Introduction

You can interact with the API through HTTP requests from any language, via our official Python bindings, our official Node.js library, or a **community-maintained library**.

To install the official Python bindings, run the following command:

```
pip install openai
```

To install the official Node.js library, run the following command in your Node.js project directory:

```
npm install openai
```

Authentication

The OpenAI API uses API keys for authentication. Visit your **API Keys** page to retrieve the API key you'll use in your requests.

Remember that your API key is a secret! Do not share it with others or expose it in any client-side code (browsers, apps). Production requests must be routed through your own backend server where your API key can be securely loaded from an environment variable or key management service.

All API requests should include your API key in an `Authorization` HTTP header as follows:

```
Authorization: Bearer OPENAI_API_KEY
```

Requesting organization

For users who belong to multiple organizations, you can pass a header to specify which organization is used for an API request. Usage from these API requests will count against the specified organization's subscription quota.

Example curl command:

```
curl https://api.openai.com/v1/models \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -H "OpenAI-Organization: org-VTK2lnpHmoMEGoDTXipj3Q9G"
```

Example with the `openai` Python package:

```
import os
import openai
openai.organization = "org-VTK2lnpHmoMEGoDTXipj3Q9G"
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Model.list()
```

Example with the `openai` Node.js package:

```
import { Configuration, OpenAIApi } from "openai";
const configuration = new Configuration({
    organization: "org-VTK2lnpHmoMEGoDTXipj3Q9G",
    apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);
const response = await openai.listEngines();
```

Organization IDs can be found on your **Organization settings** page.

Making requests

You can paste the command below into your terminal to run your first API request. Make sure to replace `$OPENAI_API_KEY` with your secret API key.

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
     "model": "gpt-3.5-turbo",
     "messages": [{"role": "user", "content": "Say this is a test!"}],
```

```
    "temperature": 0.7
  }'
```

This request queries the `gpt-3.5-turbo` model to complete the text starting with a prompt of "*Say this is a test*". You should get a response back that resembles the following:

```
{
   "id":"chatcmpl-abc123",
   "object":"chat.completion",
   "created":1677858242,
   "model":"gpt-3.5-turbo-0301",
   "usage":{
      "prompt_tokens":13,
      "completion_tokens":7,
      "total_tokens":20
   },
   "choices":[
      {
         "message":{
            "role":"assistant",
            "content":"\n\nThis is a test!"
         },
         "finish_reason":"stop",
         "index":0
      }
   ]
}
```

Now you've generated your first chat completion. We can see the `finish_reason` is `stop` which means the API returned the full completion generated by the model. In the above request, we only generated a single message but you can set the `n` parameter to generate multiple messages choices.

Models

List and describe the various models available in the API. You can refer to the **Models** documentation to understand what models are available and the differences between them.

List models

`GET https://api.openai.com/v1/models`

Lists the currently available models, and provides basic information about each one such as the owner and availability.

Example request — python — Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Model.list()
```

Response — Copy

```
{
  "data": [
    {
      "id": "model-id-0",
      "object": "model",
      "owned_by": "organization-owner",
      "permission": [...]
    },
    {
      "id": "model-id-1",
      "object": "model",
      "owned_by": "organization-owner",
      "permission": [...]
    },
    {
      "id": "model-id-2",
      "object": "model",
      "owned_by": "openai",
      "permission": [...]
    },
  ],
  "object": "list"
}
```

Retrieve model

```
GET https://api.openai.com/v1/models/{model}
```

Retrieves a model instance, providing basic information about the model such as the owner and permissioning.

Path parameters

`model` string Required

The ID of the model to use for this request

Example request text-davinci-003 python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Model.retrieve("text-davinci-003")
```

Response text-davinci-003 Copy

```
{
  "id": "text-davinci-003",
  "object": "model",
  "owned_by": "openai",
  "permission": [...]
}
```

Chat

Given a list of messages comprising a conversation, the model will return a response.

Create chat completion

