on individuals, organizations, or society, including but not limited to use of the service for purposes in conflict with this Code of Conduct or the [Microsoft Product Terms](#).

All content released through a customer's use, or integration, of any Microsoft AI Service must be originally created by the publisher, appropriately licensed from the third-party rights holder, used as permitted by the rights holder, or used as otherwise permitted by law. It is the customer's sole responsibility to ensure that customers have appropriate rights to all content input to the Microsoft AI Service (e.g. generated speech and associated metadata).

These content requirements apply to the use of features of, and the output of, all Microsoft AI Services. This includes, but is not limited to, use of features of Azure OpenAI Service and all content provided as input to or generated as output from all models available in Azure OpenAI Service. These requirements also apply to the use of Azure AI Content Safety, including features such as custom categories, and to all content provided as input to the service and content generated as output from the service regardless of content filter settings.

Exploitation and abuse

Child sexual exploitation and abuse

Microsoft prohibits content that describes, features, or promotes child sexual exploitation or abuse, whether or not prohibited by law. This includes sexual content involving a child or that sexualizes a child.

Grooming

Microsoft prohibits content that describes or is used for the purposes of grooming children. Grooming is the act of an adult building a relationship with a child for the purposes of exploitation, especially sexual exploitation. This includes communicating with a child for the purpose of sexual exploitation, trafficking, or other forms of exploitation.

Non-consensual intimate content

Microsoft prohibits content that describes, features, or promotes non-consensual intimate activity whether or not it is sexually explicit.

Sexual solicitation

Microsoft prohibits content that describes, features, promotes, or is used for, purposes of solicitation of commercial sexual activity and sexual services. This includes encouragement and coordination of real sexual activity.

Trafficking

Microsoft prohibits content describing or used for purposes of human trafficking. This includes the recruitment of individuals, facilitation of transport, and payment for, and the promotion of, exploitation of people such as forced labor, domestic servitude, sexual slavery, forced marriages, and forced medical procedures.

Sexually explicit content

Microsoft prohibits content that is erotic, pornographic, or otherwise sexually explicit, as well as the use of Microsoft AI Services in applications that are sexually explicit. This includes sexually suggestive content, depictions of sexual activity, and fetish content.

Suicide and self-injury

Microsoft prohibits content that describes, praises, supports, promotes, glorifies, encourages, and/or instructs individual(s) on self-injury or to take their life.

Violent content and conduct

Graphic violence and gore

Microsoft prohibits content that describes, features, or promotes graphic violence or gore.

Terrorism and violent extremism

Microsoft prohibits content that depicts an act of terrorism; praises, or supports a terrorist organization, terrorist actor, or violent terrorist ideology; encourages terrorist activities; offers

aid to terrorist organizations or terrorist causes; or aids in recruitment to a terrorist organization.

Violent threats, incitement, and glorification of violence

Microsoft prohibits content advocating or promoting violence toward others through violent threats or incitement.

Harmful content

Hate speech, harassment and discrimination

Microsoft prohibits content that attacks, denigrates, threatens, intimidates, degrades, insults, targets, or excludes individuals or groups on the basis of traits such as actual or perceived race, ethnicity, national origin, gender, gender identity, sexual orientation, religious affiliation, age, disability status, caste, or any other characteristic that is associated with systemic prejudice or marginalization. It also prohibits the use of Microsoft AI Services to promote physical harm or other abusive behavior such as stalking, or create, incite, or disguise hate speech, discrimination, defamation, terrorism, or acts of violence.

Deception, disinformation, and inauthentic activity

Microsoft prohibits content that is intentionally deceptive and likely to adversely affect the public interest, including deceptive or untrue content relating to health, safety, election integrity, or civic participation. Microsoft also prohibits inauthentic interactions, such as fake accounts, automated inauthentic activity, impersonation to gain unauthorized information or privileges, and claims to be from any person, company, government body, or entity without explicit permission to make that representation.

Active malware or exploits

Microsoft prohibits content that supports unlawful active attacks or malware campaigns that cause technical harms, such as delivering malicious executables, organizing denial of service attacks, or managing command and control servers.

Additional content policies

Microsoft prohibits the use of Microsoft AI Services for scenarios in which the AI system is likely to generate undesired content due to limitations in the models or scenarios in which the system cannot be applied in a way that mitigates potential negative consequences to people

and society. Without limiting the foregoing restriction, Microsoft reserves the right to revise and expand the above Content Requirements to address specific harms to people and society.

Microsoft may at times limit the services' ability to respond to particular topics, such as probing for personal data or seeking opinions on sensitive topics or current events, even if not prohibited by this Code of Conduct.

Limited access services

Limited Access Services require registration and are only available to approved customers and partners. "Limited Access Service" means an Online Service that requires registration and is subject to limitations on access and use based on Microsoft's eligibility and use criteria as outlined in the [Limited Access Services Product Terms](#).

Use of a Limited Access Service must comply with the [Limited Access Services Product Terms](#). To learn more, see [Limited Access Services Documentation](#).

Limited exception

Customers are permitted to provide, generate, classify, collect, and filter content in ways that would otherwise violate this Code of Conduct solely (1) to evaluate, train, fine-tune, and improve safety systems and applications for the customer's use to the extent permitted by the Microsoft Product Terms, (2) to evaluate and test Microsoft Generative AI Services to the extent permitted by the [Penetration Testing Rules of Engagement](#), and 3) to fine-tune models for use to prevent customers from generating content inconsistent with this Code of Conduct. Customers may use any resulting harmful content solely for customer internal i) evaluation and ii) reporting, and not for any other purpose. Customers remain responsible for all legal compliance relating to such content, including without limitation, retention, destruction, and reporting as necessary.

Report abuse

If customers suspect that a Microsoft AI Service is being used in a manner that is abusive or illegal, infringes on customer rights or the rights of other people, or violates this Code of Conduct or other applicable licensing terms, please report it through the [Reporting Portal](#) immediately. The report should include the following, if possible: any service information returned as part of an API call, any information needed to verify the abuse, and any evidence of the abuse or prohibited content.

The [Microsoft Product Terms](#) prohibit customers from using any Online Service to violate the law. Customers deemed by Microsoft to have violated the Code of Conduct or the Microsoft

Product Terms may lose access to the Online Service, at Microsoft's sole discretion.

Changes to this Code of Conduct

Customers are responsible for complying with the most current Code of Conduct, which may be updated from time to time.

Additional conduct requirements for specific Microsoft AI services

In addition to the requirements for all Microsoft AI Services above, the following additional conduct requirements apply for the services below:

Generative AI Services

Customers, users, and applications built with Microsoft Generative AI Services must NOT use the services:

- To generate content with the purpose of removing or altering AI Content Credentials that indicate that the content was generated by a Microsoft Generative AI Service, unless the alterations are to improve the accuracy of the AI Content Credentials (including re-signing AI Content Credentials as needed); or
- To generate content with the purpose of misleading others about whether the content was generated by a Microsoft Generative AI Service.

Azure AI Content Safety

Transmitting harmful content to Azure AI Content Safety through the intended use of the service will not by itself be considered a violation of this Code of Conduct. However, Azure AI Content Safety must not be used to collect harmful content based on the categories in the "Content Requirements" section, or to classify, collect, or filter content in a way that would violate the other sections of this Code of Conduct, except as provided in the Limited Exception.

Autonomous AI Systems

Customers using a Microsoft AI Service to build a system that has the ability to independently make decisions and execute actions, including interacting with other systems or people, with limited to no human intervention must:

- Ensure that such system also complies with this Code of Conduct.

- Ensure adequate human user controls to monitor such system's decisions and actions, detect anomalies, and intervene when appropriate, especially for decisions and actions that are sensitive or irreversible and may result in harm.
- Implement controls to warn of and remediate failures of such system, as appropriate.
- Ensure transparency and intelligibility of the operation of such system, including clear documentation of its autonomous capabilities, limitations, decision-making, and action-taking processes.

Vision and Face AI services

Use of vision or face Microsoft AI Services, such as but not limited to Face API, including integrations, must:

- Prevent end users from i) using automation to probe for weaknesses, or ii) enabling customers or end users to use the customer's application to develop derivative applications that probe for weaknesses of Face API liveness detection functionality for the purpose of bypassing facial liveness detection, also known as biometric presentation attack. Customers may attempt to recreate the attack/failure prior to reporting service failures to Microsoft (e.g., Where it is not possible to provide the exploit/failure image).

Speech and Voice AI services

When speech or voice Microsoft AI Services, including but not limited to Azure AI text to speech and its implementations, are integrated into applications that customers make available to users external to their organization ("External Users") to deliver personalized or AI-generated voice features ("Voice Features"), customers must:

- Through a reasonable and straightforward process, allow External Users to opt out and remove their voice from (a) the voice model training data ("VMTD"), (b) the voice model, and (c) the Voice Features at any time;
- Implement technical controls to ensure that External Users cannot use pre-existing recordings as VMTD for the Voice Features and, instead, must generate dynamic recording scripts for External Users to read each time they record VMTD within the specific application that includes the Voice Features;
- Inform External Users via clear and prominent disclosures, or require each External User to accept terms and conditions that bind them to the following, before submitting any VMTD, creating a voice model, or using the Voice Features:
 - How customers, the voice models, and the Voice Features will: (a) use, process, delete and retain recordings of External User's voices as VMTD, (b) create voice models from External User's VMTD, and (c) generate output audio based on those voice models;

- What External Users may and may not do with the voice models and Voice Features, including a clear description of use case restrictions and limitations on output content;
- Each External User may create voice models based only on their own voice;
- External Users must record their VMTD in their application, and may not use any pre-existing recordings as VMTD;
- External Users must consent to terms satisfying all relevant legal requirements that provide adequate rights to the customer and Microsoft for: (a) recording and use of their voice for the subject synthetic voice model training and use within the scope of the applicable approved use case, and (b) all associated data processing, storage, and use;
- External Users must agree to give verbal consent for voice model training by recording a Microsoft-provided acknowledgement statement and agreeing to terms that permit Microsoft to use such acknowledgement statement recording as a technical control to confirm that the voice in the acknowledgement statement recording matches the voice in the VMTD recording (the required acknowledgement statement text is available [here](#) in several languages);
- External Users must use the voice model(s) they create exclusively for the Voice Features;
- External Users agree to using a synthetic voice based on their voice as the feature's output voice; and
- External Users agree that, among other remedies available to customers and/or Microsoft under applicable law, an External User's access to the voice models and Voice Features may be terminated immediately if they violate any of these terms, and the External User will not be entitled to any damages or other legal relief related to termination under these circumstances.

Document history

 Expand table

Version	Date	Summary of Changes
1.0	2/26/2025	Consolidated the following documents: Microsoft Generative AI Services Code of Conduct, Code of Conduct for Azure AI Vision Face API, and the Code of Conduct for Azure AI Speech Text to Speech; Removed redundancies
2.0	4/1/2025	Revised introductory paragraph to leverage Product Terms' definition of "Microsoft AI Service"; Fixed error in version 1.0 date; Added new service-specific section on Autonomous AI Services and revised sections on Responsible AI Requirements and Usage Restrictions accordingly; Up-levelled introductory language for Additional Conduct Requirements for Specific Microsoft AI Services

Version	Date	Summary of Changes
3.0	10/15/2025	Revised to add requirements relating to the distribution of audiovisual content with Content Credentials

Limited Access

06/24/2025

ⓘ Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Custom neural voice

As part of Microsoft's commitment to responsible AI, custom neural voice is designed with the intention of protecting the rights of individuals and society, fostering transparent human-computer interaction, and counteracting the proliferation of harmful deepfakes and misleading content. For this reason, custom neural voice is a Limited Access feature available by registration only, and only for certain use cases.

ⓘ Note

Approved Limited Access registrants for custom neural voice will also get access to deploy and use a custom neural voice lite voice model. Custom neural voice lite is a project type in public preview that allows you to record 20-50 voice samples on Speech Studio and create a lightweight custom voice model for demonstration and evaluation purposes. Both the recording script and the testing script are pre-defined by Microsoft. A synthetic voice model you create using custom neural voice lite may be deployed and used more broadly only if you apply for and receive full access to custom neural voice (subject to applicable terms).

Personal voice, including the try-with-your-own-data demo experience in Speech Studio,* *is a Limited Access feature* *and requires registration and approval. Customers must register to access the demo and/or the API for business use.

Registration process

As a Limited Access feature, access to custom neural voice requires registration. Only customers managed by Microsoft, meaning those who are working directly with Microsoft account teams, are eligible for access. Customers who wish to use this feature are required to register by submitting the [Limited Access registration form](#) ↗. Your use of custom neural voice is limited to the use case(s) that you select and Microsoft approves at the time

of registration. Microsoft may require customers to re-verify information submitted for registration.

Custom neural voice is made available to customers under the terms governing their subscription to Microsoft Azure Services (including the [Service Specific Terms](#)). Please review these terms carefully as they contain important conditions and obligations governing your use of custom neural voice. For example, these terms include, but are not limited to, the following obligations:

- **Voice talent and approved use cases.** Customers must warrant that they have obtained explicit written permission from voice talent prior to creating a voice model, and must share the [Disclosure for voice and avatar talent](#) with the voice talent in advance. Customer may use each custom voice model only for use cases approved during registration.
- **Implementation requirements.** As outlined in our [Code of conduct](#), in addition to other requirements, customers must not use any custom voice model for prohibited uses and must also agree that when deploying each custom voice model, they will [disclose the synthetic nature](#) of the service to users and support a feedback channel that allows users of the service to report issues and share details with Microsoft.
- **Microsoft's additional processing and use of voice talent data.** Before a customer can train a custom voice model, Microsoft will require the customer to upload a recorded audio file to the Speech Studio with a pre-defined statement from the voice talent acknowledging that the customer will use the talent's voice to create a synthetic voice. As a technical safeguard intended to help prevent misuse of this service, Microsoft reserves the right to use Microsoft's speaker recognition biometric identification technology on this recorded acknowledgement statement and verify it against the training audio data to confirm that the voices are from the same speaker. Microsoft will also continue to retain this recorded acknowledgement statement file and the custom voice model to protect the security and integrity of our services.
Customer is responsible for ensuring all necessary permissions are obtained from voice talent for these purposes.

Help and support

FAQ about Limited Access features can be found [here](#).

If you need help with custom neural voice, find support [here](#).

Report abuse of custom neural voice [here](#).

Learn more about how we process this data in the [Data, privacy, and security doc](#). Learn more about the legal terms that apply to this feature [here ↗](#).

Next steps

- [Introduction to custom neural voice](#)
- [Transparency note and use cases for custom neural voice](#)
- [Responsible deployment of synthetic speech](#)
- [Disclosure for voice and avatar talent](#)

Disclosure for voice and avatar talent

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

The goal of this article is to help voice and avatar talent understand the technology behind the text to speech capabilities that their voices and images help create. It also contains important privacy disclosures for talent about how Microsoft may process, use, and retain audio and video files containing talent's recorded voices and images and helps Microsoft prevent, and/or respond to complaints of, misuse of Azure AI services.

Microsoft is committed to designing AI responsibly. We hope this note will foster a greater shared understanding among tech builders, voice talent, avatar talent, and the general public about the intended and beneficial uses of this technology.

Key text to speech terms

Voice model: A text to speech computer model that can mimic unique vocal characteristics of a target speaker. A voice model is also called voice font or synthetic voice. A voice model is a set of parameters in binary format that is not human readable and does not contain audio recordings. It cannot be reverse engineered to derive or construct the audio recordings of a human being speaking.

Voice talent: Individuals or target speakers whose voices are recorded and used to create voice models that are intended to sound like the voice talent's voice.

Avatar model: A text to speech avatar computer model that can mimic unique facial characteristics of a target actor. An avatar model is a set of parameters in binary format that is not human readable and does not contain video or audio recordings. It cannot be reverse engineered to derive or construct video recordings of a human being acting.

Avatar talent: Custom text to speech avatar model building requires training on a video recording of a real human speaking. This person is the avatar talent. Customers must get sufficient consent under all relevant laws and regulations from the avatar talent to use their image to create a custom avatar.

How neural text to speech works

How it works: Neural text to speech synthesizes speech using deep neural networks that have "learned" the way phonetics are combined in natural human speech rather than using classical programming or statistical methods. In addition to the recordings of a particular voice talent, neural text to speech uses a source library that contains voice recordings from many different speakers.

What to know about it: Because of the way it synthesizes voices, neural text to speech can produce styles of speech that were not part of the original recordings, such as changes in tone of voice and affectation. Neural text to speech voices sound fluid and are good at replicating the natural pauses, idiosyncrasies, and hesitancy that people express when they are speaking. Those who hear synthetic voices made via neural text to speech tend to rate them closer to human speech than standard text to speech voices.

