

Text to speech


Learn how to turn text into lifelike spoken audio

Introduction

The Audio API provides a `speech` endpoint based on our [TTS \(text-to-speech\) model](#). It comes with 6 built-in voices and can be used to:

- Narrate a written blog post
- Produce spoken audio in multiple languages
- Give real time audio output using streaming

Here is an example of the `alloy` voice:

 Please note that our [usage policies](#) require you to provide a clear disclosure to end users that the TTS voice they are hearing is AI-generated and not a human voice.

Quick start

The `speech` endpoint takes in three key inputs: the `model`, the `text` that should be turned into audio, and the `voice` to be used for the audio generation. A simple request would look like the following:

Generate spoken audio from input text

python ▾



```
1 from pathlib import Path
2 from openai import OpenAI
3 client = OpenAI()
4
5 speech_file_path = Path(__file__).parent / "speech.mp3"
6 response = client.audio.speech.create(
7     model="tts-1",
8     voice="alloy",
9     input="Today is a wonderful day to build something people love!"
10 )
11
12 response.stream_to_file(speech_file_path)
```

By default, the endpoint will output a MP3 file of the spoken audio but it can also be configured to output any of our [supported formats](#).

Audio quality

For real-time applications, the standard `tts-1` model provides the lowest latency but at a lower quality than the `tts-1-hd` model. Due to the way the audio is generated, `tts-1` is likely to generate content that has more static in certain situations than `tts-1-hd`. In some cases, the audio may not have noticeable differences depending on your listening device and the individual person.

Voice options

Experiment with different voices (`alloy`, `echo`, `fable`, `onyx`, `nova`, and `shimmer`) to find one that matches your desired tone and audience. The current voices are optimized for English.

Alloy



Supported output formats

The default response format is "mp3", but other formats like "opus", "aac", "flac", and "pcm" are available.

- **Opus:** For internet streaming and communication, low latency.
- **AAC:** For digital audio compression, preferred by YouTube, Android, iOS.
- **FLAC:** For lossless audio compression, favored by audio enthusiasts for archiving.
- **WAV:** Uncompressed WAV audio, suitable for low-latency applications to avoid decoding overhead.
- **PCM:** Similar to WAV but containing the raw samples in 24kHz (16-bit signed, low-endian), without the header.

Supported languages

The TTS model generally follows the Whisper model in terms of language support. Whisper [supports the following languages](#) and performs well despite the current voices being optimized for English:

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, and Welsh.

You can generate spoken audio in these languages by providing the input text in the language of your choice.

Streaming real time audio

The Speech API provides support for real time audio streaming using [chunk transfer encoding](#). This means that the audio is able to be played before the full file has been generated and made accessible.

```
1 from openai import OpenAI
2
3 client = OpenAI()
4
5 response = client.audio.speech.create(
6     model="tts-1",
7     voice="alloy",
8     input="Hello world! This is a streaming test.",
9 )
10
11 response.stream_to_file("output.mp3")
```

FAQ

How can I control the emotional range of the generated audio?

There is no direct mechanism to control the emotional output of the audio generated. Certain factors may influence the output audio like capitalization or grammar but our internal tests with these have yielded mixed results.

Can I create a custom copy of my own voice?

No, this is not something we support.

Do I own the outputted audio files?

Yes, like with all outputs from our API, the person who created them owns the output. You are still required to inform end users that they are hearing audio generated by AI and not a real person talking to them.