```
POST https://api.openai.com/v1/chat/completions
```

Creates a model response for the given chat conversation.

Request body

`model` string Required

ID of the model to use. See the **model endpoint compatibility** table for details on which models work with the Chat API.

`messages` array Required

A list of messages comprising the conversation so far. **Example Python code**.

`role` string Required

The role of the messages author. One of `system`, `user`, `assistant`, or `function`.

`content` string Required

The contents of the message. `content` is required for all messages, and may be null for assistant messages with function calls.

`name` string Optional

The name of the author of this message. `name` is required if role is `function`, and it should be the name of the function whose response is in the `content`. May contain a-z, A-Z, 0-9, and underscores, with a maximum length of 64 characters.

`function_call` object Optional

The name and arguments of a function that should be called, as generated by the model.

`functions` array Optional

A list of functions the model may generate JSON inputs for.

`name` string Required

The name of the function to be called. Must be a-z, A-Z, 0-9, or contain underscores and dashes, with a maximum length of 64.

`description` string Optional

A description of what the function does, used by the model to choose when and how to call the function.

`parameters` object Required

The parameters the functions accepts, described as a JSON Schema object. See the **guide** for examples, and the **JSON Schema reference** for documentation about the format.

To describe a function that accepts no parameters, provide the value `{"type": "object", "properties": {}}`.

`function_call` string or object Optional

Controls how the model responds to function calls. "none" means the model does not call a function, and responds to the end-user. "auto" means the model can pick between an end-user or calling a function. Specifying a particular function via `{"name":\ "my_function"}` forces the model to call that function. "none" is the default when no functions are present. "auto" is the default if functions are present.

`temperature` number Optional Defaults to 1

What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

We generally recommend altering this or `top_p` but not both.

`top_p` number Optional Defaults to 1

An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

We generally recommend altering this or `temperature` but not both.

`n` integer Optional Defaults to 1

How many chat completion choices to generate for each input message.

`stream` boolean Optional Defaults to false

If set, partial message deltas will be sent, like in ChatGPT. Tokens will be sent as data-only **server-sent events** as they become available, with the stream terminated by a `data: [DONE]` message. **Example Python code**.

`stop` string or array Optional Defaults to null

Up to 4 sequences where the API will stop generating further tokens.

`max_tokens` integer Optional Defaults to inf

The maximum number of **tokens** to generate in the chat completion.

The total length of input tokens and generated tokens is limited by the model's context length. **Example Python code** for counting tokens.

`presence_penalty` number Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

See more information about frequency and presence penalties.

`frequency_penalty` number Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

See more information about frequency and presence penalties.

`logit_bias` map Optional Defaults to null

Modify the likelihood of specified tokens appearing in the completion.

Accepts a json object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request gpt-3.5-turbo python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")

completion = openai.ChatCompletion.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ]
)

print(completion.choices[0].message)
```

Parameters gpt-3.5-turbo Copy

```
{
  "model": "gpt-3.5-turbo",
  "messages": [{"role": "system", "content": "You are a helpful assistant."}, {"role
}
```

Response Copy

```
{
  "id": "chatcmpl-123",
  "object": "chat.completion",
  "created": 1677652288,
  "choices": [{
    "index": 0,
    "message": {
      "role": "assistant",
      "content": "\n\nHello there, how may I assist you today?",
    },
    "finish_reason": "stop"
  }],
  "usage": {
    "prompt_tokens": 9,
    "completion_tokens": 12,
    "total_tokens": 21
  }
}
```

Completions

Given a prompt, the model will return one or more predicted completions, and can also return the probabilities of alternative tokens at each position. Note: We recommend most users use our Chat Completions API. **Learn more**

Create completion Legacy

`POST https://api.openai.com/v1/completions`

Creates a completion for the provided prompt and parameters.

Request body

`model` string Required

ID of the model to use. You can use the **List models** API to see all of your available models, or see our **Model overview** for descriptions of them.

`prompt` string or array Required

The prompt(s) to generate completions for, encoded as a string, array of strings, array of tokens, or array of token arrays.

Note that <|endoftext|> is the document separator that the model sees during training, so if a prompt is not specified the model will generate as if from the beginning of a new document.

`suffix` string Optional Defaults to null

The suffix that comes after a completion of inserted text.

`max_tokens` integer Optional Defaults to 16

The maximum number of **tokens** to generate in the completion.

The token count of your prompt plus `max_tokens` cannot exceed the model's context length. **Example Python code** for counting tokens.

`temperature` number Optional Defaults to 1

What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

We generally recommend altering this or `top_p` but not both.

`top_p` number Optional Defaults to 1

An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

We generally recommend altering this or `temperature` but not both.

`n` integer Optional Defaults to 1

How many completions to generate for each prompt.

Note: Because this parameter generates many completions, it can quickly consume your token quota. Use carefully and ensure that you have reasonable settings for `max_tokens` and `stop`.

`stream` boolean Optional Defaults to false

Whether to stream back partial progress. If set, tokens will be sent as data-only **server-sent events** as they become available, with the stream terminated by a `data: [DONE]` message. **Example Python code**.