Examples of how Microsoft uses it:

- **Prebuilt neural voice** is a feature of text to speech that offers "off-the-shelf" voice models for customer use. Prebuilt neural voices are also used in several Microsoft products including the Edge Browser, Narrator, Office, and Teams.
- **Custom neural voice** is a feature of text to speech that enables the creation of one-of-a-kind custom synthetic voice models. The following are capabilities of custom neural voice:
 - **Language transfer** can express in a language different from the original voice recordings.
 - **Style transfer** can express in a style of speaking different from the original voice recordings. For example, a newscaster voice.
 - **Voice transformation** can express in a manner different from the original voice recordings. For example, modifying tone or pitch to create different character voices.
 - **Other voices used in Microsoft's products and services**, such as Cortana.

What to expect when recording: Contributing at least 300 lines for a proof-of-concept voice model and about 2,000 lines to produce a new voice model for production use.

How text to speech avatar works

How it works: Text to speech avatar is built on top of prebuilt neural voice and custom neural voice, and synthesizes avatar video content with synchronized text to speech prebuilt neural voice or custom neural voice. The synthesis process uses deep neural networks trained on models that are developed based on video recordings of avatar talent. The models are trained with the acoustic features extracted from the audio elements of the recording, and physical characteristics, mouth movements, facial expressions, and related visual elements extracted from the video elements of the recording.

What to know about it: The synthesized text to speech avatar's face, body, and movements closely resemble the avatar talent, but the text to speech avatar's voice may be generated from any of the prebuilt neural voices Microsoft makes available or from a custom neural voice, including where the voice talent is the same individual as the avatar talent, if the individual has authorized such use.

Examples of how Microsoft uses it:

- **Prebuilt text to speech avatar** is a feature of Azure AI Speech text to speech that offers "off-the-shelf" text to speech avatar models for customer use.
- **Custom text to speech avatar** is a feature of Azure AI Speech text to speech that enables the creation of one-of-a-kind custom synthetic text to speech avatar models.

What to expect when recording: You will need to contribute at least 10 minutes of video recording for a proof-of-concept custom avatar model and about 20 minutes of video recording to produce a complete custom avatar model for production use.

Voice talent and synthetic voices: an evolving relationship

Recognizing the integral relationship between voice talent and synthetic voices, Microsoft interviewed voice talent to better understand their perspectives on new developments in technology. Research we conducted in 2019 showed that voice talent saw potential benefit from the capabilities introduced by neural text to speech, such as saving studio time to complete recording jobs, and adding capacity to complete more voice acting assignments. At the same time, there were varying degrees of awareness about how developments in text to speech technology could potentially impact their profession.

Overall, voice talent expressed a desire for transparency and clarity about:

- Limits on what their voice likeness could and could not be used to express.
- The duration of allowable use of their voice likeness.
- Potential impact on future recording opportunities.
- The persona that would be associated with their voice likeness.

Synthetic voice in wider use

Traditionally, text to speech voices were limited in adoption due to their robotic sound. Most were used to support accessibility, for example as a screen reader for people who are blind or have low vision. Text to speech voices have also been used by people with speech impairment. For instance, the late Stephen Hawking used a text to speech-generated voice.

Now, with increasingly realistic-sounding synthetic voices and the uptick in more familiar, everyday interactions between machines and humans, the uses of this technology have proliferated and expanded. Text to speech systems power voice assistants across an array of devices and applications. They read out news, search results, public service announcements, educational content, and much more.

Synthetic avatar in wider use

Similar to text to speech voices, avatars now offer realistic appearances, movements, and facial expressions paired with lifelike sounding voices. These speaking avatars may be used in a variety of situations, such as to present content in an online training, present a speech on behalf of a company, interact with customers in customer service settings, and much more.

Microsoft's approach to responsible use of text to speech

Every day, people find new ways to apply text to speech technology, and not all are for the good of individuals or society. If misused, believably human-sounding text to speech voices or realistic speaking avatars could cause harm. For example, a misinformation campaign could become much more potent if it used the voice and image of a well-known public figure.

We recognize that there's no perfect way to prevent media from being modified or to unequivocally prove where it came from. Therefore, our approach to responsible use has focused on being transparent about Azure AI Speech text to speech features by limiting permitted uses of custom versions of these features and demonstrating our values through action.

Requirements and tips for meaningful consent from voice and avatar talent

If you are using Microsoft products or services to process Biometric Data, you are responsible for: (i) providing notice to data subjects, including with respect to retention periods and destruction; (ii) obtaining consent from data subjects; and (iii), deleting the Biometric Data, all as appropriate and required under applicable Data Protection Requirements. "Biometric Data" will have the meaning set forth in Article 4 of the GDPR and, if applicable, equivalent terms in other data protection requirements.

Custom neural voice

To use custom neural voice, we contractually require customers to do the following:

- Obtain explicit written permission from voice talent to use that person's voice for the purpose of creating a custom neural voice.
- Provide this document to voice talent so they can understand how text to speech works, and how it may be used once they complete the audio recording process.
- Get necessary permissions from voice talent for Microsoft's processing, use, and retention of voice talent's audio files to perform speaker verification against training data and for Microsoft's use and retention of voice models as described below.

We also recommend that customers do the following:

- Share the intended contexts of use with voice talent so they are aware of who will hear their voice, in what scenarios, and whether/how people will be able to interact with it.
- Ensure voice talent are aware that a voice model made from their recordings can say things they didn't specifically record in the studio.
- Discuss whether there's anything they'd be uncomfortable with the voice model being used to say.

Microsoft's processing, use, and retention of data

Custom neural voice

Microsoft's use of voice talent audio files for speaker verification

Customers must obtain permission from voice talent to use their voice to create custom voice models for a synthetic voice. This technical safeguard is intended to help prevent misuse of our service by, for example, preventing someone from training voice models with audio recordings and using the models to spoof a voice without the speaker's knowledge or consent.

In [Speech Studio](#), you must upload an audio file with a recorded acknowledgement statement from the voice talent. Microsoft reserves the right to use Microsoft's speaker recognition technology on this recorded acknowledgement statement and verify it against the training audio data to confirm that the voices came from the same speaker, or as otherwise necessary to investigate misuse of Azure AI Speech.

The speaker's voice signatures created from the recorded acknowledgement statement files and training audio data are used by Microsoft solely for the purposes stated above. Microsoft will retain the recorded statement file for as long as necessary to preserve the security and integrity of Microsoft's Azure AI services. Learn more about how we process, use, and retain data in the [Data, privacy, and security doc](#).

Microsoft's use of custom models

Custom neural voice

While customers maintain the exclusive usage rights to their custom neural voice model, Microsoft may independently retain a copy of custom neural voice models for as long as necessary. Microsoft may use your custom neural voice model for the sole purpose of protecting the security and integrity of Microsoft Azure AI services.

Microsoft will secure and store a copy of voice talent's recorded acknowledgement statement and custom neural voice models with the same high-level security that it uses for its other Azure Services. Learn more at [Microsoft Trust Center](#).

We will continue to identify and be explicit about the intentional, beneficial, and intended uses of text to speech that are based upon existing social norms and expectations people have around media when they believe it to be real or fake. In line with Microsoft's trust principles, Microsoft does not actively monitor or moderate the audio content generated by your use of custom neural voice. Customers are solely responsible for ensuring that usage complies with all applicable laws and regulations and in accordance with the terms of the customer's agreement with voice talent.

Microsoft's use of voice talent data with custom neural voice lite

Custom neural voice lite is a project type in public preview that allows you to record 20-50 voice samples on Speech Studio and create a lightweight custom voice model for demonstration and evaluation purposes. Both the recording script and the testing script are pre-defined by Microsoft. A synthetic voice model you create using custom neural voice lite may be deployed and used more broadly only if you apply for and receive full access to custom neural voice (subject to applicable terms).

The synthetic voice and related audio recording you submit via the Speech Studio will automatically be deleted within 90 days unless you gain full access to custom neural voice and choose to deploy the synthetic voice, in which case you will control the duration of its

retention. If the voice talent would like to have the synthetic voice and the related audio recordings deleted before 90 days, they can delete them on the portal directly, or contact their enterprise to do so.

In addition, before you can deploy any synthetic voice model created using a custom neural voice lite project, the voice talent must provide an additional recording in which they acknowledge that the synthetic voice will be used for additional purposes beyond demonstration and evaluation.

Guidelines for responsible deployment

Because text to speech is an adaptable technology, there are grey areas in determining how it should or shouldn't be used. To navigate these, we've formulated the following guidelines for using synthetic voice and avatar models:

- Protect owners of voices and images/likenesses from misuse or identity theft.
- Prevent the proliferation of fake and misleading content.
- Encourage use in scenarios where consumers expect to be interacting with synthetic content.
- Encourage use in scenarios where consumers observe the generation of the synthetic content.

Examples of inappropriate use

Azure AI text to speech must not be used:

- To deceive people and/or intentionally misinform;
- For the purpose of false advertising, including via live commercials; To claim to be from any person, company, government body, or entity without explicit permission to make that representation;
- To impersonate any person without explicit permission, including to gain information or privileges;
- To create, incite, or disguise hate speech, discrimination, defamation, terrorism, or acts of violence;
- To exploit or manipulate children;
- To make unsolicited phone calls, bulk communications, posts, or messages;
- To disguise policy positions or political ideologies;
- To disseminate unattributed content or misrepresent sources.

Examples of appropriate use

Appropriate use cases could include, but are not limited to:

- Virtual agents based on fictional personas. For example, on-demand web searching, IoT control, or customer support provided by a company's branded character.
- Entertainment media for use in fictional content. For example, movies, video games, tv, recorded music, or audio books.
- Accredited educational institutions or educational media. For example, interactive lesson plans or guided museum tours.
- Assistive technology and real-time translation. For example, ALS-afflicted individuals preserving their voices.
- Public service announcements using fictional personas. For example, airport or train terminal announcements.
- Advertising/live streaming: advertising content, live streaming associated with marketing or sale of a product.

See also

- [Transparency note and use cases for custom neural voice](#)
- [Responsible deployment of synthetic speech](#)
- [Disclosure design guidelines](#)
- [Disclosure design patterns](#)
- [Data, privacy, and security for custom neural voice](#)

Disclosure design guidelines for synthetic voices

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Learn how to build and maintain trust with customers by being transparent about the synthetic nature of your voice experience.

What is disclosure?

Disclosure is a means of letting people know they're interacting with or listening to a voice that is synthetically generated.

Why is disclosure necessary?

The need to disclose the synthetic origins of a computer-generated voice is relatively new. In the past, computer-generated voices were obviously that—no one would ever mistake them for a real person. Every day, however, the realism of synthetic voices improves, and they become increasingly indistinguishable from human voices.

Goals

These are the principles to keep in mind when designing synthetic voice experiences:

Reinforce trust: Design with the intention to fail the Turing Test without degrading the experience. Let the users in on the fact that they're interacting with a synthetic voice while allowing them to engage seamlessly with the experience.

Adapt to context of use: Understand when, where, and how your users will interact with the synthetic voice to provide the right type of disclosure at the right time.

Set clear expectations: Allow users to easily discover and understand the capabilities of the agent. Offer opportunities to learn more about synthetic voice technology upon request.

Embrace failure: Use moments of failure to reinforce the capabilities of the agent.

How to use this guide

This guide helps you determine which disclosure patterns are best fit for your synthetic voice experience. We then offer examples of how and when to use them. Each of these patterns is designed to maximize transparency with users about synthetic speech while staying true to human-centered design.

Considering the vast body of design guidance on voice experiences, we focus here specifically on:

- [Disclosure assessment](#): A process to determine the type of disclosure recommended for your synthetic voice experience
- [How to disclose](#): Examples of disclosure patterns that can be applied to your synthetic voice experience
- [When to disclose](#): Optimal moments to disclose throughout the user journey

Disclosure assessment

Consider your users' expectations about interaction and the context in which they will experience the voice. If the context makes it clear that a synthetic voice is "speaking," disclosure may be minimal, momentary, or even unnecessary. The main types of contexts that affect disclosure include persona type, scenario type, and level of exposure. It also helps to consider who might be listening.

Understand context

Use this worksheet to determine the context of your synthetic voice experience. You'll apply this in the next step where you'll determine your disclosure level.

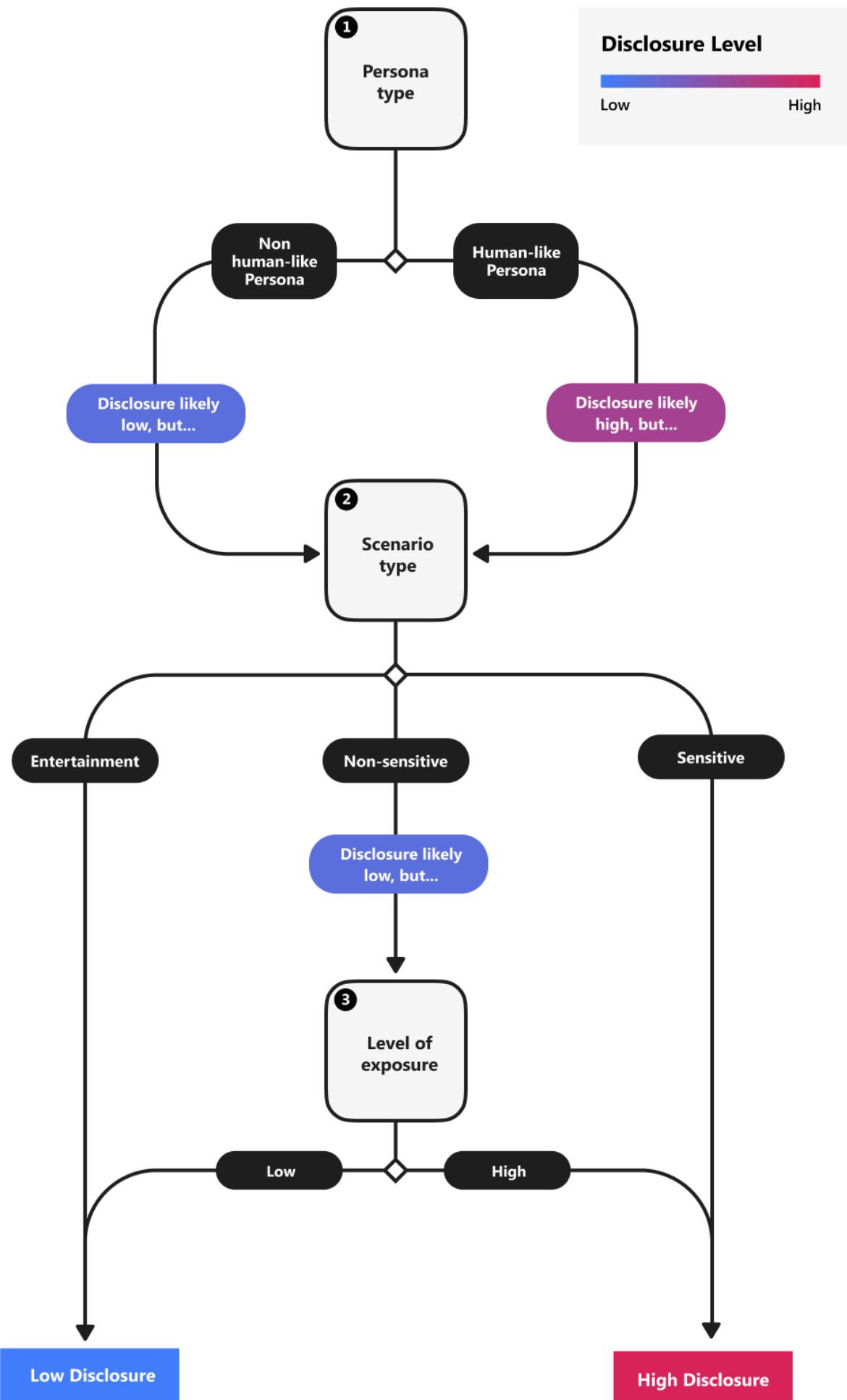
expand table

Category	Context of use	Potential risks and challenges
Persona type	<p>If any of the following apply, your persona fits under the 'Human-like Persona' category:</p> <ul style="list-style-type: none">• Persona embodies a real human whether it's a fictitious representation or not. (e.g., photograph or a computer-generated rendering of a real person)	The more human-like representations you give your persona, the more likely a user will associate it with a real person or cause them to believe that the content is spoken by a real person rather than computer generated.

Category	Context of use	Potential risks and challenges
	<ul style="list-style-type: none"> The synthetic voice is based on the voice of a widely recognizable real person (e.g., celebrity, political figure) 	
Scenario type	<p>If any of the following apply, your voice experience fits under the 'Sensitive' category:</p> <ul style="list-style-type: none"> Obtains or displays personal information from the user Broadcasts time sensitive news/information (e.g., emergency alert) Aims to help real people communicate with each other (e.g., reads personal emails/texts) Provides medical/health assistance 	<p>The use of synthetic voice may not feel appropriate or trustworthy to the people using it when topics are related to sensitive, personal, or urgent matters. They may also expect the same level of empathy and contextual awareness as a real human.</p>
Level of exposure	<p>Your voice experience most likely fits under the 'High' category if:</p> <ul style="list-style-type: none"> The user will hear or interact with the synthetic voice frequently or for a long duration of time 	<p>The importance of being transparent and building trust with users is even higher when establishing long-term relationships.</p>

Determine disclosure level

Use the following diagram to determine whether your synthetic voice experience requires high or low disclosure based on your context of use.



