Learn more (https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/)

Explore audio capabilities with the Gemini API

Python ✓ JavaScript Go REST

Gemini can respond to prompts about audio. For example, Gemini can:

- Describe, summarize, or answer questions about audio content.
- Provide a transcription of the audio.
- Provide answers or a transcription about a specific segment of the audio.

Note: You can't generate audio *output* with the Gemini API.

This guide demonstrates different ways to interact with audio files and audio content using the Gemini API.

Before you begin

Before calling the Gemini API, ensure you have your SDK of choice (/gemini-api/docs/downloads) installed, and a Gemini API key (/gemini-api/docs/api-key) configured and ready to use.

Supported audio formats

Gemini supports the following audio format MIME types:

- WAV - `audio/wav`
- MP3 - `audio/mp3`
- AIFF - `audio/aiff`
- AAC - `audio/aac`
- OGG Vorbis - `audio/ogg`
- FLAC - `audio/flac`

Technical details about audio

Gemini imposes the following rules on audio:

- Gemini represents each second of audio as 32 tokens; for example, one minute of audio is represented as 1,920 tokens.
- Gemini can only infer responses to English-language speech.
- Gemini can "understand" non-speech components, such as birdsong or sirens.
- The maximum supported length of audio data in a single prompt is 9.5 hours. Gemini doesn't limit the *number* of audio files in a single prompt; however, the total combined length of all audio files in a single prompt cannot exceed 9.5 hours.
- Gemini downsamples audio files to a 16 Kbps data resolution.
- If the audio source contains multiple channels, Gemini combines those channels down to a single channel.

Make an audio file available to Gemini

You can make an audio file available to Gemini in either of the following ways:

- Upload (#upload-audio) the audio file *prior* to making the prompt request.
- Provide the audio file as inline data (#inline-data) to the prompt request.

Upload an audio file and generate content

You can use the File API to upload an audio file of any size. Always use the File API when the total request size (including the files, text prompt, system instructions, etc.) is larger than 20 MB.

Note: The File API lets you store up to 20 GB of files per project, with a per-file maximum size of 2 GB. Files are stored for 48 hours. They can be accessed in that period with your API key, but cannot be downloaded from the API. The File API is available at no cost in all regions where the Gemini API is available.

Call **`media.upload`** (/api/rest/v1beta/media/upload) to upload a file using the File API. The following code uploads an audio file and then uses the file in a call to **`models.generateContent`** (/api/generate-content#method:-models.generatecontent).

```
// Make sure to include the following import:
// import {GoogleGenAI} from '@google/genai';
const ai = new GoogleGenAI({ apiKey: process.env.GEMINI_API_KEY });
const myfile = await ai.files.upload({
  file: path.join(media, "sample.mp3"),
  config: { mimeType: "audio/mpeg" },
});
console.log("Uploaded file:", myfile);

const result = await ai.models.generateContent({
  model: "gemini-2.0-flash",
  contents: createUserContent([
    createPartFromUri(myfile.uri, myfile.mimeType),
    "Describe this audio clip",
  ]),
});
console.log("result.text=", result.text);
```

`files.js` (https://github.com/google-gemini/api-examples/blob/0d35c672a03be147b83251874b61a8aabce1d0bc/javascript/files.js#L83-L99)

Get metadata for a file

You can verify the API successfully stored the uploaded file and get its metadata by calling **`files.get`** (/api/rest/v1beta/files/get).

```
// Make sure to include the following import:
// import {GoogleGenAI} from '@google/genai';
const ai = new GoogleGenAI({ apiKey: process.env.GEMINI_API_KEY });
const myfile = await ai.files.upload({
  file: path.join(media, "poem.txt"),
});
const fileName = myfile.name;
console.log(fileName);

const fetchedFile = await ai.files.get({ name: fileName });
```

```
console.log(fetchedFile);
```

`files.js` (https://github.com/google-gemini/api-examples/blob/0d35c672a03be147b83251874b61a8aabce1d0bc/javascript/files.js#L180-L190)