`logprobs` integer Optional Defaults to null

Include the log probabilities on the `logprobs` most likely tokens, as well the chosen tokens. For example, if `logprobs` is 5, the API will return a list of the 5 most likely tokens. The API will always return the `logprob` of the sampled token, so there may be up to `logprobs+1` elements in the response.

The maximum value for `logprobs` is 5.

`echo` boolean Optional Defaults to false

Echo back the prompt in addition to the completion

`stop` string or array Optional Defaults to null

Up to 4 sequences where the API will stop generating further tokens. The returned text will not contain the stop sequence.

`presence_penalty` number Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

See more information about frequency and presence penalties.

`frequency_penalty` number Optional Defaults to 0

Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

See more information about frequency and presence penalties.

`best_of` integer Optional Defaults to 1

Generates `best_of` completions server-side and returns the "best" (the one with the highest log probability per token). Results cannot be streamed.

When used with `n`, `best_of` controls the number of candidate completions and `n` specifies how many to return – `best_of` must be greater than `n`.

Note: Because this parameter generates many completions, it can quickly consume your token quota. Use carefully and ensure that you have reasonable settings for `max_tokens` and `stop`.

`logit_bias` map Optional Defaults to null

Modify the likelihood of specified tokens appearing in the completion.

Accepts a json object that maps tokens (specified by their token ID in the GPT tokenizer) to an associated bias value from -100 to 100. You can use this **tokenizer tool** (which works for both GPT-2 and GPT-3) to convert text to token IDs. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

As an example, you can pass `{"50256": -100}` to prevent the <|endoftext|> token from being generated.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request text-davinci-003 python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Completion.create(
  model="text-davinci-003",
  prompt="Say this is a test",
  max_tokens=7,
  temperature=0
)
```

Parameters text-davinci-003 Copy

```json
{
  "model": "text-davinci-003",
  "prompt": "Say this is a test",
  "max_tokens": 7,
  "temperature": 0,
  "top_p": 1,
  "n": 1,
  "stream": false,
  "logprobs": null,
  "stop": "\n"
}
```

Response text-davinci-003 Copy

```json
{
  "id": "cmpl-uqkvlQyYK7bGYrRHQ0eXlWi7",
```

```
  "object": "text_completion",
  "created": 1589478378,
  "model": "text-davinci-003",
  "choices": [
    {
      "text": "\n\nThis is indeed a test",
      "index": 0,
      "logprobs": null,
      "finish_reason": "length"
    }
  ],
  "usage": {
    "prompt_tokens": 5,
    "completion_tokens": 7,
    "total_tokens": 12
  }
}
```

Images

Given a prompt and/or an input image, the model will generate a new image.

Related guide: **Image generation**

Create image

```
POST https://api.openai.com/v1/images/generations
```

Creates an image given a prompt.

Request body

`prompt` string Required

A text description of the desired image(s). The maximum length is 1000 characters.

`n` integer Optional Defaults to 1

The number of images to generate. Must be between 1 and 10.

`size` string Optional Defaults to 1024x1024

The size of the generated images. Must be one of `256x256`, `512x512`, or `1024x1024`.

`response_format` string Optional Defaults to url

The format in which the generated images are returned. Must be one of `url` or `b64_json`.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Image.create(
  prompt="A cute baby sea otter",
  n=2,
  size="1024x1024"
)
```

Parameters Copy

```json
{
  "prompt": "A cute baby sea otter",
  "n": 2,
  "size": "1024x1024"
}
```

Response Copy

```json
{
  "created": 1589478378,
  "data": [
    {
      "url": "https://..."
    },
    {
      "url": "https://..."
    }
  ]
}
```

Create image edit

`POST https://api.openai.com/v1/images/edits`

Creates an edited or extended image given an original image and a prompt.

Request body

`image` string Required

The image to edit. Must be a valid PNG file, less than 4MB, and square. If mask is not provided, image must have transparency, which will be used as the mask.

`mask` string Optional

An additional image whose fully transparent areas (e.g. where alpha is zero) indicate where `image` should be edited. Must be a valid PNG file, less than 4MB, and have the same dimensions as `image`.

`prompt` string Required

A text description of the desired image(s). The maximum length is 1000 characters.

`n` integer Optional Defaults to 1

The number of images to generate. Must be between 1 and 10.

`size` string Optional Defaults to 1024x1024

The size of the generated images. Must be one of `256x256`, `512x512`, or `1024x1024`.