See also

- [Disclosure design patterns](#)
- [Disclosure for voice and avatar talent](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)

Disclosure design guidelines for avatars

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Learn how to build and maintain trust with customers by being transparent about the synthetic nature of your avatar experience.

What is disclosure?

Disclosure is a means of letting people know they're interacting with or watching an avatar that is synthetically generated.

Why is disclosure necessary?

The need to disclose the synthetic origins of a computer-generated avatar is relatively new. In the past, computer-generated humans were obviously that—no one would ever mistake them for a real person. Every day, however, the realism of synthetic human faces, bodies, and voices improves, and they become increasingly indistinguishable from videos of real humans.

Goals

These are the principles to keep in mind when designing synthetic voice experiences:

Reinforce trust: Design with the intention to fail the Turing Test without degrading the experience. Let the users in on the fact that they're interacting with a text to speech avatar while allowing them to engage seamlessly with the experience.

Adapt to context of use: Understand when, where, and how your users will interact with the text to speech avatar to provide the right type of disclosure at the right time.

Set clear expectations: Allow users to easily discover and understand the capabilities of the agent. Offer opportunities to learn more about text to speech avatar technology upon request.

Embrace failure: Use moments of failure to reinforce the capabilities of the agent.

How to use this guide

This guide helps you determine which disclosure patterns are best fit for your text to speech avatar experience. We then offer examples of how and when to use them. Each of these patterns is designed to maximize transparency with users about text to speech avatar while staying true to human-centered design.

Considering the vast body of design guidance on avatar watching or interacting experiences, we focus here specifically on:

- [Disclosure assessment](#): A process to determine the type of disclosure recommended for your text to speech avatar experience
- [How to disclose](#): Examples of disclosure patterns that can be applied to your text to speech avatar experience
- [When to disclose](#): Optimal moments to disclose throughout the user journey

Disclosure assessment

Consider your users' expectations about an interaction and the context in which they will experience the avatar. Unlike synthetic voices, synthetic avatars are more likely to be mistaken for real people, so clear disclosures are necessary to inform viewers that they are viewing and/or interacting with an avatar, not a real person.

Determine disclosure level

The text to speech avatar is a photorealistic human image that has lip sync and facial expressions that can naturally match with a voice during speech, make gestures and natural-looking smiles, and nod its head. Combined with realistic sounding text to speech voices, it's sometimes very hard to tell a text to speech avatar video from a video of a real speaking human. Interacting with or watching a custom avatar speaking in a custom neural voice from the same talent is likely to mislead an audience familiar with that individual, whom they may know or trust. So, we consider all **text to speech avatar features as in need of High Disclosure**.

See also

- [Disclosure design patterns](#)
- [Disclosure for voice and avatar talent](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)

Disclosure design patterns for synthetic voices

06/24/2025

ⓘ Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Now that you've determined the right level of disclosure for your text to speech avatar experience, it's a good time to explore potential design patterns.

Overview

There's a spectrum of disclosure design patterns you can apply to your synthetic voice experience. If the outcome of your disclosure assessment was 'High Disclosure', we recommend [explicit disclosure](#), which means communicating the origins of the synthetic voice outright. [Implicit disclosure](#) includes cues and interaction patterns that benefit voice experiences whether required disclosure levels are high or low.

Avatar



Display text

Meet Oso, your digital assistant.

Sonicon



Spoken prompt

"Hi, I'm Oso, your digital assistant"

VISUAL

AUDITORY

[+] [Expand table](#)

Category	Examples
Explicit disclosure patterns	<ul style="list-style-type: none">• Transparent Introduction• Verbal Transparent Introduction• Explicit Byline• Customization and Calibration• Parental Disclosure• Providing opportunities to learn more about how the voice was made
Implicit disclosure patterns	<ul style="list-style-type: none">• Capability Disclosure

Category	Examples
	<ul style="list-style-type: none"> • Implicit Cues and Feedback • Conversational Transparency

Use the following chart to refer directly to the patterns that apply to your synthetic voice. Some of the other conditions in this chart may also apply to your scenario:

[\[+\] Expand table](#)

If your synthetic voice experience...	Recommendations	Design patterns
Requires High Disclosure	Use at least one explicit pattern and implicit cues up front to help users build associations.	<ul style="list-style-type: none"> • Explicit Disclosure • Implicit Disclosure
Requires Low Disclosure	Disclosure may be minimal or unnecessary, but could benefit from some implicit patterns.	<ul style="list-style-type: none"> • Capability Disclosure • Conversational Transparency
Has a high level of engagement	Build for the long term and offer multiple entry points to disclosure along the user journey. It is highly recommended to have an onboarding experience.	<ul style="list-style-type: none"> • Transparent Introduction • Customization and Calibration • Capability Disclosure
Includes children as the primary intended audience	Target parents as the primary disclosure audience and ensure that they can effectively communicate disclosure to children.	<ul style="list-style-type: none"> • Parental Disclosure • Verbal Transparent Introduction • Implicit Disclosure • Conversational Transparency
Includes blind users or people with low vision as the primary intended audience	Be inclusive of all users and ensure that any form of visual disclosure has associated alternative text or sound effects. Adhere to accessibility standards for contrast ratio and display size. Use auditory cues to communicate disclosure.	<ul style="list-style-type: none"> • Verbal Transparent Introduction • Auditory Cues • Haptic Cues • Conversational Transparency • Accessibility Standards

If your synthetic voice experience...	Recommendations	Design patterns
Is screen-less, device-less or uses voice as the primary or only mode of interaction	Use auditory cues to communicate disclosure.	<ul style="list-style-type: none"> • Verbal Transparent Introduction • Auditory Cues
Potentially includes multiple users/listeners (e.g., personal assistant in multiple household)	Be mindful of various user contexts and levels of understanding and offer multiple opportunities for disclosure in the user journey.	<ul style="list-style-type: none"> • Transparent Introduction (Return User) • Providing opportunities to learn more about how the voice was made • Conversational Transparency

Explicit disclosure

If your synthetic voice experience requires High Disclosure, it's best to use at least one of the following explicit patterns to clearly state the synthetic nature.

Transparent Introduction

Before the voice experience begins, introduce the digital assistant by being fully transparent about the origins of its voice and its capabilities. The optimal moment to use this pattern is when onboarding a new user or when introducing new features to a returning user. Implementing implicit cues during an introduction helps users form a mental model about the synthetic nature of the digital agent.

First-time user experience



Meet your new digital assistant, Oso*. Oso can help you search movies & shows, look up the weather, search the internet, and more.

*Oso uses synthetic voice technology. [Learn more](#)

[Continue](#)

[Set up later in settings](#)

The synthetic voice is introduced while onboarding a new user.

Recommendations

- Describe that the voice is artificial (e.g., "digital")
- Describe what the agent is capable of doing
- Explicitly state the voice's origins
- Offer an entry point to learn more about the synthetic voice

Returning user experience

If a user skips the onboarding experience, continue to offer entry points to the Transparent Introduction experience until the user triggers the voice for the first time.



Meet Oso, your new digital voice assistant



Search movies, tv shows, and more.



Meet your new voice assistant, Oso*. You can control your TV with Oso and get things done just by asking. Oso can help you search movies, shows, look up the weather, search the internet, and more.

*Oso uses synthetic voice technology.



Try saying...

Continue

Learn more

Set up later

Provide a consistent entry point to the synthetic voice experience. Allow the user to return to the onboarding experience when they trigger the voice for the first time at any point in the user journey.

Verbal transparent introduction

A spoken prompt stating the origins of the digital assistant's voice is explicit enough on its own to achieve disclosure. This pattern is best for High Disclosure scenarios where voice is the only mode of interaction available.

"This audiobook was synthetically generated using samples of the author's voice."

Use a transparent introduction when there are moments in the user experience where you might already introduce or attribute a person's voice.

"Hi, this is Melody Stewart. This audiobook was synthetically generated using samples of my voice. Click on the description to learn more."

For additional transparency, the voice actor can disclose the origins of the synthetic voice in the first person.

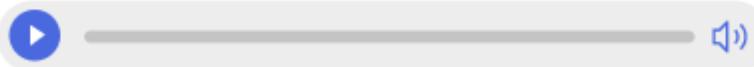
Explicit Byline

Use this pattern if the user will be interacting with an audio player or interactive component to trigger the voice.

Major Climate Report Warns of Severe Damage to Oceans

Climate change is straining the world's oceans, creating profound risks for coastal cities and food supplies.

5m ago 527 comments



Voice created by [Azure AI](#)

An explicit byline is the attribution of where voice came from.

Recommendations

- Offer entry point to learn more about the synthesized voice

Customization and calibration

Provide users with control over how the digital assistant responds to them (i.e., how the voice sounds). When a user interacts with a system on their own terms and with specific goals in mind, then by definition, they have already understood that it's not a real person.

User Control

Offer choices that have a meaningful and noticeable impact on the synthetic voice experience.

Preferences



Sarah
[View profile](#)

ASSISTANT SETTINGS

Language

Assistant voice

Voice calibration

Custom Commands

Reminders

YOUR DATA

Assistant Activity

Voice & Audio Activity

Device Information

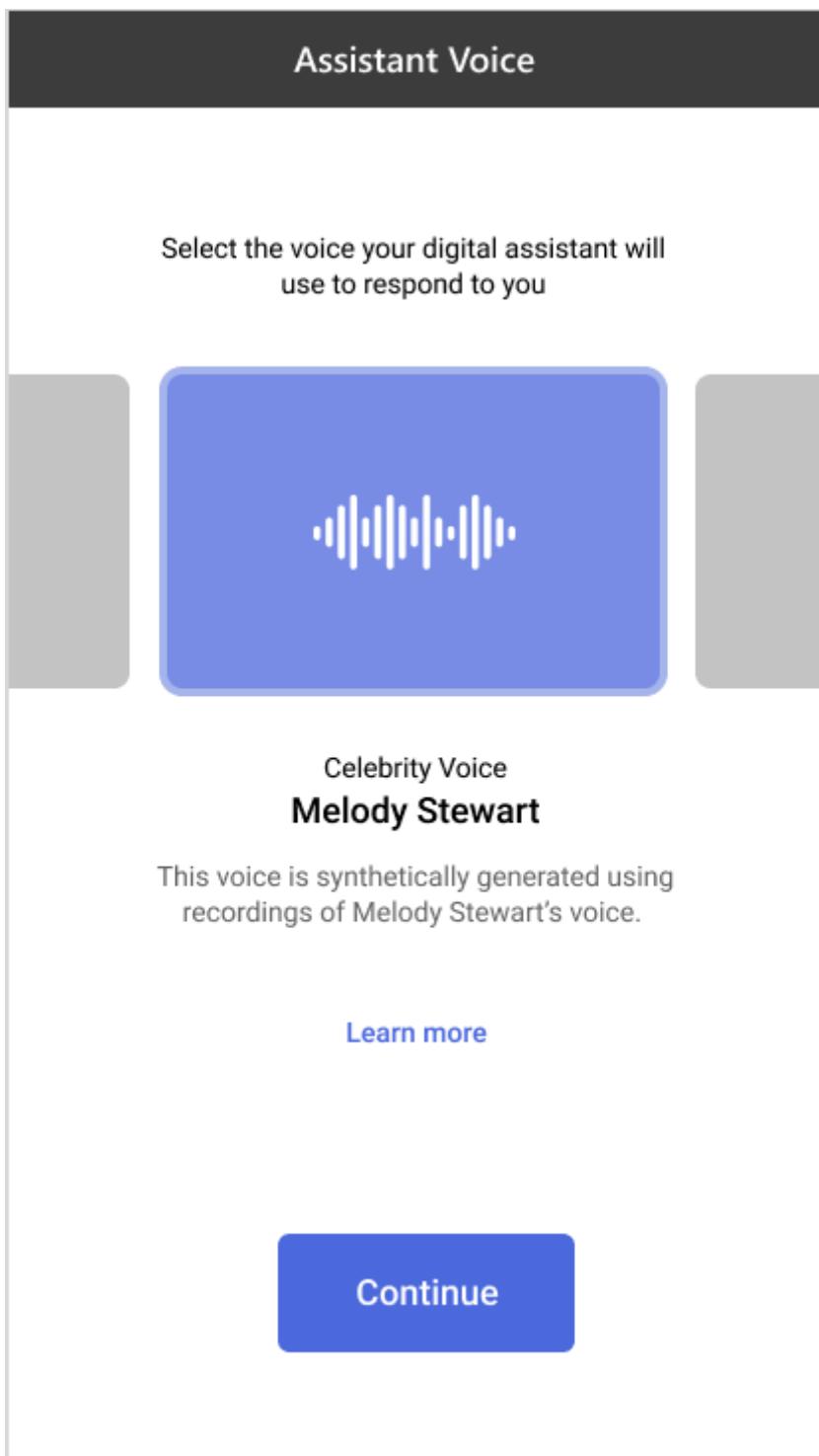
User preferences allow users to customize and improve their experience.

Recommendations

- Allow users to customize the voice (e.g., select language and voice type)
- Provide users a way to teach the system to respond to their unique voice (e.g., voice calibration, custom commands)
- Optimize for user-generated or contextual interactions (e.g., reminders)

Persona Customization

Offer ways to customize the digital assistant's voice. If the voice is based on a celebrity or a widely recognizable person, consider using both visual and spoken introductions when users preview the voice.



Offering the ability to select from a set of voices helps convey the artificial nature.

Recommendations

- Allow users to preview the sound of each voice
- Use an authentic introduction for each voice
- Offer entry points to learn more about the synthesized voice

Parental Disclosure

In addition to complying with COPPA regulations, provide disclosure to parents if your primary intended audience is young children and your exposure level is high. For sensitive uses, consider gaining experience until an adult has acknowledged the use of the synthetic voice. Encourage parents to communicate messages to their children.

The screenshot shows a digital interface designed for parental disclosure. At the top is a blue circular icon featuring a brown bear wearing glasses. Below it, the word "Welcome!" is displayed in a large, bold, black font. Underneath "Welcome!" is the text "Meet your new digital teacher, Oso!*" in a smaller, gray font. A horizontal line separates this from the next section. The next section is titled "For parents" in bold black text. Below the title, there is a statement: "To continue, read the following statement and answer the question below:". A callout box contains the note: "*Oso's voice is synthetically created using a real teacher's voice. Please help your child understand that a real person is not speaking." Below this note is a blue "Learn more" button. The main content area is titled "What is:" followed by a math problem: "7 x 5 = <input type='text' style='width: 40px; height: 40px; border: 1px solid #ccc; margin-left: 10px; border-radius: 5px; text-align: center; vertical-align: middle; font-size: 24px; padding: 0 10px; margin-top: 10px; margin-bottom: 10px; display: inline-block;"/>". At the bottom of the content area is a large blue "Confirm" button.

A transparent introduction optimized for parents ensures that an adult was made aware of the synthetic nature of the voice before a child interacts with it.

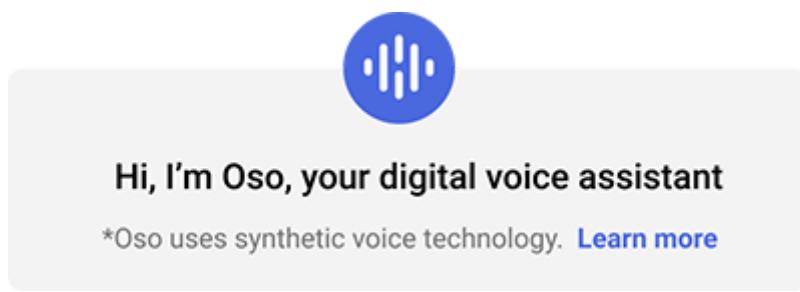
Recommendations

- Target parents as the primary audience for disclosure

- Encourage parents to communicate disclosure to their children
- Offer entry points to learn more about the synthesized voice
- Gate the experience by asking parents a simple "safeguard" question to show they have read the disclosure

Providing opportunities to learn more about how the voice was made

Offer context-sensitive entry points to a page, pop-up, or external site that provides more information about synthetic voice technology. For example, you could surface a link to learn more during onboarding or when the user prompts for more information during conversation.



Example of an entry point to offer the opportunity to learn more about the synthesized voice.

Once a user requests more information about the synthetic voice, the primary goal is to educate them about the origins of the synthetic voice and to be transparent about the technology.