List uploaded files

You can upload multiple audio files (and other kinds of files). The following code generates a list of all the files uploaded:

```
// Make sure to include the following import:
// import {GoogleGenAI} from '@google/genai';
const ai = new GoogleGenAI({ apiKey: process.env.GEMINI_API_KEY });
console.log("My files:");
// Using the pager style to list files
const pager = await ai.files.list({ config: { pageSize: 10 } });
let page = pager.page;
const names = [];
while (true) {
  for (const f of page) {
    console.log("  ", f.name);
    names.push(f.name);
  }
  if (!pager.hasNextPage()) break;
  page = await pager.nextPage();
}
```

`files.js` (https://github.com/google-gemini/api-examples/blob/0d35c672a03be147b83251874b61a8aabce1d0bc/javascript/files.js#L158-L173)

Provide the audio file as inline data in the request

Instead of uploading an audio file, you can pass audio data in the same call that contains the prompt.

Then, pass that downloaded small audio file along with the prompt to Gemini:

```
import { GoogleGenAI } from "@google/genai";
import * as fs from "node:fs";

const ai = new GoogleGenAI({ apiKey: "GEMINI_API_KEY" });
const base64AudioFile = fs.readFileSync("path/to/small-sample.mp3", {
  encoding: "base64",
});

const contents = [
  { text: "Please summarize the audio." },
  {
    inlineData: {
      mimeType: "audio/mpeg",
      data: base64AudioFile,
    },
  },
];

const response = await ai.models.generateContent({
  model: "gemini-2.0-flash",
  contents: contents,
});
console.log(response.text);
```

Note the following about providing audio as inline data:

- The maximum request size is 20 MB, which includes text prompts, system instructions, and files provided inline. If your file's size will make the *total request size* exceed 20 MB, then use the File API (#upload-audio) to upload files for use in requests.
- If you're using an audio sample multiple times, it is more efficient to use the File API (#upload-audio).

More ways to work with audio

This section provides a few additional ways to get more from audio.

Get a transcript of the audio file

To get a transcript, just ask for it in the prompt. For example:

```
import {
  GoogleGenAI,
  createUserContent,
  createPartFromUri,
} from "@google/genai";

const ai = new GoogleGenAI({ apiKey: "GEMINI_API_KEY" });
const myfile = await ai.files.upload({
  file: "path/to/sample.mp3",
  config: { mimeType: "audio/mpeg" },
});
```

```
const result = await ai.models.generateContent({
  model: "gemini-2.0-flash",
  contents: createUserContent([
    createPartFromUri(myfile.uri, myfile.mimeType),
    "Generate a transcript of the speech.",
  ]),
});
console.log("result.text=", result.text);
```

Refer to timestamps in the audio file

A prompt can specify timestamps of the form `MM:SS` to refer to particular sections in an audio file. For example, the following prompt requests a transcript that:

- Starts at 2 minutes 30 seconds from the beginning of the file.
- Ends at 3 minutes 29 seconds from the beginning of the file.

```
// Create a prompt containing timestamps.
const prompt = "Provide a transcript of the speech from 02:30 to 03:29."
```

Count tokens

Call the **`countTokens`** (/api/rest/v1/models/countTokens) method to get a count of the number of tokens in the audio file. For example:

```
import {
  GoogleGenAI,
  createUserContent,
  createPartFromUri,
} from "@google/genai";

const ai = new GoogleGenAI({ apiKey: "GEMINI_API_KEY" });
const myfile = await ai.files.upload({
  file: "path/to/sample.mp3",
  config: { mimeType: "audio/mpeg" },
});

const countTokensResponse = await ai.models.countTokens({
  model: "gemini-2.0-flash",
  contents: createUserContent([
    createPartFromUri(myfile.uri, myfile.mimeType),
  ]),
});
console.log(countTokensResponse.totalTokens);
```

What's next

This guide shows how to upload audio files using the File API and then generate text outputs from audio inputs. To learn more, see the following resources:

- File prompting strategies (/gemini-api/docs/file-prompting-strategies): The Gemini API supports prompting with text, image, audio, and video data, also known as multimodal prompting.

- System instructions (/gemini-api/docs/system-instructions): System instructions let you steer the behavior of the model based on your specific needs and use cases.
- Safety guidance (/gemini-api/docs/safety-guidance): Sometimes generative AI models produce unexpected outputs, such as outputs that are inaccurate, biased, or offensive. Post-processing and human evaluation are essential to limit the risk of harm from such outputs.