`response_format` string Optional Defaults to url

The format in which the generated images are returned. Must be one of `url` or `b64_json`.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
```

```
openai.Image.create_edit(
  image=open("otter.png", "rb"),
  mask=open("mask.png", "rb"),
  prompt="A cute baby sea otter wearing a beret",
  n=2,
  size="1024x1024"
)
```

Response

Copy

```
{
  "created": 1589478378,
  "data": [
    {
      "url": "https://..."
    },
    {
      "url": "https://..."
    }
  ]
}
```

Create image variation

`POST https://api.openai.com/v1/images/variations`

Creates a variation of a given image.

Request body

`image` string Required

The image to use as the basis for the variation(s). Must be a valid PNG file, less than 4MB, and square.

`n` integer Optional Defaults to 1

The number of images to generate. Must be between 1 and 10.

`size` string Optional Defaults to 1024x1024

The size of the generated images. Must be one of `256x256`, `512x512`, or `1024x1024`.

`response_format` string Optional Defaults to url

The format in which the generated images are returned. Must be one of `url` or `b64_json`.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Image.create_variation(
  image=open("otter.png", "rb"),
  n=2,
  size="1024x1024"
)
```

Response Copy

```
{
  "created": 1589478378,
  "data": [
    {
      "url": "https://..."
    },
    {
      "url": "https://..."
    }
  ]
}
```

Embeddings

Get a vector representation of a given input that can be easily consumed by machine learning models and algorithms.

Related guide: **Embeddings**

Create embeddings

`POST https://api.openai.com/v1/embeddings`

Creates an embedding vector representing the input text.

Request body

`model` string Required

ID of the model to use. You can use the **List models** API to see all of your available models, or see our **Model overview** for descriptions of them.

`input` string or array Required

Input text to embed, encoded as a string or array of tokens. To embed multiple inputs in a single request, pass an array of strings or array of token arrays. Each input must not exceed the max input tokens for the model (8191 tokens for `text-embedding-ada-002`). **Example Python code** for counting tokens.

`user` string Optional

A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. **Learn more**.

Example request — python — Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Embedding.create(
  model="text-embedding-ada-002",
  input="The food was delicious and the waiter..."
)
```

Parameters — Copy

```
{
  "model": "text-embedding-ada-002",
  "input": "The food was delicious and the waiter..."
}
```

Response

Copy

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "embedding": [
        0.0023064255,
        -0.009327292,
        .... (1536 floats total for ada-002)
        -0.0028842222,
      ],
      "index": 0
    }
  ],
  "model": "text-embedding-ada-002",
  "usage": {
    "prompt_tokens": 8,
    "total_tokens": 8
  }
}
```

Audio

Learn how to turn audio into text.

Related guide: **Speech to text**

Create transcription

`POST https://api.openai.com/v1/audio/transcriptions`

Transcribes audio into the input language.

Request body

`file` file Required

The audio file object (not file name) to transcribe, in one of these formats: mp3, mp4, mpeg, mpga, m4a, wav, or webm.

`model` string Required

ID of the model to use. Only `whisper-1` is currently available.

`prompt` string Optional

An optional text to guide the model's style or continue a previous audio segment. The **prompt** should match the audio language.

`response_format` string Optional Defaults to json

The format of the transcript output, in one of these options: json, text, srt, verbose_json, or vtt.

`temperature` number Optional Defaults to 0

The sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use **log probability** to automatically increase the temperature until certain thresholds are hit.

`language` string Optional

The language of the input audio. Supplying the input language in **ISO-639-1** format will improve accuracy and latency.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
audio_file = open("audio.mp3", "rb")
transcript = openai.Audio.transcribe("whisper-1", audio_file)
```

Parameters Copy

```
{
  "file": "audio.mp3",
  "model": "whisper-1"
}
```

Response Copy

```
{
  "text": "Imagine the wildest idea that you've ever had, and you're curious about h
}
```

Create translation

`POST https://api.openai.com/v1/audio/translations`

Translates audio into English.

Request body

`file` file Required

The audio file object (not file name) translate, in one of these formats: mp3, mp4, mpeg, mpga, m4a, wav, or webm.

`model` string Required

ID of the model to use. Only `whisper-1` is currently available.

`prompt` string Optional

An optional text to guide the model's style or continue a previous audio segment. The **prompt** should be in English.

`response_format` string Optional Defaults to json

The format of the transcript output, in one of these options: json, text, srt, verbose_json, or vtt.

`temperature` number Optional Defaults to 0

The sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use **log probability** to automatically increase the temperature until certain thresholds are hit.

Example request python Copy

```
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
```

```
audio_file = open("german.m4a", "rb")
transcript = openai.Audio.translate("whisper-1", audio_file)
```

Parameters Copy