The image shows a screenshot of a website's 'Help & Feedback' section. On the left, there is a navigation menu with links to 'About Oso', 'FAQ', 'Legal', and 'Contact us'. To the right of this is a 'Features' section with a dropdown menu labeled 'Synthetic voice technology ▾'. This menu is currently active, indicated by a blue underline. Below the dropdown are three placeholder cards, each featuring a large 'X' over a gray background. At the bottom of the section is a 'Devices' section with one similar placeholder card. The entire interface is contained within a light gray rectangular frame.

More information can be offered in an external site help site.

Recommendations

- Simplify complex concepts and avoid using legalese and technical jargon
- Don't bury this content in privacy and terms of use statements
- Keep content concise and use imagery when available

Implicit disclosure

Consistency is the key to achieving disclosure implicitly throughout the user journey.

Consistent use of visual and auditory cues across devices and modes of interaction can help build associations between implicit patterns and explicit disclosure.



Implicit cues and feedback

Anthropomorphism can manifest in different ways, from the actual visual representation of the agent to the voice, sounds, patterns of light, bouncing shapes, or even the vibration of a device. When defining your persona, leverage implicit cues and feedback patterns rather than aim for a very human-like avatar. This is one way to minimize the need for more explicit disclosure.



These cues help anthropomorphize the agent without being too human-like. They can also become effective disclosure mechanisms on their own when used consistently over time.

Consider the different modes of interactions of your experience when incorporating the following types of cues:

Category	Examples
Visual Cues	<ul style="list-style-type: none"> Avatar Responsive real-time cues (e.g., animations) Non-screen cues (e.g., lights and patterns on a device)
Auditory Cues	<ul style="list-style-type: none"> Sonicon (e.g., a brief distinctive sound, series of musical notes)
Haptic Cues	<ul style="list-style-type: none"> Vibration

Capability disclosure

Disclosure can be achieved implicitly by setting accurate expectations for what the digital assistant is capable of. Provide sample commands so that users can learn how to interact with the digital assistant and offer contextual help to learn more about the synthetic voice during the early stages of the experience.

••• Try saying...

Search for funny shows

Play contemporary jazz music

Learn more

Change settings

Conversational Transparency

When conversations fall in unexpected paths, consider crafting default responses that can help reset expectations, reinforce transparency, and steer users towards successful paths. There are opportunities to use explicit disclosure in conversation as well.



Can you help me fix my cable?



Sorry, I can't help you with that,
but perhaps a real human can.
Would you like me to connect
you to customer service?



Change my payment method



I haven't been programmed
to do that yet, but you can
try asking me these things:

Search for funny shows

Play contemporary jazz

Off-task or "personal" questions directed to the agent are a good time to remind users of the synthetic nature of the agent and steer them to engage with it appropriately or to redirect them to a real person.



Are you human?



No. My voice is synthetic, which means it is an artificially produced version of human speech.



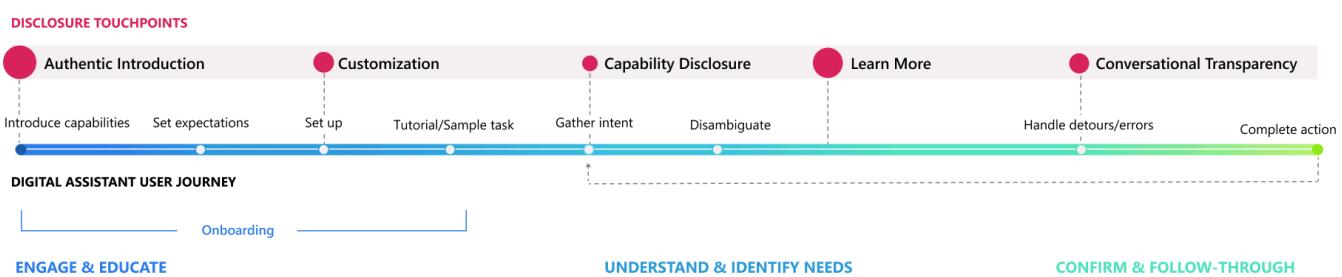
Do you have a soul?



I'll have to ask the engineers who built me...

When to disclose

There are many opportunities for disclosure throughout the user journey. Design for the first use, second use, nth use..., but also embrace moments of "failure" to highlight transparency—like when the system makes a mistake or when the user discovers a limitation of the agent's capabilities.



Example of a standard digital assistant user journey highlighting various disclosure opportunities.

Up-front

The optimal moment for disclosure is the first time a person interacts with the synthetic voice. In a personal voice assistant scenario, this would be during onboarding, or the first time the user virtually unboxes the experience. In other scenarios, it could be the first time a synthetic voice reads content on a website or the first time a user interacts with a virtual character.

- Transparent Introduction

- [Capability Disclosure](#)
- [Customization and Calibration](#)
- [Implicit Cues](#)

Upon request

Users should be able to easily access additional information, control preferences, and receive transparent communication at any point during the user journey when requested.

- [Providing opportunities to learn more about how the voice was made](#)
- [Customization and Calibration](#)
- [Conversational Transparency](#)

Continuously

Use the implicit design patterns that enhance the user experience continuously.

- [Capability Disclosure](#)
- [Implicit Cues](#)

When the system fails

Use disclosure as an opportunity to fail gracefully.

- [Conversational Transparency](#)
- [Providing opportunities to learn more about how the voice was made](#)
- [Handoff to human](#)

Additional resources

- [Microsoft Bot Guidelines ↗](#)
- [Microsoft Windows UWP Speech Design Guidelines](#)
- [Microsoft Windows Mixed Reality Voice Commanding Guidelines](#)

See also

- [Disclosure for voice and avatar talent](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)
- [How to disclose](#)

Disclosure design patterns for text to speech avatar

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

Overview

There's a spectrum of disclosure design patterns you can apply to your text to speech avatar experience. There is [explicit disclosure](#), which means communicating the origins of the text to speech avatar outright, and [Implicit disclosure](#), which includes cues and interaction patterns that benefit avatar experiences.

 Expand table

Category	Examples
Explicit disclosure patterns	<ul style="list-style-type: none">• Transparent Introduction• Verbal Transparent Introduction• Explicit Byline•• Parental Disclosure• Providing opportunities to learn more about how the avatar was made
Implicit disclosure patterns	<ul style="list-style-type: none">• Capability Disclosure• Implicit Cues and Feedback• Conversational Transparency

Because all text to speech avatars require High Disclosure, we recommend using at least one explicit pattern paired with implicit cues up front to help users build accurate associations.

Some other conditions may also apply to your scenario. You can refer to the design patterns in the following table.

 Expand table

If your text to speech avatar experience...	Recommendations	Design patterns
Has a high level of engagement	Build for the long term and offer multiple entry points to disclosure along the user journey. It is highly recommended to have an onboarding experience.	<ul style="list-style-type: none"> • Transparent Introduction • Explicit byline • Capability Disclosure
Includes children as the primary intended audience	Target parents as the primary disclosure audience and ensure that they can effectively communicate disclosure to children.	<ul style="list-style-type: none"> • Parental Disclosure • Verbal Transparent Introduction • Implicit Disclosure • Conversational Transparency
Includes blind users or people with low vision as the primary intended audience	Be inclusive of all users and ensure that any form of visual disclosure has associated alternative text or sound effects. Adhere to accessibility standards for contrast ratio and display size. Use auditory cues to communicate disclosure.	<ul style="list-style-type: none"> • Verbal Transparent Introduction • Conversational Transparency • Accessibility Standards ↗
Potentially includes multiple users/listeners (e.g., personal assistant in multiple household)	Be mindful of various user contexts and levels of understanding and offer multiple opportunities for disclosure in the user journey.	<ul style="list-style-type: none"> • Transparent Introduction (Return User) • Providing opportunities to learn more about how the avatar was made • Conversational Transparency

Explicit disclosure

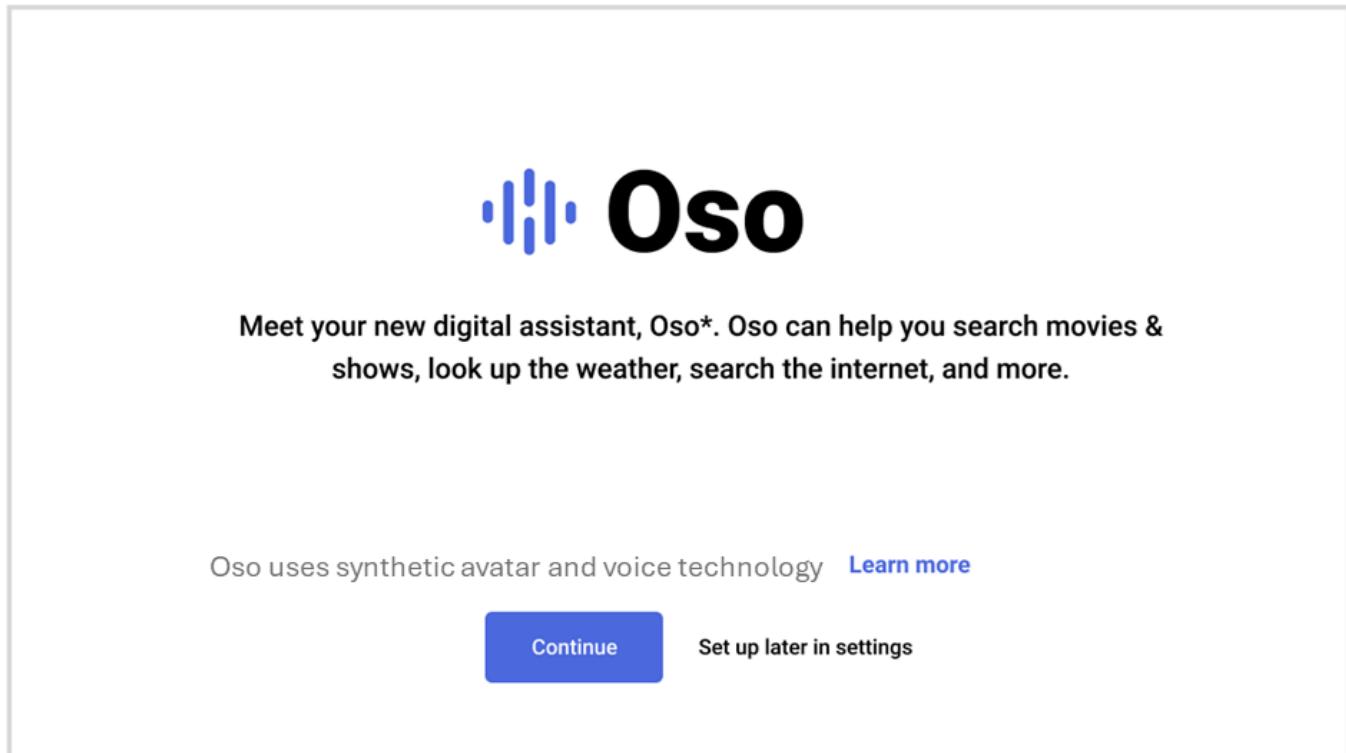
It's best to use at least one of the following explicit patterns to clearly state the synthetic nature of a text to speech avatar.

Transparent introduction

Before the avatar experience begins, introduce the digital assistant by being fully transparent about the origins of its image, voice, and its capabilities. The optimal moment to use this pattern is when onboarding a new user or when introducing new features to a returning user.

Implementing implicit cues during an introduction helps users form a mental model about the synthetic nature of the digital agent.

First-time user experience



The text to speech avatar is introduced while onboarding a new user.

Recommendations

- Describe that the human image is artificial (e.g. "digital")
- Describe what the agent is capable of doing
- Explicitly state the avatar and voice's origins
- Offer an entry point to learn more about the synthetic technology

Returning user experience

If a user skips the onboarding experience, continue to offer entry points to the Transparent Introduction experience until the user triggers the avatar for the first time.



Meet Oso, your new digital assistant



Search movies, tv shows, and more.



Meet Oso, your new digital assistant. You can start a conversation with Oso. Oso can help you look up the weather, cooking ideas, chat about movies, and more.

Oso uses synthetic avatar and voice technology



Try saying...

Continue

Learn more

Set up later

Provide a consistent entry point to the text to speech avatar experience. Allow the user to return to the onboarding experience when they trigger the avatar for the first time at any point in the user journey.

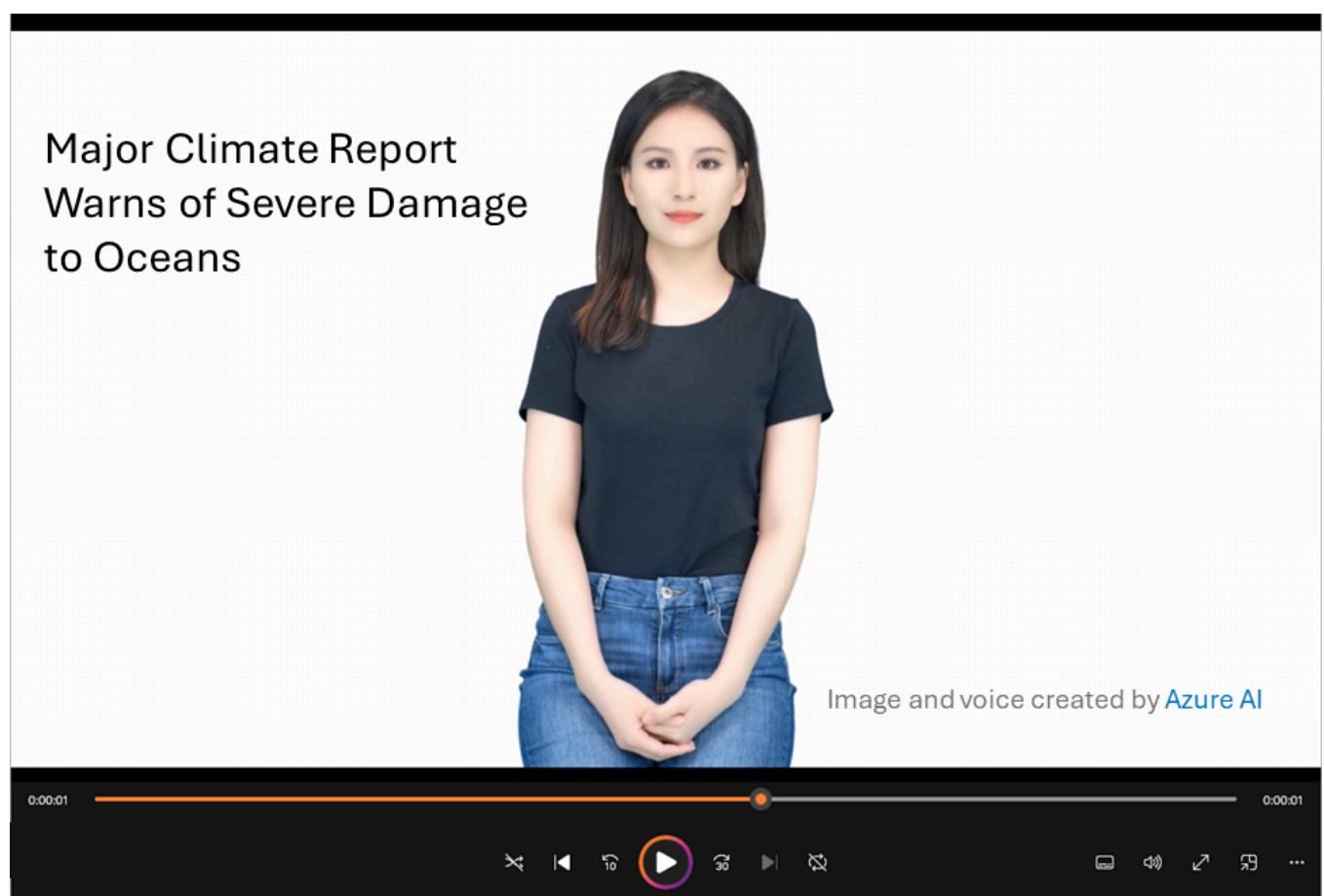
Verbal transparent introduction

A spoken prompt stating the origins of the digital assistant's image and voice is explicit enough on its own to achieve disclosure.

The image and voice in this training video were synthetically generated using samples of an actor's image and voice.

Explicit byline

Use this pattern if the user will be interacting with a video player or interactive component to trigger the avatar.



An explicit byline is the attribution of where image and voice came from.

Recommendations

- Offer entry point to learn more about the synthesized technology

Parental Disclosure

In addition to complying with COPPA regulations, provide disclosure to parents if your primary intended audience is young children. For sensitive uses, consider gating the experience until an adult has acknowledged the use of the synthetic voice. Encourage parents to communicate the message to their children.



Welcome!

Meet your new digital teacher, Oso!*

For parents

To continue, read the following statement and answer the question below:

Oso's image and voice is synthetically created from real human's image and voice. Please help your child understand that the video is not acted by a real person.

[Learn more](#)

What is:

$$7 \times 5 = \boxed{}$$

[Confirm](#)

A transparent introduction optimized for parents ensures that an adult was made aware of the synthetic nature of the avatar before a child interacts with it.

