Audio generation

Copy page

Learn how to generate audio from a text or audio prompt.

In addition to generating text and images, some models let you generate spoken audio and prompt the model with audio. With richer data than text alone, audio lets the model detect tone, inflection, and other nuances in the input.

You can use audio capabilities to:

- Generate a spoken audio summary of a body of text (text in, audio out)
- Perform sentiment analysis on a recording (audio in, text out)
- Async speech-to-speech interactions with a model (audio in, audio out)

OpenAI provides other models for simple speech-to-text (STT) and text-to-speech (TTS). Use STT and TTS if you don't need to generate dynamic audio, as those models are more performant and cost-efficient.

Quickstart

To generate audio or use audio as an input, use the chat completions endpoint. You can either use the REST API from the HTTP client of your choice or one of OpenAI's official SDKs.

Audio output from model | Audio input to model

Create a human-like audio response to a prompt

javascript

```javascript
import { writeFileSync } from "node:fs";
import OpenAI from "openai";

const openai = new OpenAI();

// Generate an audio response to the given prompt
const response = await openai.chat.completions.create({
  model: "gpt-4o-audio-preview",
  modalities: ["text", "audio"],
  audio: { voice: "alloy", format: "wav" },
  messages: [
    {
      role: "user",
      content: "Is a golden retriever a good family dog?"
    }
  ],
```

```
  store: true,
});

// Inspect returned data
console.log(response.choices[0]);

// Write audio data to a file
writeFileSync(
  "dog.wav",
  Buffer.from(response.choices[0].message.audio.data, 'base64'),
  { encoding: "utf-8" }
);
```

Multi-turn conversations

Using audio outputs from the model as inputs to **multi-turn conversations** requires a generated ID. Find this ID in the response data for an audio generation. Here's an example of a message you might receive from `/chat/completions` in a JSON data structure:

```
{
  "index": 0,
  "message": {
    "role": "assistant",
    "content": null,
    "refusal": null,
    "audio": {
      "id": "audio_abc123",
      "expires_at": 1729018505,
      "data": "<bytes omitted>",
      "transcript": "Yes, golden retrievers are known to be ..."
    }
  },
  "finish_reason": "stop"
}
```

The value of `message.audio.id` above provides an identifier you can use in an `assistant` message for a new `/chat/completions` request, as in the example below.

```
curl "https://api.openai.com/v1/chat/completions" \
    -H "Content-Type: application/json" \
    -H "Authorization: Bearer $OPENAI_API_KEY" \
    -d '{
        "model": "gpt-4o-audio-preview",
        "modalities": ["text", "audio"],
        "audio": { "voice": "alloy", "format": "wav" },
        "messages": [
            {
                "role": "user",
                "content": "Is a golden retriever a good family dog?"
            },
            {
```

```
                "role": "assistant",
                "audio": {
                    "id": "audio_abc123"
                }
            },
            {
                "role": "user",
                "content": "Why do you say they are loyal?"
            }
        ]
    }'
```

FAQ

Which modalities does gpt-4o-audio-preview support?

Currently, `gpt-4o-audio-preview` requires either audio output or audio input. Acceptable combinations of input and output are:

Text in → text + audio out

Audio in → text + audio out

Audio in → text out

Text + audio in → text + audio out

Text + audio in → text out

How is audio in chat completions different from the Realtime API?

The underlying GPT-4o audio model is exactly the same. The Realtime API operates the same model at lower latency.

How do I think about audio input to the model in terms of tokens?

We're working on better tooling to expose this, but roughly one hour of audio input equals 128k tokens, the max context window currently supported by this model.

How do I control which output modalities I receive?

The model programmatically allows modalities = `["text", "audio"]`. In the future, this parameter will give more controls.

How does tool/function calling work?

Tool and function calling works the same here as other models in chat completions. Learn more.

Next steps

Now that you know how to generate audio outputs and send audio inputs, you might want to learn a few other techniques.

Text to speech

Use a specialized model to turn text into speech.

Speech to text

Use a specialized model to turn audio files with speech into text.

Realtime API

Learn to use the Realtime API to prompt a model over a WebSocket.

Full API reference

Check out all the options for audio generation in the API reference.