```
{
  "file": "german.m4a",
  "model": "whisper-1"
}
```

Response Copy

```
{
  "text": "Hello, my name is Wolfgang and I come from Germany. Where are you heading
}
```

Files

Files are used to upload documents that can be used with features like **Fine-tuning**.

List files

```
GET https://api.openai.com/v1/files
```

Returns a list of files that belong to the user's organization.

Example request python Copy

```
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.File.list()
```

Response Copy

```
  {
    "data": [
```

```
    {
      "id": "file-ccdDZrC3iZVNiQVeEA6Z66wf",
      "object": "file",
      "bytes": 175,
      "created_at": 1613677385,
      "filename": "train.jsonl",
      "purpose": "search"
    },
    {
      "id": "file-XjGxS3KTG0uNmNOK362iJua3",
      "object": "file",
      "bytes": 140,
      "created_at": 1613779121,
      "filename": "puppy.jsonl",
      "purpose": "search"
    }
  ],
  "object": "list"
}
```

Upload file

`POST https://api.openai.com/v1/files`

Upload a file that contains document(s) to be used across various endpoints/features. Currently, the size of all the files uploaded by one organization can be up to 1 GB. Please contact us if you need to increase the storage limit.

Request body

`file` string Required

Name of the **JSON Lines** file to be uploaded.

If the `purpose` is set to "fine-tune", each line is a JSON record with "prompt" and "completion" fields representing your **training examples**.

`purpose` string Required

The intended purpose of the uploaded documents.

Use "fine-tune" for **Fine-tuning**. This allows us to validate the format of the uploaded file.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.File.create(
  file=open("mydata.jsonl", "rb"),
  purpose='fine-tune'
)
```

Response Copy

```
{
  "id": "file-XjGxS3KTG0uNmNOK362iJua3",
  "object": "file",
  "bytes": 140,
  "created_at": 1613779121,
  "filename": "mydata.jsonl",
  "purpose": "fine-tune"
}
```

Delete file

`DELETE https://api.openai.com/v1/files/{file_id}`

Delete a file.

Path parameters

`file_id` string Required

The ID of the file to use for this request

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.File.delete("file-XjGxS3KTG0uNmNOK362iJua3")
```

Response Copy

```
{
  "id": "file-XjGxS3KTG0uNmNOK362iJua3",
  "object": "file",
  "deleted": true
}
```

Retrieve file

```
GET https://api.openai.com/v1/files/{file_id}
```

Returns information about a specific file.

Path parameters

`file_id` string Required

The ID of the file to use for this request

Example request python Copy

```
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.File.retrieve("file-XjGxS3KTG0uNmNOK362iJua3")
```

Response Copy

```
{
  "id": "file-XjGxS3KTG0uNmNOK362iJua3",
  "object": "file",
  "bytes": 140,
  "created_at": 1613779657,
  "filename": "mydata.jsonl",
  "purpose": "fine-tune"
}
```

Retrieve file content

`GET https://api.openai.com/v1/files/{file_id}/content`

Returns the contents of the specified file

Path parameters

`file_id` string Required

The ID of the file to use for this request

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
content = openai.File.download("file-XjGxS3KTG0uNmNOK362iJua3")
```

Fine-tunes

Manage fine-tuning jobs to tailor a model to your specific training data.

Related guide: **Fine-tune models**

Create fine-tune

`POST https://api.openai.com/v1/fine-tunes`

Creates a job that fine-tunes a specified model from a given dataset.

Response includes details of the enqueued job including job status and the name of the fine-tuned models once complete.

Learn more about Fine-tuning

Request body

`training_file` string Required

The ID of an uploaded file that contains training data.

See **upload file** for how to upload a file.

Your dataset must be formatted as a JSONL file, where each training example is a JSON object with the keys "prompt" and "completion". Additionally, you must upload your file with the purpose `fine-tune`.

See the **fine-tuning guide** for more details.

`validation_file` string Optional

The ID of an uploaded file that contains validation data.

If you provide this file, the data is used to generate validation metrics periodically during fine-tuning. These metrics can be viewed in the **fine-tuning results file**. Your train and validation data should be mutually exclusive.

Your dataset must be formatted as a JSONL file, where each validation example is a JSON object with the keys "prompt" and "completion". Additionally, you must upload your file with the purpose `fine-tune`.

See the **fine-tuning guide** for more details.

`model` string Optional Defaults to curie

The name of the base model to fine-tune. You can select one of "ada", "babbage", "curie", "davinci", or a fine-tuned model created after 2022-04-21. To learn more about these models, see the **Models** documentation.