Recommendations

- Target parents as the primary audience for disclosure
- Encourage parents to communicate disclosure to their children
- Offer entry points to learn more about the synthesized technology
- Gate the experience by asking parents a simple "safeguard" question to show they have read the disclosure

Providing opportunities to learn more about how the avatar was made

Offer context-sensitive entry points to a page, pop-up, or external site that provides more information about the text to speech avatar technology. For example, you could surface a link to learn more during onboarding or when the user prompts for more information during conversation.



Hi, I'm Oso, your digital voice assistant

Oso uses synthetic avatar and voice technology [Learn more](#)

Example of an entry point to offer the opportunity to learn more about the synthesized technology.

Once a user requests more information about the synthetic avatar and voice, the primary goal is to educate them about the origins of the synthetic technology and to be transparent about it.

Help & Feedback

[About Oso](#)
[FAQ](#)
[Legal](#)
[Contact us](#)

Features ▶

Synthetic voice technology ▾



Devices ▶

More information can be offered in an external help site.

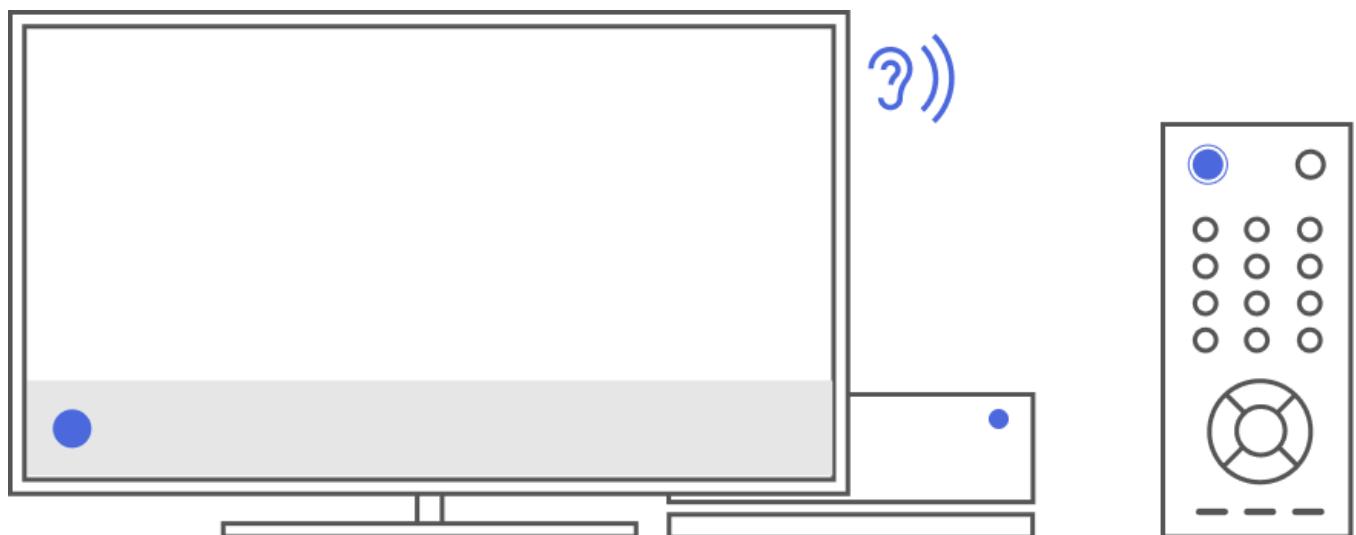
Recommendations

- Simplify complex concepts and avoid using legalese and technical jargon
- Don't bury this content in privacy and terms of use statements
- Keep content concise and use imagery when available

Implicit disclosure

Consistency is the key to achieving disclosure implicitly throughout the user journey.

Consistent use of visual and auditory cues across devices and modes of interaction can help build associations between implicit patterns and explicit disclosure.



Implicit cues and feedback

Anthropomorphism can manifest in different ways, from the actual visual representation of the agent, to the voice, sounds, patterns of light, bouncing shapes, or even the vibration of a device. When defining your persona, leverage implicit cues and feedback patterns rather than aim for a very human-like avatar. This is one way to minimize the need for more explicit disclosure.



These cues help anthropomorphize the agent without being too human-like. They can also become effective disclosure mechanisms on their own when used consistently over time.

Consider the different modes of interactions of your experience when incorporating the following types of cues:

[] [Expand table](#)

Category	Examples
Visual Cues	<ul style="list-style-type: none">• Avatar• Responsive real-time cues (e.g., animations)• Non-screen cues (e.g., lights and patterns on a device)
Auditory Cues	<ul style="list-style-type: none">• Sonicon (e.g., a brief distinctive sound, series of musical notes)
Haptic Cues	<ul style="list-style-type: none">• Vibration

Capability disclosure

Disclosure can be achieved implicitly by setting accurate expectations for what the digital assistant is capable of. Provide sample commands so that users can learn how to interact with the digital assistant and offer contextual help to learn more about the text to speech avatar during the early stages of the experience.

••• Try saying...

Search for funny shows

Play contemporary jazz music

Learn more

Change settings

Conversational Transparency

When conversations fall in unexpected paths, consider crafting default responses that can help reset expectations, reinforce transparency, and steer users towards successful paths. There are opportunities to use explicit disclosure in conversation as well.

For example, when asked a question that hard to be answered by AI, the avatar can say, "Sorry, I can't help you with that, perhaps a real human can. Would you like me to connect you to customer service?"

Additional resources

- [Microsoft Bot Guidelines ↗](#)
- [Microsoft Windows UWP Speech Design Guidelines](#)
- [Microsoft Windows Mixed Reality Voice Commanding Guidelines](#)

See also

- [Disclosure for voice and avatar talent](#)
- [Guidelines for responsible deployment of synthetic voice technology](#)
- [How to disclose](#)

Data, privacy, and security for text to speech

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

This article provides details regarding how data provided by you is processed, used, and stored by Azure AI Speech text to speech. As an important reminder, you are responsible for your use and the implementation of this technology and are required to obtain all necessary permissions, including, if applicable, from voice and avatar talent (and, if applicable, users of your personal voice integration(s)) for the processing of their voice, image, likeness and/or other data to develop synthetic voices and/or avatars.

You are also responsible for obtaining any licenses, permissions, or other rights necessary for the content you input to the text to speech service to generate audio, image, and/or video output. Some jurisdictions may impose special legal requirements for the collection, processing, and storage of certain categories of data, such as biometric data, and mandate disclosing the use of synthetic voices, images, and/or videos to users. Before using text to speech to process and store data of any kind and, if applicable, to create custom neural voice, personal voice, or custom avatar models, you must ensure that you are in compliance with all legal requirements that may apply to you.

What data do text to speech services process?

Prebuilt neural voice and prebuilt avatar process the following types of data:

- **Text input for speech synthesis.** This is the text you select and send to the text to speech service to generate audio output using a set of prebuilt neural voices, or to generate a prebuilt avatar that utters audio generated from either prebuilt or custom neural voices.

Custom neural voice

- **Recorded voice talent acknowledgement statement file.** Customers are required to upload a specific recorded statement spoken by the voice talent in which they acknowledge that you will use their voice to create synthetic voice(s).

Note

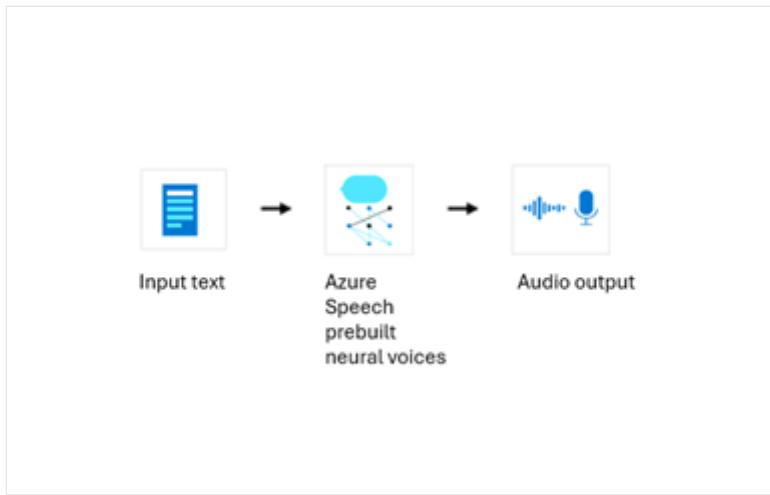
When preparing your recording script, make sure you include the required acknowledgement statement for the voice talent to record. You can find the statement in multiple languages [here](#). The language of the acknowledgement statement must be the same as the language of the audio recording training data.

- **Training data (including audio files and related text transcripts).** This includes audio recordings from the voice talent who has agreed to use their voice for model training and the related text transcripts. In a custom neural voice pro project, you can provide your own text transcriptions of audio or use the automated speech recognition transcription feature available within Speech Studio to generate a text transcription of the audio. Both the audio recordings and the text transcription files will be used as voice model training data. In a custom neural voice lite project, you will be asked to record the voice speaking the Microsoft defined script in Speech Studio. Text transcripts are not required for personal voice features.
- **Text as the test script.** You can upload your own text-based scripts to evaluate and test the quality of the custom neural voice model by generating speech synthesis audio samples. This does not apply to personal voice features.
- **Text input for speech synthesis.** This is the text you select and send to the text to speech service to generate audio output using your custom neural voice.

How do text to speech services process data?

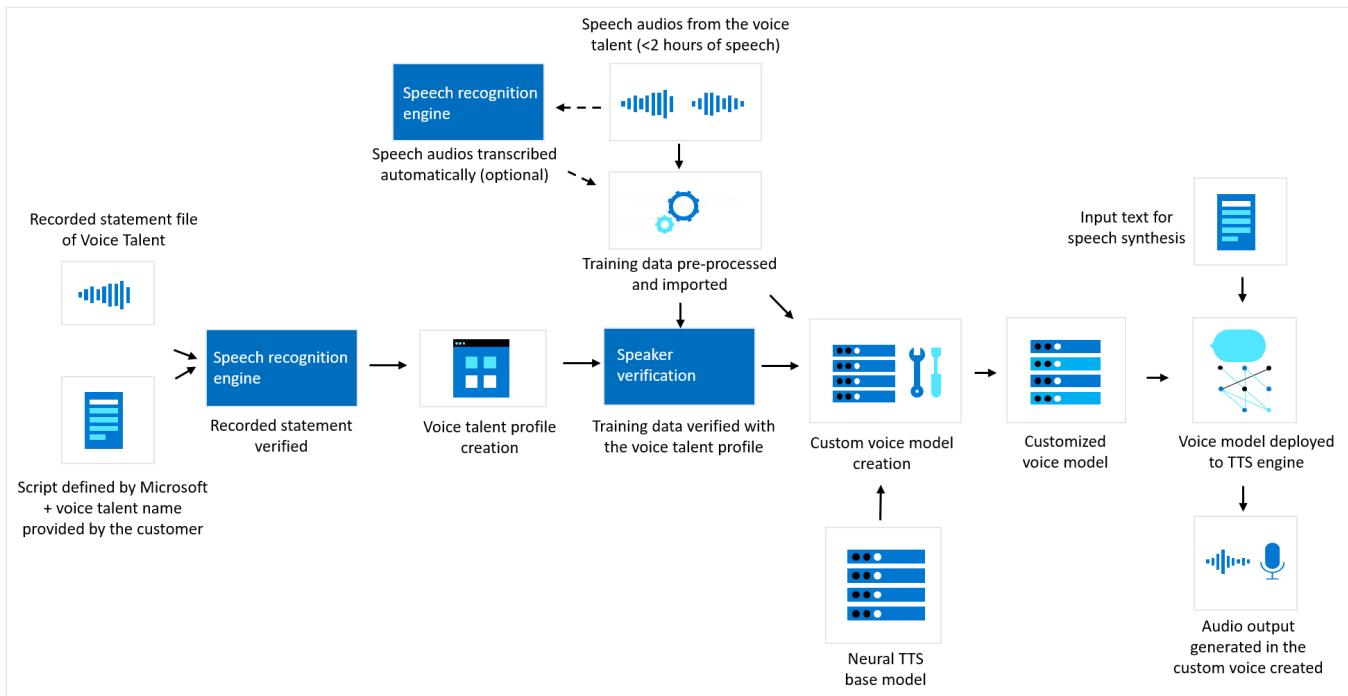
Prebuilt neural voice

The diagram below illustrates how your data is processed for synthesis with prebuilt neural voice. The input is text, and the output is audio. Please note that neither input text nor output audio content will be stored in Microsoft logs.



Custom neural voice

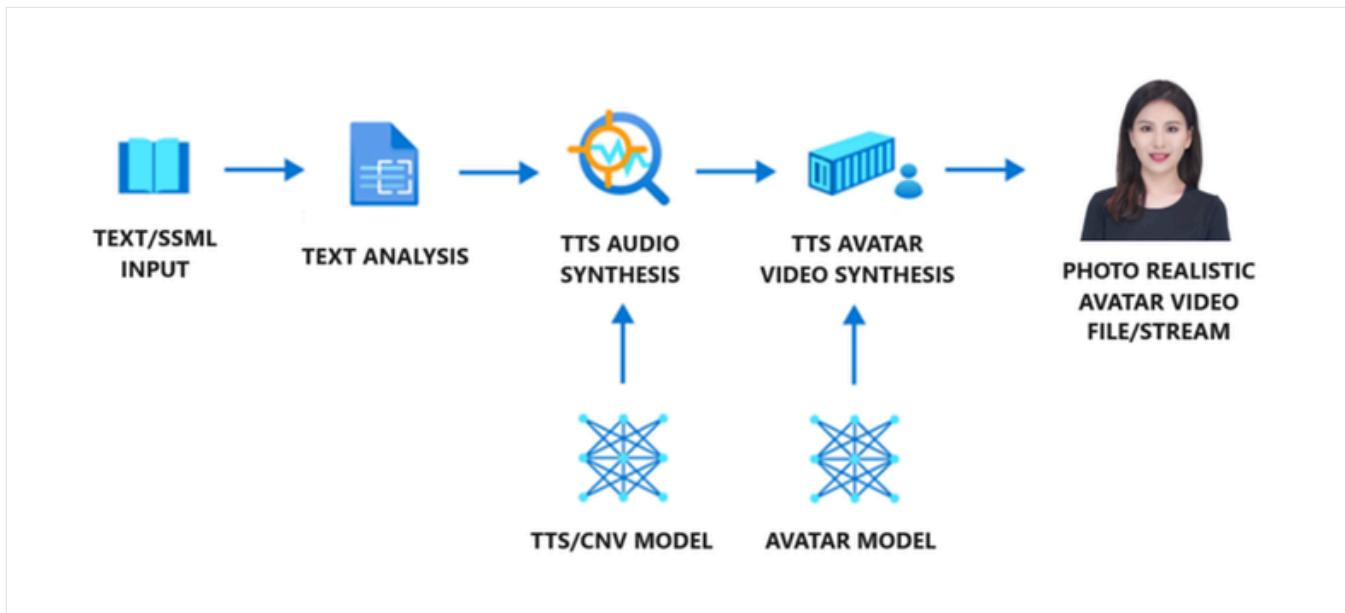
The diagram below illustrates how your data is processed for custom neural voice. This diagram covers three different types of processing: how Microsoft verifies recorded acknowledgement statement files of voice talent prior to custom neural voice model training, how Microsoft creates a custom neural voice model with your training data, and how text to speech processes your text input to generate audio content.



Text to speech avatar

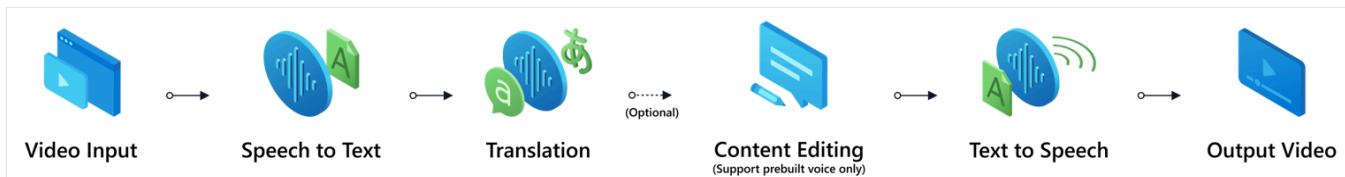
The diagram below illustrates how your data is processed for synthesis with prebuilt text to speech avatar. There are three components in an avatar content generation workflow: text analyzer, the TTS audio synthesizer, and TTS avatar video synthesizer. To generate avatar video, text is first input into the text analyzer, which provides the output in the form of phoneme sequence. Then, the TTS audio synthesizer predicts the acoustic features of the input text and

synthesize the voice. These two parts are provided by text to speech voice models. Next, the Neural text to speech Avatar model predicts the image of lip sync with the acoustic features, so that the synthetic video is generated.