`n_epochs` integer Optional Defaults to 4

The number of epochs to train the model for. An epoch refers to one full cycle through the training dataset.

`batch_size` integer Optional Defaults to null

The batch size to use for training. The batch size is the number of training examples used to train a single forward and backward pass.

By default, the batch size will be dynamically configured to be ~0.2% of the number of examples in the training set, capped at 256 - in general, we've found that larger batch sizes tend to work better for larger datasets.

`learning_rate_multiplier` number Optional Defaults to null

The learning rate multiplier to use for training. The fine-tuning learning rate is the original learning rate used for pretraining multiplied by this value.

By default, the learning rate multiplier is the 0.05, 0.1, or 0.2 depending on final `batch_size` (larger learning rates tend to perform better with larger batch sizes). We recommend experimenting with values in the range 0.02 to 0.2 to see what produces the best results.

`prompt_loss_weight` number Optional Defaults to 0.01

The weight to use for loss on the prompt tokens. This controls how much the model tries to learn to generate the prompt (as compared to the completion which always has a weight of 1.0), and can add a stabilizing effect to training when completions are short.

If prompts are extremely long (relative to completions), it may make sense to reduce this weight so as to avoid over-prioritizing learning the prompt.

`compute_classification_metrics` boolean Optional Defaults to false

If set, we calculate classification-specific metrics such as accuracy and F-1 score using the validation set at the end of every epoch. These metrics can be viewed in the **results file**.

In order to compute classification metrics, you must provide a `validation_file`. Additionally, you must specify `classification_n_classes` for multiclass classification or `classification_positive_class` for binary classification.

`classification_n_classes` integer Optional Defaults to null

The number of classes in a classification task.

This parameter is required for multiclass classification.

`classification_positive_class` string Optional Defaults to null

The positive class in binary classification.

This parameter is needed to generate precision, recall, and F1 metrics when doing binary classification.

`classification_betas` array Optional Defaults to null

If this is provided, we calculate F-beta scores at the specified beta values. The F-beta score is a generalization of F-1 score. This is only used for binary classification.

With a beta of 1 (i.e. the F-1 score), precision and recall are given the same weight. A larger beta score puts more weight on recall and less on precision. A smaller beta score puts more weight on precision and less on recall.

`suffix` string Optional Defaults to null

A string of up to 40 characters that will be added to your fine-tuned model name.

For example, a `suffix` of "custom-model-name" would produce a model name like `ada:ft-your-org:custom-model-name-2022-02-15-04-21-04`.

Example request

python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.FineTune.create(training_file="file-XGinujblHPwGLSztz8cPS8XY")
```

Response

Copy

```json
{
  "id": "ft-AF1WoRqd3aJAHsqc9NY7iL8F",
  "object": "fine-tune",
  "model": "curie",
  "created_at": 1614807352,
  "events": [
    {
      "object": "fine-tune-event",
      "created_at": 1614807352,
      "level": "info",
      "message": "Job enqueued. Waiting for jobs ahead to complete. Queue number: 0
    }
  ],
  "fine_tuned_model": null,
  "hyperparams": {
    "batch_size": 4,
    "learning_rate_multiplier": 0.1,
    "n_epochs": 4,
    "prompt_loss_weight": 0.1,
  },
  "organization_id": "org-...",
  "result_files": [],
  "status": "pending",
  "validation_files": [],
  "training_files": [
    {
      "id": "file-XGinujblHPwGLSztz8cPS8XY",
      "object": "file",
      "bytes": 1547276,
      "created_at": 1610062281,
      "filename": "my-data-train.jsonl",
      "purpose": "fine-tune-train"
    }
  ],
```

```
  "updated_at": 1614807352,
}
```

List fine-tunes

`GET https://api.openai.com/v1/fine-tunes`

List your organization's fine-tuning jobs

Example request — python — Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.FineTune.list()
```

Response — Copy

```
{
  "object": "list",
  "data": [
    {
      "id": "ft-AF1WoRqd3aJAHsqc9NY7iL8F",
      "object": "fine-tune",
      "model": "curie",
      "created_at": 1614807352,
      "fine_tuned_model": null,
      "hyperparams": { ... },
      "organization_id": "org-...",
      "result_files": [],
      "status": "pending",
      "validation_files": [],
      "training_files": [ { ... } ],
      "updated_at": 1614807352,
    },
    { ... },
    { ... }
  ]
}
```

Retrieve fine-tune

```
GET https://api.openai.com/v1/fine-tunes/{fine_tune_id}
```

Gets info about the fine-tune job.

Learn more about Fine-tuning

Path parameters

`fine_tune_id` string Required

The ID of the fine-tune job

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.FineTune.retrieve(id="ft-AF1WoRqd3aJAHsqc9NY7iL8F")
```

Response Copy

```
{
  "id": "ft-AF1WoRqd3aJAHsqc9NY7iL8F",
  "object": "fine-tune",
  "model": "curie",
  "created_at": 1614807352,
  "events": [
    {
      "object": "fine-tune-event",
      "created_at": 1614807352,
      "level": "info",
      "message": "Job enqueued. Waiting for jobs ahead to complete. Queue number: 0
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807356,
      "level": "info",
      "message": "Job started."
    },
    {
```

```
      "object": "fine-tune-event",
      "created_at": 1614807861,
      "level": "info",
      "message": "Uploaded snapshot: curie:ft-acmeco-2021-03-03-21-44-20."
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807864,
      "level": "info",
      "message": "Uploaded result files: file-QQm6ZpqdNwAaVC3aSz5sWwLT."
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807864,
      "level": "info",
      "message": "Job succeeded."
    }
  ],
  "fine_tuned_model": "curie:ft-acmeco-2021-03-03-21-44-20",
  "hyperparams": {
    "batch_size": 4,
    "learning_rate_multiplier": 0.1,
    "n_epochs": 4,
    "prompt_loss_weight": 0.1,
  },
  "organization_id": "org-...",
  "result_files": [
    {
      "id": "file-QQm6ZpqdNwAaVC3aSz5sWwLT",
      "object": "file",
      "bytes": 81509,
      "created_at": 1614807863,
      "filename": "compiled_results.csv",
      "purpose": "fine-tune-results"
    }
  ],
  "status": "succeeded",
  "validation_files": [],
  "training_files": [
    {
      "id": "file-XGinujblHPwGLSztz8cPS8XY",
      "object": "file",
      "bytes": 1547276,
      "created_at": 1610062281,
      "filename": "my-data-train.jsonl",
      "purpose": "fine-tune-train"
    }
  ],
```

```
  "updated_at": 1614807865,
}
```

Cancel fine-tune

`POST https://api.openai.com/v1/fine-tunes/{fine_tune_id}/cancel`

Immediately cancel a fine-tune job.

Path parameters

`fine_tune_id` string Required

The ID of the fine-tune job to cancel

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.FineTune.cancel(id="ft-AF1WoRqd3aJAHsqc9NY7iL8F")
```

Response Copy

```
{
  "id": "ft-xhrpBbvVUzYGo8oUO1FY4nI7",
  "object": "fine-tune",
  "model": "curie",
  "created_at": 1614807770,
  "events": [ { ... } ],
  "fine_tuned_model": null,
  "hyperparams": { ... },
  "organization_id": "org-...",
  "result_files": [],
  "status": "cancelled",
  "validation_files": [],
  "training_files": [
    {
      "id": "file-XGinujblHPwGLSztz8cPS8XY",
      "object": "file",
      "bytes": 1547276,
```

```
        "created_at": 1610062281,
        "filename": "my-data-train.jsonl",
        "purpose": "fine-tune-train"
      }
    ],
    "updated_at": 1614807789,
}
```

List fine-tune events

`GET https://api.openai.com/v1/fine-tunes/{fine_tune_id}/events`

Get fine-grained status updates for a fine-tune job.

Path parameters

`fine_tune_id` string Required

The ID of the fine-tune job to get events for.

Query parameters

`stream` boolean Optional Defaults to false

Whether to stream events for the fine-tune job. If set to true, events will be sent as data-only **server-sent events** as they become available. The stream will terminate with a `data: [DONE]` message when the job is finished (succeeded, cancelled, or failed).

If set to false, only events generated so far will be returned.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.FineTune.list_events(id="ft-AF1WoRqd3aJAHsqc9NY7iL8F")
```

Response Copy

```
{
  "object": "list",
  "data": [
    {
      "object": "fine-tune-event",
      "created_at": 1614807352,
      "level": "info",
      "message": "Job enqueued. Waiting for jobs ahead to complete. Queue number: 0
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807356,
      "level": "info",
      "message": "Job started."
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807861,
      "level": "info",
      "message": "Uploaded snapshot: curie:ft-acmeco-2021-03-03-21-44-20."
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807864,
      "level": "info",
      "message": "Uploaded result files: file-QQm6ZpqdNwAaVC3aSz5sWwLT."
    },
    {
      "object": "fine-tune-event",
      "created_at": 1614807864,
      "level": "info",
      "message": "Job succeeded."
    }
  ]
}
```


Delete fine-tune model