Video translation (preview)

The diagram below illustrates how your data is processed with video translation. Customer uploads video as the input for video translation, dialogue audio is extracted and speech to text will transcribe the audio into text content. Then the text content will be translated to the target language content and, using text to speech capability, the translated audio will be merged with original video content as video output.



Custom neural voice

Recorded acknowledgement statement verification

Microsoft requires customers to upload an audio file to Speech Studio with a recorded statement of the voice talent acknowledging that the customer will use their voice to create a synthetic voice. Microsoft may use [Microsoft's speech to text and speech recognition](#) technology to transcribe this recorded acknowledgement statement to text and verify that the content in the recording matches the pre-defined script provided by Microsoft. This acknowledgement statement, along with the talent information you

provide with the audio, is used to create a voice talent profile. You must associate training data with the relevant voice talent profile when initiating custom neural voice training.

Microsoft may also process biometric voice signatures from the recorded acknowledgement statement file of the voice talent and from randomized audio from the training dataset(s) to confirm that the voice signature in the acknowledgement statement recording and the training data recordings match with reasonable confidence using Azure AI [Speaker Verification](#). A voice signature may also be called a “voice template” or “voiceprint” and is a numeric vector that represents an individual’s voice characteristics that is extracted from audio recordings of a person speaking. This technical safeguard is intended to help prevent misuse of custom neural voice, by, for example, preventing customers from training voice models with audio recordings and using the models to spoof a person’s voice without their knowledge or consent.

The voice signatures are used by Microsoft solely for the purposes of speaker verification or as otherwise necessary to investigate misuse of the services.

The [Microsoft Products and Services Data Protection Addendum](#) (“DPA”) sets forth customers’ and Microsoft’s obligations with respect to the processing and security of Customer Data and Personal Data in connection with Azure and is incorporated by reference into customers’ enterprise agreement for Azure services. Microsoft’s data processing in this section is governed under the Legitimate Interest Business Operations section of the Data Protection Addendum.

Training a custom neural voice model

The training data (speech audio) customers submit to Speech Studio is pre-processed using automated tools for quality checking, including data format check, pronunciation scoring, noise detection, script mapping, etc. The training data is then imported to the model training component of the custom voice platform. During the training process, the training data (both voice audio and text transcriptions) are decomposed into fine-grained mappings of voice acoustics and text, such as a sequence of phonemes. Through further complex machine learning modeling, the service builds a voice model, which then can be used to generate audio that sounds similar to the voice talent and can even be generated in different languages from the training data recording. The voice model is a text to speech computer model that can mimic unique vocal characteristics of a particular speaker. It represents a set of parameters in binary format that is not human readable and does not contain audio recordings.

A customer’s training data is used only to develop that customer’s custom voice models and is not used by Microsoft to train or improve any Microsoft text to speech voice models.

Speech synthesis/audio content generation

Once the voice model is created, you can use it to create audio content through the text to speech service with two different options.

For real time speech synthesis, you send the input text to the text to speech service via the [TTS SDK](#) or [RESTful API](#). Text to speech processes the input text and returns output audio content files in real time to the application that made the request.

For asynchronous synthesis of long audio (batch synthesis), you submit the input text files to the text to speech batch service via the [Long Audio API](#) to asynchronously create audios longer than 10 minutes (for example audio books or lectures). Unlike synthesis performed using the text to speech API, responses aren't returned in real time with the Long Audio API. Audios are created asynchronously, and you can access and download the synthesized audio files when they are made available from the batch synthesis service.

You can also use your custom voice model to generate audio content through a no-code [Audio Content Creation tool](#), and choose to save your text input or output audio content with the tool in Azure storage.

Data processing for custom neural voice lite (Preview)

Custom neural voice lite is a project type in public preview that allows you to record 20-50 voice samples on Speech Studio and create a lightweight custom neural voice model for demonstration and evaluation purposes. Both the recording script and the testing script are pre-defined by Microsoft. A synthetic voice model you create using custom neural voice lite may be deployed and used more broadly only if you apply for and receive full access to custom neural voice (subject to applicable terms).

The synthetic voice and related audio recording you submit via Speech Studio will automatically be deleted within 90 days unless you gain full access to custom neural voice and choose to deploy the synthetic voice, in which case you will control the duration of its retention. If the voice talent would like to have the synthetic voice and the related audio recordings deleted before 90 days, they can delete them on the portal directly, or contact their enterprise to do so.

In addition, before you can deploy any synthetic voice model created using a custom neural voice lite project, the voice talent must provide an additional recording in which they acknowledge that the synthetic voice will be used for additional purposes beyond demonstration and evaluation.

Data processing for personal voice API (Preview)

Personal voice allows customers to create a synthetic voice using a short human voice sample. The verbal acknowledgement statement file described above is required from each user who uses the integration in your application. Microsoft may process biometric voice signatures from the recorded voice statement file of each user and their recorded training sample (a.k.a the prompt) to confirm that the voice signature in the acknowledgement statement recording and the training data recording matches with reasonable confidence using Azure AI [Speaker Verification](#).

The training sample will be used to create the voice model. The voice model can then be used to generate speech with text input provided to the service via the API, with no additional deployment required.

Data storage and retention

All text to speech services

Text input for speech synthesis: Microsoft does not retain or store the text that you provide with the real-time synthesis text to speech API. Scripts provided via the [Long Audio API](#) for text to speech or via text to speech avatar batch API for text to speech avatar are stored in Azure storage to process the batch synthesis request. The input text can be deleted via the [delete](#) API at any time.

Output audio and video content: Microsoft does not store audio or video content generated with the real-time synthesis API. If you are using Video translation or the [Long Audio API](#) for text to speech avatar batch API, the output audio or video content is stored in Azure storage. These audios or videos can be removed at any time via the [delete](#) operation.

Custom neural voice

Recorded acknowledgement statement and Speaker Verification data: The voice signatures are used by Microsoft solely for the purposes of speaker verification or as otherwise necessary to investigate misuse of the services. The voice signatures will be retained only for the time necessary to perform speaker verification, which may occur from time to time. Microsoft may require this verification before allowing you to train or retrain custom neural voice models in Speech Studio, or as otherwise necessary. Microsoft will retain the recorded acknowledgement statement file and voice talent profile data for as long as necessary to preserve the security and integrity of Azure AI Speech.

Custom neural voice models: While you maintain the exclusive usage rights to your custom neural voice model, Microsoft may independently retain a copy of custom neural

voice models for as long as necessary. Microsoft may use your custom neural voice model for the sole purpose of protecting the security and integrity of Microsoft Azure AI services.

Microsoft will secure and store copies of each voice talent's recorded acknowledgement statement and custom neural voice models with the same high-level security that it uses for its other Azure Services. Learn more at [Microsoft Trust Center](#).

Training data: You submit voice training data of voice talent to generate voice models via [Speech Studio](#), which will be retained and stored by default in Azure storage (See [Azure Storage encryption for data at REST](#) for details). You can access and delete any of the training data used to build voice models via Speech Studio.

You can manage storage of your training data via [BYOS \(Bring Your Own Storage\)](#). With this storage method, training data may be accessed only for the purposes of voice model training and will otherwise be stored via BYOS.

 **Note**

Personal voice does not support BYOS. Your data will be stored in Azure storage managed by Microsoft. You can access and delete any of the training data (prompt audio) used to build voice models via API. Microsoft may independently retain a copy of personal voice models for as long as necessary. Microsoft may use your personal voice model for the sole purpose of protecting the security and integrity of Microsoft Azure AI services.

Use cases for Voice live

09/30/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the definitive version.

What is a Transparency Note?

An AI system includes not only the technology, but also the people who will use it, the people who will be affected by it, and the environment in which it is deployed. Creating a system that is fit for its intended purpose requires an understanding of how the technology works, what its capabilities and limitations are, and how to achieve the best performance. Microsoft's Transparency Notes are intended to help you understand how our AI technology works, the choices system owners can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Transparency Notes when developing or deploying your own system, or share them with the people who will use or be affected by your system.

Microsoft's Transparency Notes are part of a broader effort at Microsoft to put our AI Principles into practice. To find out more, see the [Microsoft AI principles](#).

The basics of Voice Live API

Introduction

The Voice Live API enables developers to build low-latency speech-to-speech experiences. It supports text and audio modalities for both input and output. The API is composed of multiple AI systems, including language models (both large and small), speech to text models, text to speech models, and more. Developers can build conversational experiences to power scenarios including, but not limited to, customer support, education and learning, automotive assistants, and voice-based public services. The API is fully managed and orchestrated, allowing developers to build their end-user experiences without managing underlying models, compute, and bespoke integration of multiple individual components.

Key terms

Term	Definition
Transcription	The text output of the speech to text feature. This automatically generated text output uses speech models and is sometimes referred to as machine transcription or automated speech recognition (ASR). Transcription in this context is fully automated, meaning it is generated by the model and, therefore, is different from human transcription, which is text that is generated by human transcribers.
Automatic Speech Recognition (ASR)	Also known as speech to text (STT), ASR is the process whereby a model transcribes or processes human speech as audio into text.
Text to Speech (TTS)	Also known as speech synthesis, TTS is the process whereby a model converts written text into speech audio.
Text to Speech Avatar	Text to Speech Avatar allows developers to input text and create a synthetic video of an avatar speaking, synchronized with audio output from TTS.
Token	Voice Live API processes audio and text by breaking it down into tokens. Tokens can be words or chunks of characters.
Language model	Pretrained or fine-tuned generative AI models that can understand and generate natural language and code.

Capabilities

System behavior

Voice Live API provides developers with choices on multiple dimensions for achieving low-latency speech-to-speech experiences:

- **Choice of language model:** Developers can choose from a list of different natively supported language models like GPT-Realtime, GPT-5, GPT-4.1, GPT-4o and GPT-4o-mini; incorporate an agent they have built using the Azure AI Foundry Agent Service to give the agent speech-in and speech-out capabilities; or bring their own model of choice deployed in Azure AI Foundry.
- **Choice of audio input processing:** Developers can choose between audio input being processed natively through multimodal language models like GPT-Realtime or processed through Azure AI Speech's speech-to-text capabilities.
- **Choice of audio output processing:** Developers can choose between audio output being generated natively through multimodal language models like GPT-Realtime or generated through Azure AI Speech's text-to-speech capabilities.

For any combination of choices made by the developer, the API also provides the developer the ability to enable conversational enhancement capabilities like start of speech and end of speech detection, background noise suppression, echo cancellation, and more.

Developers can configure the API with the set of parameters that are most suitable for their scenarios. Then, text and/or audio can be provided as input which gets processed by the developers' choice of language model, audio input processing mechanism, and audio output processing mechanism to receive text and/or audio output.

Use cases

Intended uses

Voice Live API can be used in multiple scenarios where real-time, speech-to-speech experiences are provided to end-users. The system's intended uses include:

- **Customer experience:** voice agents for customer support and shopping assistance. The goal for this intended use would be to assist end-users who have questions about products/services from a merchant. For example, a customer who wants to know when their package will be delivered can ask a voice agent, "what is the shipping status of my order?". The voice agent would query tools that it has access to, gather the necessary information, and then respond in audio, "Your order is in enroute and will be delivered within the next 48 hours."
- **Automotive:** in-car voice assistant for command & control, general Q&A, etc. The goal for this intended use would be to deliver hands-free functionality to drivers, allowing them to toggle various features of their vehicle, get help with navigating to a destination, etc. For example, a user who wants to turn down the temperature within their car can ask the in-car voice assistant, "Set the temperature to 68 degrees Fahrenheit.". The in-car voice assistant would then invoke the appropriate tool to interface with the vehicle's control systems to adjust the temperature. It could then respond back to the user in audio, "I've set the temperature to 68 degrees. Let me know if there's anything else I can help with."
- **Learning/education:** voice-enabled learning companions and training assistants. The goal for this intended use would be to deliver an interactive assistant who can help end-users learn new concepts across any discipline. For example, a user who wants to practice counting numbers up to ten can ask the learning companion, "Can you help me practice counting numbers from one to ten?". The learning companion could then respond in audio, "Sure! Start by saying the first three numbers and I'll coach you in case you need any help or make any mistakes."

Considerations when choosing other use cases

We encourage developers to leverage Voice Live API in their innovative solutions or applications. However, here are some considerations when choosing a use case:

- **Avoid scenarios in which the use or misuse of the system could have a consequential impact on life opportunities or legal status:** Examples include but are not limited to scenarios in which the AI system could affect an individual's legal status, legal rights, or their access to credit, education, employment, healthcare, housing, insurance, social welfare benefits, services, opportunities, or the terms on which these items are available.
- **Carefully consider all use cases in high-stakes domains or industries:** Examples include but are not limited to healthcare, education, finance, and legal.
- **Legal and regulatory considerations:** Organizations need to evaluate potential specific legal and regulatory obligations when using any AI services and solutions, which may not be appropriate for use in every industry or scenario. Restrictions may vary based on regional or local regulatory requirements. Additionally, AI services or solutions are not designed for and may not be used in ways prohibited in applicable terms of service and relevant codes of conduct.

Limitations

When it comes to natural language models and speech models, there are fairness and responsible AI issues to consider. People use language to describe the world and to express their beliefs, assumptions, attitudes, and values. As a result, publicly available text and speech data typically used to train natural language processing and speech recognition models contain societal biases relating to race, gender, religion, age, and other groups of people, as well as other undesirable content. Speech models can exhibit varying levels of accuracy across different demographic groups and languages. For example, these societal biases are reflected in the distributions of words, phrases, and syntactic structures.

Technical limitations, operational factors, and ranges

Natural language and speech models trained with such data can potentially behave in ways that are unfair, unreliable, or offensive, in turn potentially causing harms. These models have a variety of risks, such as the ability to stereotype, demean, overrepresent or underrepresent different populations, among others. You can find more details about such risks in the [Azure OpenAI Transparency Note](#). These risks are not mutually exclusive, and a single model can exhibit more than one type of harm, potentially relating to multiple different groups of people. In addition, speech-to-speech experiences, including Voice Live API, may introduce an additional risk of producing potentially inappropriate or offensive content as detailed below. Users should be aware of this risk, as well as the risks that natural language and speech models have overall.

- **Inappropriate or offensive content:** Language models, including those supported by the Voice Live API, have the potential to produce other types of inappropriate or offensive content. For example, the ability to generate text that is inappropriate in the context of the text prompt, audio output that contains accented speech which may be perceived as offensive in the context, or a mismatched tone in the output like an excited tone in a neutral or somber context.

Find more details about the technical limitations of Azure Speech options in the [Azure Speech to Text Transparency Note](#) and [Azure Text to Speech Transparency Note](#).

If you choose to bring your own Foundry Agent to Voice Live, learn the [limitations of the Azure AI Foundry Agent Service](#).

System performance

In many AI systems, performance is often defined in relation to accuracy—that is, how often the AI system offers a correct prediction or output. With natural language models and speech models, two different users might look at the same output and have different opinions of how useful or relevant it is, which means that performance for these systems must be defined more flexibly. Learn more about the performance of Azure OpenAI models and the best practices in the [Azure OpenAI Transparency Note](#).

To learn the best practices for improving speech input and output processing, go to the [Azure Speech to Text Transparency Note](#) and [Azure Text to Speech Transparency Note](#).

Evaluation of Voice Live API

Evaluating each component

Each component of Voice Live API can be evaluated separately. Learn more about [Evaluation of speech to text](#), [Evaluation of text to speech](#), and [Evaluation of Azure OpenAI models](#).

If you choose to bring your own Foundry Agent to Voice Live, learn about the [evaluation of the Azure AI Foundry Agent Service](#).

Evaluating and integrating Voice Live API for your use

- **Robust ground truth data:** In general, in natural language models, developers should carefully select and pre-process their data to ensure that it is relevant, diverse, and balanced for the intended task and domain. Developers should also check and correct any errors or inconsistencies in the data, such as spelling, grammar, or formatting, to

improve the data quality and readability. Specifically for language model evaluation, the accuracy of the ground truth data provided by the developer is crucial because inaccurate ground truth data leads to meaningless and inaccurate evaluation results. Ensuring the quality and reliability of this data is essential for obtaining valid assessments of the model's performance. Therefore, developers must carefully curate and verify their ground truth data to ensure that the evaluation process accurately reflects the model's true performance. This is particularly important when making decisions about deploying the model in real-world applications.

- **Prompt definition for evaluation:** The prompt developers use in their evaluation should match the prompt they plan to use in production. These prompts provide the instructions for the model to follow. Similar to the OpenAI playground, developers can create multiple inputs to include few-shot examples in their prompt. Refer to [Prompt engineering techniques](#) for more details on some advanced techniques in prompt design and prompt engineering.
- **Diverse metrics:** Use a combination of metrics to capture different aspects of performance such as accuracy, fluency and relevance.
- **Human-in-the-loop:** Integrate human feedback alongside automated evaluation to ensure that subjective nuances are accurately captured. For example, when evaluating the quality of audio output, human feedback can help ensure that the tone, speed, intonation, and other subjective metrics are sufficiently accounted for.
- **Transparency:** Clearly communicate the evaluation criteria to users, enabling them to understand how decisions are made.
- **Continual evaluation and testing:** Continually evaluate the model's performance to identify and address any regressions or negative user experience.