```
DELETE https://api.openai.com/v1/models/{model}
```

Delete a fine-tuned model. You must have the Owner role in your organization.

Path parameters

`model` string Required

The model to delete

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Model.delete("curie:ft-acmeco-2021-03-03-21-44-20")
```

Response Copy

```
{
  "id": "curie:ft-acmeco-2021-03-03-21-44-20",
  "object": "model",
  "deleted": true
}
```

Moderations

Given a input text, outputs if the model classifies it as violating OpenAI's content policy.

Related guide: **Moderations**

Create moderation

`POST https://api.openai.com/v1/moderations`

Classifies if text violates OpenAI's Content Policy

Request body

`input` string or array Required

The input text to classify

`model` string Optional Defaults to text-moderation-latest

Two content moderations models are available: `text-moderation-stable` and `text-moderation-latest`.

The default is `text-moderation-latest` which will be automatically upgraded over time. This ensures you are always using our most accurate model. If you use `text-moderation-stable`, we will provide advanced notice before updating the model. Accuracy of `text-moderation-stable` may be slightly lower than for `text-moderation-latest`.

Example request python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Moderation.create(
  input="I want to kill them.",
)
```

Parameters Copy

```
{
  "input": "I want to kill them."
}
```

Response Copy

```
{
  "id": "modr-XXXXX",
  "model": "text-moderation-005",
  "results": [
    {
      "flagged": true,
      "categories": {
        "sexual": false,
        "hate": false,
        "harassment": false,
        "self-harm": false,
        "sexual/minors": false,
        "hate/threatening": false,
        "violence/graphic": false,
        "self-harm/intent": false,
        "self-harm/instructions": false,
```

```
      "harassment/threatening": true,
      "violence": true,
    },
    "category_scores": {
      "sexual": 1.2282071e-06,
      "hate": 0.010696256,
      "harassment": 0.29842457,
      "self-harm": 1.5236925e-08,
      "sexual/minors": 5.7246268e-08,
      "hate/threatening": 0.0060676364,
      "violence/graphic": 4.435014e-06,
      "self-harm/intent": 8.098441e-10,
      "self-harm/instructions": 2.8498655e-11,
      "harassment/threatening": 0.63055265,
      "violence": 0.99011886,
    }
  }
 ]
}
```

Edits

Given a prompt and an instruction, the model will return an edited version of the prompt.

Create edit **Deprecated**

`POST https://api.openai.com/v1/edits`

Creates a new edit for the provided input, instruction, and parameters.

Request body

`model` string Required

ID of the model to use. You can use the `text-davinci-edit-001` or `code-davinci-edit-001` model with this endpoint.

`input` string Optional Defaults to ''

The input text to use as a starting point for the edit.

`instruction` string Required

The instruction that tells the model how to edit the prompt.

`n` integer Optional Defaults to 1

How many edits to generate for the input and instruction.

`temperature` number Optional Defaults to 1

What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

We generally recommend altering this or `top_p` but not both.

`top_p` number Optional Defaults to 1

An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

We generally recommend altering this or `temperature` but not both.

Example request text-davinci-edit-001 python Copy

```python
import os
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.Edit.create(
  model="text-davinci-edit-001",
  input="What day of the wek is it?",
  instruction="Fix the spelling mistakes"
)
```

Parameters text-davinci-edit-001 Copy

```json
{
  "model": "text-davinci-edit-001",
  "input": "What day of the wek is it?",
  "instruction": "Fix the spelling mistakes"
}
```

Response Copy

```
{
  "object": "edit",
  "created": 1589478378,
  "choices": [
    {
      "text": "What day of the week is it?",
      "index": 0,
    }
  ],
  "usage": {
    "prompt_tokens": 25,
    "completion_tokens": 32,
    "total_tokens": 57
  }
}
```

Parameter details

Frequency and presence penalties

The frequency and presence penalties found in the **Completions API** can be used to reduce the likelihood of sampling repetitive sequences of tokens. They work by directly modifying the logits (un-normalized log-probabilities) with an additive contribution.

```
mu[j] -> mu[j] - c[j] * alpha_frequency - float(c[j] > 0) * alpha_pre
```

Where:

- `mu[j]` is the logits of the j-th token
- `c[j]` is how often that token was sampled prior to the current position
- `float(c[j] > 0)` is 1 if `c[j] > 0` and 0 otherwise