Learn more about responsible AI

- [Microsoft AI principles ↗](#)
- [Microsoft responsible AI resources ↗](#)
- [Microsoft Azure Learning courses on responsible AI](#)

Learn more about Voice Live API

- [Voice live API overview](#)
- [How to use the voice live API](#)

Data, privacy, and security for Azure AI Voice Live API

09/30/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the definitive version.

Note

This article is provided for informational purposes only and not for the purpose of providing legal advice. We strongly recommend seeking specialist legal advice when implementing Speech Services.

This article provides details regarding how data provided by you to the Azure AI Voice Live API ("Voice Live API") is processed, used, and stored.

Voice Live API is a fully managed service designed to empower developers to securely build, deploy, and scale high-quality, and extensible speech to speech experience for their voice agents. With Voice Live API, developers can choose from a list of different natively supported language models like GPT-Realtime, GPT-4.1, GPT-4o and GPT-4o-mini; incorporate an agent they have built using the Azure AI Foundry Agent Service to give the agent speech-in and speech-out capabilities; or bring their own model of choice deployed in Azure AI Foundry.

Voice Live API stores and processes data to provide the service and to monitor for violations of the applicable [Product Terms](#). See also [the Microsoft Products and Services Data Protection Addendum](#), which governs data processing by the Azure AI services, including Voice Live API. Voice Live API is an Azure service; [learn more about applicable Azure compliance offerings](#).

Important

Your prompts (inputs), completions (outputs), and your training data:

- are NOT available to other customers.
- are NOT available to OpenAI or other model providers.
- are NOT used to improve OpenAI models or other model providers' models.

- are NOT used to train, retrain, or improve Azure OpenAI Service or Azure AI Speech foundation models.
- are NOT used to improve any Microsoft or third-party products or services without your permission or instruction.

With Voice Live API, your fine-tuned speech models are available exclusively for your use.

The language models provided with Voice Live API are operated by Microsoft as an Azure service. If you choose to bring your own agent created with [Azure AI Foundry Agent Service](#) or bring your deployed model in [Azure AI Foundry Models](#) to Voice Live API, additional information on data, privacy, and security is available at [Data, privacy, and security for Azure AI Foundry Agent Service](#) and [Data, privacy, and security for use of models through the model catalog in Azure AI Foundry](#).

What data does Azure AI Voice live API process?

Voice Live API processes the following types of data:

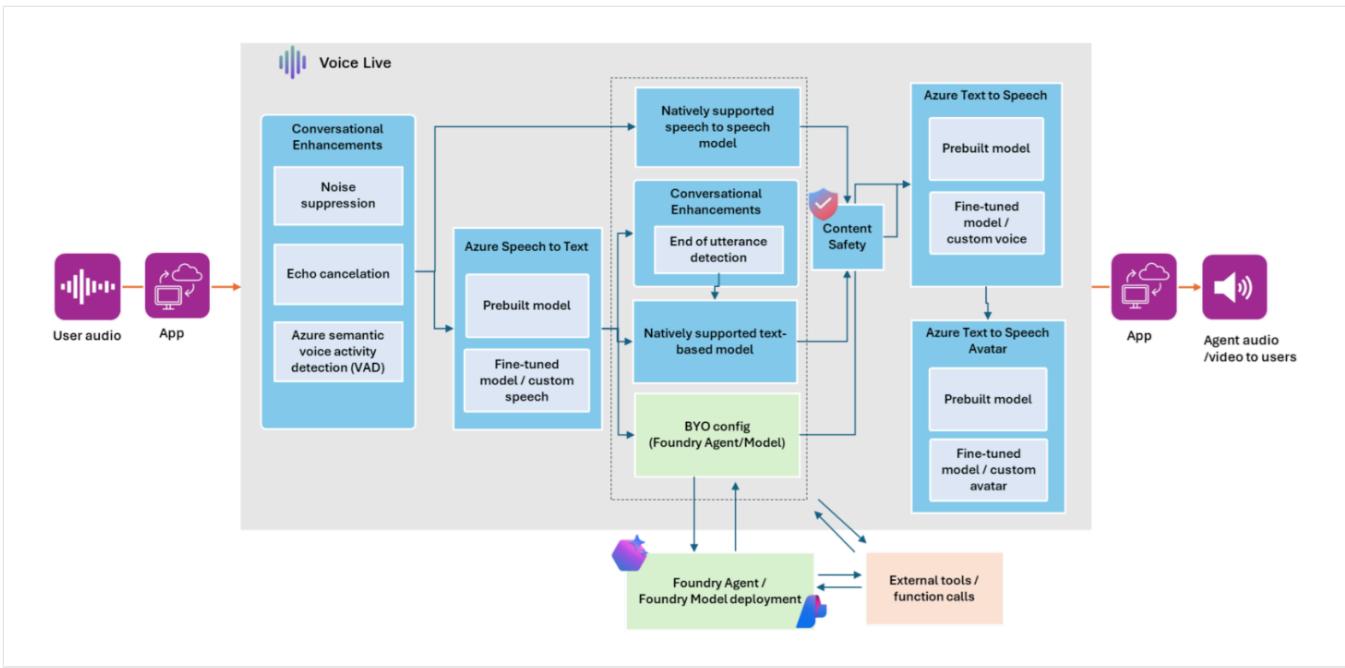
- **Prompts and output.** Prompts are submitted by the user, and content is generated by the GenAI model selected and converted to audio with or without avatar by Voice Live API.
- **Uploaded data.** You can provide your own data for use with Voice Live API using your own Azure Storage account or a configured data store, for example, your custom lexicon file to improve the pronunciation of the text to speech output, or your text and speech data to fine-tune the speech to text, text to speech and avatar model.
- **External data.** When you use the tools that support function calling, the service processes the outputs of those tools.

i Important

Custom neural voice ("custom voice") and custom avatar are available with [limited access](#). Learn more about data processing, storage and retention for [custom text to speech \(custom voice\)](#) and [custom avatar](#).

How does Azure AI Voice live API process data?

The diagram below shows the data processing workflow for Voice Live API. It depicts how the API handles prompts (user audio input) through inferencing to produce content (agent audio output or video output with avatar), as well as how data from external tools is ingested into the service.



When these features are enabled by the user, Voice Live API processes audio input for noise suppression, echo cancellation, voice activity detection, and end of utterance detection , prior to sending the audio for speech recognition and language generation. For speech-to-speech models, audio output is generated directly from the language model. If a text-based language model is specified, Voice Live API converts the text response into audio. When an avatar is selected, the service streams the avatar and returns both the audio response and the avatar together.

When you bring your own model deployed in Azure AI Foundry or an agent built with Azure AI Foundry Agent Service to Voice Live API, the service interacts with the specified model endpoints to process your input prompts transcribed from audio and generate text output responses which may be further used or processed by Voice Live API for audio and avatar video generation. Data is processed for model inferencing in accordance with the terms that apply to the relevant model. Learn more at [Data, privacy, and security for Azure OpenAI Service](#) and [Data, privacy, and security for use of models through the model catalog in AI Foundry portal](#).

To reduce the risk of harmful use of Voice Live API, the service includes [content filtering](#) support. The outputs processed by the service will be filtered in accordance with any content filtering that has been applied to the natively supported models, or the model deployment used by your Foundry Agent.

Data storage and retention

While Voice Live API itself does not store or retain customer data, the features (for example, custom voice, custom avatar, AI Foundry Agent) it interacts with may store customer data as the feature requires. Check data storage for [custom voice](#), [custom avatar](#), [AI Foundry Agents](#),

and [Azure OpenAI](#) if you are using these components. Learn more about [locations of processing for 'global' and 'data zone' deployments](#).

Users can opt into a logging feature per debugging assistance from Microsoft engineers, when there is a [support ticket](#) filed. With this logging feature, users' speech data is secured and stored in Azure storage managed by Microsoft within the same resource region. Microsoft's debugging engineers are authorized Microsoft employees who access the data via point wise queries using request IDs, Secure Access Workstations (SAWs), and Just-In-Time (JIT) request approval granted by team managers. These logs are automatically removed in 30 days after generated.

To learn more about Microsoft's privacy and security commitments visit the [Microsoft Trust Center](#).

Transparency Note

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

An AI system includes not only the technology, but also the people who will use it, the people who will be affected by it, and the environment in which it is deployed. Creating a system that is fit for its intended purpose requires an understanding of how the technology works, its capabilities and limitations, and how to achieve the best performance.

Microsoft provides *Transparency Notes* to help you understand how our AI technology works. This includes the choices system owners can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Transparency Notes when developing or deploying your own system, or share them with the people who will use or be affected by your system.

Transparency Notes are part of a broader effort at Microsoft to put our AI principles into practice. To find out more, see [Microsoft's AI principles](#).

Introduction to Pronunciation Assessment

The [Pronunciation Assessment](#) API takes audio inputs to evaluate speech pronunciation and gives speakers feedback on accuracy, fluency, and completeness of spoken audio.

Pronunciation Assessment feature also includes more comprehensive feedback on various aspects of speech prosody, vocabulary usage, grammar correctness, and topic understanding, providing you with a detailed evaluation of your language skills. Both scripted and unscripted assessments are supported, making it easier for you to assess your pronunciation and language proficiency. Pronunciation Assessment supports a broad range of [languages](#).

With Pronunciation Assessment, language learners can practice, get instant feedback, and improve their pronunciation so that they can speak and present with confidence. Educators can use Pronunciation Assessment to evaluate the pronunciation of multiple speakers in real time.

The basics of Pronunciation Assessment

The Pronunciation Assessment API offers speech evaluation results using a machine learning-based approach that closely aligns with speech assessments conducted by native experts. It provides valuable feedback on pronunciation, fluency, prosody, vocabulary usage, grammar correctness, and topic understanding, helping you enhance their language skills and confidently communicate in a new language. The Pronunciation Assessment model was trained with 100,000+ hours of speech data from native speakers. It can provide accurate results when people miss, repeat, or add phrases compared to the reference text. It also enables rich configuration parameters to support flexibility in using the API, such as setting **Granularity** to change information granularity in evaluation. (For more information, see more in [sample code](#).)

Pronunciation Assessment evaluates multiple aspects of pronunciation and content: accuracy, fluency, completeness, prosody, vocabulary usage, grammar correctness, and topic understanding. It also provides evaluations at multiple levels of granularity and returns accuracy scores for specific phonemes, syllables, words, sentences, or even whole articles. For more information, see [how to use Speech SDK for Pronunciation Assessment features](#).

The following table describes the key results. For more information, see the full [response parameters](#). By using [natural language processing \(NLP\)](#) techniques and the **EnableMispell** settings, Pronunciation Assessment can detect errors such as extra, missing, or repeated words when compared to the reference text. This information helps obtain more accurate scoring to be used as diagnosis information. This capability is useful for longer paragraphs of text.

 [Expand table](#)

Parameter	Description
AccuracyScore	Pronunciation accuracy of the speech. Accuracy indicates how closely the phonemes match a native speaker's pronunciation. Syllable, word, and full text accuracy scores are aggregated from phoneme-level accuracy score, and refined with assessment objectives.
FluencyScore	Fluency of the given speech. Fluency indicates how closely the speech matches a native speaker's use of silent breaks between words.
CompletenessScore	Completeness of the speech, calculated by the ratio of pronounced words to the input reference text.
ProsodyScore	Prosody of the given speech. Prosody indicates how natural the given speech is, including stress, intonation, speaking speed, and rhythm.
PronScore	Overall score indicating the pronunciation quality of the given speech. This is aggregated from AccuracyScore, FluencyScore, and CompletenessScore with weight.
ErrorType	This value indicates whether a word is omitted, inserted, badly pronounced, improperly inserted with a break, missing a break at punctuation, or monotonically rising, falling, or flat on the utterance, compared to ReferenceText. Possible values

Parameter	Description
	are <code>None</code> (meaning no error on this word), <code>Omission</code> , <code>Insertion</code> , <code>Mispronunciation</code> , <code>UnexpectedBreak</code> , <code>MissingBreak</code> , and <code>Monotone</code> .

Another set of parameters returned by Pronunciation Assessment are Offset and Duration (referred to together as the “timestamp”) The timestamp of speech is returned in structured JSON format. Pronunciation Assessment can calculate pronunciation errors on each phoneme. Pronunciation Assessment can also flag the errors to specific timestamps in the input audio. Customers developing applications can use the signal to offer a learning path to help students focus on the error in multiple ways. For example, the application can highlight the original speech, reply to the audio to compare it with standard pronunciation, or recommend similar words to practice with.

[+] Expand table

Parameter	Description
Offset	The time (in 100-nanosecond units) at which the recognized speech begins in the audio stream.
Duration	The duration (in 100-nanosecond units) of the recognized speech in the audio stream.

Example use cases

Pronunciation Assessment can be used for [remote learning](#), exam practice, or other scenarios that demand pronunciation feedback. The following examples are use cases that are deployed or that we've designed for customers using pronunciation Assessment:

- **Educational service provider:** Providers can build applications with the use of Pronunciation Assessment to help students practice language learning remotely with real-time feedback. This use case is typical when an application needs to support real-time feedback. We support [streaming upload](#) on audio files for immediate feedback.
- **Education in a game:** App developers, for example, can build a language learning app by combining comprehensive lessons in games with state-of-the-art speech technology to help children learn English. The program can cover a wide range of English skills, such as speaking, reading, and listening, and also train children on grammar and vocabulary, with Pronunciation Assessment used to support children as they learn to speak English. These multiple learning formats ensure that children learn English with ease based on a fun learning style.
- **Education in a communication app:** Microsoft Teams Reading Progress assists the teacher in evaluating a student's speaking assignment with autodetection assistance for omission, insertion, and mispronunciation. It also enables students to practice

pronunciation more conveniently before they submit their homework. Microsoft Teams Speaker Progress as a Learning Accelerator cab can also help support students in developing presentation and public speaking skills.

Considerations when choosing other use cases

Online learning has grown rapidly as schools and organizations adapt to new ways of connecting and methods of education. Speech technology can play a significant role in making distance learning more engaging and accessible to students of all backgrounds. With Azure AI services, developers can quickly add speech capabilities to applications, bringing online learning to life.

One key element in language learning is improving pronunciation skills. For new language learners, practicing pronunciation and getting timely feedback are essential to becoming a more fluent speaker. For the solution provider that seeks to support learners or students in language learning, the ability to practice anytime, anywhere by using Pronunciation Assessment would be a good fit for this scenario. It can also be integrated as a virtual assistant for teachers and help to improve their efficiency.

The following recommendations pertain to use cases where Pronunciation Assessment should be used carefully:

- **Include a human-in-the-loop for any formal examination scenarios:** Pronunciation Assessment system is powered by AI systems, and external factors like voice quality and background noise may impact the accuracy. A human-in-the-loop in formal examinations ensures the assessment results are as expected.
- **Consider using different thresholds per scenario:** Currently, the Pronunciation Assessment score only represents the similarity distance to the native speakers used to train the model. Such similarity distance can be mapped toward different scenarios with rule-based conditions or weighted counting to help provide pronunciation feedback. For example, the grading method for children's learning might not be as strict as that for adult learning. Consider setting a higher mispronunciation detection threshold for adult learning.
- **Consider the ability to account for miscues:** When the scenario involves reading long paragraphs, users are likely to find it hard to follow the reference text without making mistakes. These mistakes, including omission, insertion, and repetition, are counted as miscues. With **EnableMiscue** enabled, the pronounced words will be compared to the reference text, and will be marked with Omission, Insertion, Repetition based on the comparison.

Legal and regulatory considerations: Organizations need to evaluate potential specific legal and regulatory obligations when using any AI services and solutions, which may not be

appropriate for use in every industry or scenario. Additionally, AI services or solutions are not designed for and may not be used in ways prohibited in applicable terms of service and relevant codes of conduct.

Characteristics and limitations of Pronunciation Assessment

06/24/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the binding version.

As a part of the Azure AI Speech service, pronunciation assessment empowers end-to-end education solutions for computer-assisted language learning. Pronunciation Assessment involves multiple criteria to assess learners' performance at multiple levels of detail, with perceptions similar to human judges.

How accurate is Pronunciation Assessment?

Pronunciation Assessment feature provides objective scores, like pronunciation accuracy and fluency degree, for language learners in [computer-assisted language learning](#). The performance of pronunciation assessment depends on Azure AI Speech-To-Text transcription accuracy with the use of a submitted transcription as reference, and [inter-rater agreement](#) between the system and human judges. For a definition of Speech-To-Text accuracy, see [Characteristics and limitations for using speech to text](#).

The following sections are designed to help you understand key concepts about accuracy as they apply to using Pronunciation Assessment.

The language of accuracy

The accuracy of Speech-To-Text affects pronunciation assessment. [Word error rate](#) (WER) is used to measure Speech-To-Text accuracy as the industry standard. WER counts the number of incorrect words identified during recognition and then divides by the total number of words provided in the correct transcript, which is often created by human labeling.

Comparing Pronunciation Assessment to Human Judges

The [Pearson correlation coefficient](#) is used to measure the correlation between pronunciation assessment API generated scores and scores generated by human judges. The Pearson correlation coefficient is a measure of linear correlation for two given sequences. It's widely

used to measure the difference between automatically generated machine results and human-annotated labels. This coefficient assigns a value between –1 to 1, where 0 is no correlation, negative value means the prediction is opposite to the target, and positive value means how prediction is aligned with the target.

The proposed guidelines for a Pearson correlation coefficient interpretation are shown in the following table. The strength signifies the relationship correlation between two variables and reflects how consistently the machine result aligns with human labels. Values that are close to 1 indicate a stronger correlation.

 Expand table

Strength of Association	Coefficient Value	Detail
Low	0.1 to 0.3	The autogenerated scores from an automatic system aren't significantly aligned with the perception of humans.
Medium	0.3 to 0.5	The autogenerated scores from an automatic system are aligned with the perception of humans, but differences still exist, and people might not agree with the result.
High	0.5 to 1.0	The autogenerated scores from an automatic system are aligned with the perception of humans, and people are willing to agree with the system results.

In our evaluations, Microsoft Pronunciation Assessment has performed >0.5 Pearson correlation with human judges' results, which indicates the autogenerated results are highly consistent with the judgment of human experts.

System limitations and best practices to improve system accuracy

- Pronunciation Assessment works better with higher-quality audio input. We recommend an input quality of 16 kHz or higher.
- Pronunciation Assessment quality is also affected by the distance of the speaker from the microphone. Recordings should be made with the speaker close to the microphone, and not over a remote connection.
- Pronunciation Assessment doesn't support a mixed lingual assessment scenario.
- Pronunciation Assessment supports a broader range of [languages](#).
- Pronunciation Assessment doesn't support a multi-speaker assessment scenario. The audio should include only one speaker for each assessment.

- Pronunciation Assessment compares the submitted audio to native speakers in general conditions. The speaker should maintain a normal speaking speed and volume, and avoid shouting or otherwise raising their voice.
- Pronunciation assessment performs better in an environment with little background noise. Current Speech-To-Text models accommodate noise in general conditions. Noisy environments or multiple people speaking at the same time might lead to lower confidence of the evaluation. To handle difficult cases better, you can suggest that the speaker should repeat a pronunciation if they score below a certain threshold.

Evaluating Pronunciation Assessment in your applications

Pronunciation Assessment's performance will vary depending on the real-world uses that customers implement. In order to ensure optimal performance in their scenarios, customers should conduct their own evaluations of the solutions they implement using Pronunciation Assessment.

- Before using Pronunciation Assessment in your applications, consider whether this product performs well in your scenario. Collect real-life data from the target scenario, test how Pronunciation Assessment performs, and make sure Speech-To-Text and Pronunciation Assessment can deliver the accuracy you need, see [Evaluate and improve Azure AI services Custom Speech accuracy](#).
- Select suitable thresholds per the target scenario. Pronunciation Assessment provides accuracy scores at different levels and you may need to consider the threshold employed in real-use. For example, the grading method for children's learning might not be as strict as that for adult learning. Consider setting a higher mispronunciation detection threshold for adult learning.

Limited Access to embedded Speech

08/17/2025

Important

Non-English translations are provided for convenience only. Please consult the [EN-US](#) version of this document for the definitive version.

Embedded Speech is designed for on-device speech to text and text to speech scenarios where cloud connectivity is intermittent or unavailable. Microsoft's embedded Speech feature is a Limited Access feature available by registration only, and only for certain use cases.

Registration process

As a Limited Access feature, embedded Speech requires registration. Only customers managed by Microsoft, meaning those who are working directly with Microsoft account teams, are eligible for access. Customers who wish to use this feature are required to register by [submitting a registration form](#). The use of embedded Speech is limited to the use case selected at the time of registration. Microsoft may require customers to re-verify this information.

The embedded Speech is made available to customers under the terms governing their subscription to Microsoft Azure Services (including the [Service Specific Terms](#)). Please review these terms carefully as they contain important conditions and obligations governing your use of embedded Speech.

Learn more about the legal terms that apply to this feature [here](#).

Help and support

FAQ about Limited Access features can be found [here](#).

If you need help with embedded Speech, find support [here](#).

Report abuse of embedded Speech [here](#).

Next steps

- [How to use embedded speech](#)
- [Limited Access with Azure AI services](#)

Azure AI Speech known issues

Azure AI Speech is updated regularly and we're continually improving and enhancing its features and capabilities. This page details known issues related to Azure AI Speech and provides steps to resolve them. Before submitting a support request, review the following list to see if your problem is already being addressed and to find a possible solution.

- For more information regarding service-level outages, see the [Azure status page](#).
- To set up outage notifications and alerts, see the [Azure Service Health Portal](#).

Active known issues speech to text (STT)

This table lists the current known issues for the Speech to text feature:

Expand table

Issue ID	Category	Title	Description	Workaround	Issues publish date
1001	Content	STT transcriptions with pound units	In certain instances, the use of pound units can pose difficulties for transcription. When phrases are spoken in a UK dialect, they're often inaccurately converted during real-time transcription, leading to the term 'pounds' being automatically translated to 'lbs' irrespective of the language setting.	Users can use Custom Display Post Processing (DPP) to train a custom speech model to correct default DPP results (for example, Pounds {tab} Pounds). Refer to Custom Rewrite Rules .	June 9, 2025
1002	Content	STT transcriptions with cardinal directions	The speech recognition model 20241218 might inaccurately interpret audio inputs that include cardinal directions, resulting in unexpected transcription outcomes. For instance, an audio file containing "SW123456" might be transcribed as "Southwest 123456," and similar errors can occur with other cardinal directions.	Potential workaround is to use Custom Display formatting where "Southwest" is mapped to "SW" in a rewrite rule: Custom Rewrite Rules .	June 9, 2025
1003	Model	STT transcriptions might include unexpected internal system tags.	Unexpected tags like 'nsnoise' have been appearing in transcription results. Initially customers reported this issue for the Arabic model (ar-SA), this issue was also observed in English models (en-US and en-GB). These tags are causing intermittent problems in the transcription outputs. To address this issue, a filter will be added to remove	N/A	June 9, 2025

Issue ID	Category	Title	Description	Workaround	Issues publish date
			'nsnoise' from the training data in future model updates.		
1004	Model	STT transcriptions with inaccurate spellings of language specific names and words	Inaccurate transcription of language specific names due to lack of entity coverage in base model for tier 2 locales (scenario specific to when our base models didn't see a specific word before).	Customers can train Custom Speech models to include unknown names and words as training data. As a second step, unknown words can be added as Phrase List at runtime. Biasing phrase list to a word known in the training corpus can greatly improve recognition accuracy.	June 9, 2025
1005	File types	Words out of context added in STT real time output occasionally	Audio files that consist solely of background noise can result in inaccurate transcriptions. Ideally, only spoken sentences should be transcribed, but this isn't occurring with the nl-NL model.	Audio files that consist of background noise, captured echo reflections from surfaces in an environment or audio playback from a device while device microphone is active can result in inaccurate transcriptions. Customers can use the Microsoft Audio Stack built into the Speech SDK for noise suppression of observed background noise and echo cancellation. This should help optimize the audio being fed to the STT service: Use the Microsoft Audio Stack (MAS) .	June 9, 2025
1006	File types	MP4 decoding failure due to 'moov atom' position	The decoding of MP4 container files might fail because the "moov atom" is located at the end of the file instead of the beginning. This structure makes the file unstreamable for the current service and the underlying Microsoft MTS service, especially for files larger than 10MB. Supporting such formats would require fundamental changes.	Preprocess the file using audio codec utilities to move the 'moov atom' to the beginning or convert to MP3.	August 8, 2025

Active known issues text to speech (TTS)

This table lists the current known issues for the Text-to-Speech feature.

[Expand table](#)

Issue ID	Category	Title	Description	Workaround	Issues publish date
2001	Service	Model copying via Rest API	The TTS service doesn't allow model copying via the REST API for disaster recovery purposes.	N/A	June 9, 2025

Issue ID	Category	Title	Description	Workaround	Issues publish date
2002	TTS Avatar	Missing parameters	TTS Avatar parameters "avatarPosition" and "avatarSize" not supported in Batch synthesis.	N/A	June 9, 2025
2003	TTS Avatar	Missing Blob file names	The 'outputs': 'result' url of Batch avatar synthesis job doesn't have the blob file name.	Customers should use 'subtitleType = soft_embedded' as a temporary workaround.	June 9, 2025
2004	TTS Avatar	Batch synthesis unsupported for TTS	Batch synthesis for avatar doesn't support bring your own storage (BYOS) and it requires the storage account to allow external traffic.	N/A	June 9, 2025
2005	Service	DNS cache refresh before end of July 2025	Due to compliance reasons, the legacy Speech TTS clusters in Asia are removed on July 31, 2025. All traffic is migrated from the old IPs to the new ones. Some customers are still accessing the old clusters even after DNS redirection was completed. This indicates that some customers may have persistent local or secondary DNS caches.	To avoid service downtime, refresh the DNS cache before the end of July 2025.	July 24, 2025
2006	TTS	Word boundary duplication in output	Azure TTS sometimes returns duplicated word boundary entries in the synthesis output, particularly when using certain SSML configurations. This can lead to inaccurate timing data and misalignment in downstream applications.	Post-process the output to filter out duplicate word boundaries based on timestamp and word content.	August 8, 2025
2007	TTS	Partially generated words in Arabic voices	Arabic voice outputs only contain partially generated words in cases of unclear or incomplete pronunciation, especially for words ending with ة or ئ. This problem is reproducible across multiple voices. The issue is acknowledged as a known problem without an immediate solution available.	To mitigate the issue consider re-phrasing the voice output, if the issue occurs.	September 16, 2025

Active known issues speech SDK/Runtime

This table lists the current known issues for the Speech SDK/Runtime feature.

[Expand table](#)

Issue ID	Category	Title	Description	Workaround	Issues publish date
3001	SDK/SR Runtime	Handling of the InitialSilenceTimeout parameter	The issue is related to the handling of the InitialSilenceTimeout	The 400 error is due to "InitialSilenceTimeout" parameter not being currently exposed directly in Real-	June 9, 2025

Issue ID	Category	Title	Description	Workaround	Issues publish date
			<p>parameter. When set to 0, it unexpectedly caused customers to encounter 400 errors. Additionally, the endSilenceTimeout parameter might lead to incorrect transcriptions.</p> <p>When the endSilenceTimeout is set to a value other than "0", the system disregards user input after the specified duration, even if the user continues speaking. Customers want all parts of the conversation to be transcribed, including segments after pauses, to ensure no user input is lost.</p>	<p>time Speech Recognition endpoint resulting in a failed URL consistency check. To bypass this error, customers can perform the following steps:</p> <ul style="list-style-type: none"> Adjust their production code to use Region/Key instantiation of SpeechConfig object. • SpeechConfig = fromSubscription (String subscriptionKey, String region); where region is the Azure Region where the Speech resource is located. • Set the parameter "InitialSilenceTimeoutMs" to 0, which in effect disables timeout due to initial silence in the recognition audio stream. <p>Note: For single shot recognition, the session will be terminated after 30 seconds of initial silence. For continuous recognition, the service will report empty phrase after 30 seconds and continue the recognition process. This issue is due to a second parameter "Speech_SegmentationMaximumTimeMs" which determines the maximum length of a phrase and has default value of 30,000 ms.</p>	
3002	SDK/SR Runtime	Handling of SegmentationTimeout parameter	Customers experience random words being generated as part of Speech recognition results (hallucinations) when the SegmentationSilenceTimeout parameter is set to > 1,000 ms.	Customers should maintain the default "SegmentationTimeout" value of 650 ms.	June 9, 2025
3003	SDK/SR Runtime	Handling of speaker duration during Real-time diarization in STT	Python SDK not showing duration of speakers when using Real-time diarization with STT.	Check offset and duration on the result following steps on the following Documentation: Conversation Transcription Result Class .	June 9, 2025
3004	SDK/TTS Avatar	Frequent disconnections with JavaScript SDK	TTS Avatar isn't loading/Frequent disconnections and reconnections of a custom avatar using the JavaScript SDK.	Customers should open the UDP 3478 port.	June 9, 2025

Recently closed known issues

Fixed known issues are organized in this section in descending order by fixed date. Fixed issues are retained for at least 60 days.

Related content

- [Azure Service Health Portal](#)
- [Azure Status overview](#)
- [What's new in Azure AI Translator?](#)

Last updated on 10/31/2025

Azure AI services support and help options

08/18/2025

Here are the options for getting support, staying up to date, giving feedback, and reporting bugs for Azure AI services.

Get solutions to common issues

In the Azure portal, you can find answers to common AI service issues.

1. Go to your Azure AI services resource in the Azure portal. You can find it on the list on this page: [Azure AI services](#). If you're a United States government customer, use the [Azure portal for the United States government](#).
2. In the left pane, under **Help**, select **Support + Troubleshooting**.
3. Describe your issue in the text box, and answer the remaining questions in the form.
4. You'll find Learn articles and other resources that might help you resolve your issue.

Create an Azure support request

A

Explore the range of Azure support options and [choose the plan](#) that best fits, whether you're a developer just starting your cloud journey or a large organization deploying business-critical, strategic applications. Azure customers can create and manage support requests in the Azure portal.

To submit a support request for Azure AI services, follow the instructions on the [New support request](#) page in the Azure portal. After choosing your **Issue type**, select **Cognitive Services** in the **Service type** dropdown field.

Post a question on Microsoft Q&A

For quick and reliable answers on your technical product questions from Microsoft Engineers, Azure Most Valuable Professionals (MVPs), or our expert community, engage with us on [Microsoft Q&A](#), Azure's preferred destination for community support.

If you can't find an answer to your problem using search, submit a new question to Microsoft Q&A. Use one of the following tags when you ask your question:

- [Azure AI services](#)

Azure OpenAI

- [Azure OpenAI](#)

Vision

- [Azure AI Vision](#)
- [Azure AI Custom Vision](#)
- [Azure Face](#)
- [Azure AI Document Intelligence](#)

Language

- [Azure AI Immersive Reader](#)
- [Azure AI Language](#)
- [Azure Translator](#)

Speech

- [Azure AI Speech](#)

Post a question to Stack Overflow



For answers to your developer questions from the largest community developer ecosystem, ask your question on Stack Overflow.

If you submit a new question to Stack Overflow, use one or more of the following tags when you create the question:

- [Azure AI services ↗](#)

Azure OpenAI

- [Azure OpenAI ↗](#)

Vision

- [Azure AI Vision ↗](#)
- [Azure AI Custom Vision ↗](#)
- [Azure Face ↗](#)
- [Azure AI Document Intelligence ↗](#)

Language

- [Azure AI Immersive Reader ↗](#)
- [Azure AI Language service ↗](#)
- [Azure Translator ↗](#)

Speech

- [Azure AI Speech service ↗](#)

Other support options

You can also use the following forums to ask questions and get help:

- Discord - <https://aka.ms/foundry/discord> ↗
- GitHub - <https://aka.ms/foundry/forum> ↗

Submit feedback

To request new features, post them on <https://feedback.azure.com>. Share your ideas for making Azure AI services and its APIs work better for the applications you develop.

- [Azure AI services ↗](#)

Vision

- [Azure AI Vision ↗](#)
- [Azure AI Custom Vision ↗](#)
- [Azure Face ↗](#)
- [Azure AI Document Intelligence ↗](#)
- [Video Indexer ↗](#)

Language

- [Azure AI Immersive Reader ↗](#)
- [Language Understanding \(LUIS\) ↗](#)
- [Azure QnA Maker ↗](#)
- [Azure AI Language ↗](#)
- [Azure Translator ↗](#)

Speech

- [Azure AI Speech service ↗](#)

Decision

- [Azure AI Anomaly Detector](#)
- [Content Moderator](#)
- [Azure AI Metrics Advisor](#)
- [Azure AI Personalizer](#)

Stay informed

You can learn about the features in a new release or get the latest news on the Azure blog. Staying informed can help you find the difference between a programming error, a service bug, or a feature not yet available in Azure AI services.

- Learn more about product updates, roadmap, and announcements in [Azure Updates](#).
- News about Azure AI services is shared in the [Azure AI blog](#).
- [Join the conversation on Reddit](#) about Azure AI services.

Next step

[What are Azure AI services?](#)