

# Rate limits

## Overview

### What are rate limits?

A rate limit is a restriction that an API imposes on the number of times a user or client can access the server within a specified period of time.


### Why do we have rate limits?

Rate limits are a common practice for APIs, and they're put in place for a few different reasons:

**They help protect against abuse or misuse of the API.** For example, a malicious actor could flood the API with requests in an attempt to overload it or cause disruptions in service. By setting rate limits, OpenAI can prevent this kind of activity.

**Rate limits help ensure that everyone has fair access to the API.** If one person or organization makes an excessive number of requests, it could bog down the API for everyone else. By throttling the number of requests that a single user can make, OpenAI ensures that the most number of people have an opportunity to use the API without experiencing slowdowns.

**Rate limits can help OpenAI manage the aggregate load on its infrastructure.** If requests to the API increase dramatically, it could tax the servers and cause performance issues. By setting rate limits, OpenAI can help maintain a smooth and consistent experience for all users.

 Please work through this document in its entirety to better understand how OpenAI's rate limit system works. We include code examples and possible solutions to handle common issues. It is recommended to follow this guidance before filling out the [Rate Limit Increase Request form](#) with details regarding how to fill it out in the last section.

### What are the rate limits for our API?

You can view the rate limits for your organization under the [rate limits](#) section of the account management page. Rate limits are automatically adjusted based on history of good use.

We enforce rate limits at the [organization level](#), not user level, based on the specific endpoint used as well as the type of account you have. Rate limits are measured in three ways: **RPM** (requests per minute), **RPD** (requests per day), and **TPM** (tokens per minute). Rate limit can be hit by any of the three options depending on what occurs first. For example, you might send 20 requests with only 100 tokens to the Completions endpoint and that would fill your limit (if your RPM was 20), even if you did not send 150k tokens (if your TPM limit was 150k) within those 20 requests.

## GPT-4 rate limits

During the [rollout of GPT-4](#), the model will have more aggressive rate limits to keep up with demand. You can [view your current rate limits](#) in the rate limits section of the account page. We are unable to accommodate requests for rate limit increases due to capacity constraints. We are prioritizing general access to GPT-4 first and will subsequently raise rate limits automatically as capacity allows.

## How do rate limits work?

If your rate limit is 60 requests per minute and 150k tokens per minute, you'll be limited either by reaching the requests/min cap or running out of tokens—whichever happens first. For example, if your max requests/min is 60, you should be able to send 1 request per second. If you send 1 request every 800ms, once you hit your rate limit, you'd only need to make your program sleep 200ms in order to send one more request otherwise subsequent requests would fail. With the default of 3,000 requests/min, customers can effectively send 1 request every 20ms, or every .02 seconds.

## Rate limits in headers

In addition to seeing your rate limit on your [account page](#), you can also view important information about your rate limits such as the remaining requests, tokens, and other metadata in the headers of the HTTP response.

You can expect to see the following header fields:

FIELD	SAMPLE VALUE	DESCRIPTION
x-ratelimit-limit-requests	60	The maximum number of requests that are permitted before exhausting the rate limit.
x-ratelimit-limit-tokens	150000	The maximum number of tokens that are permitted before exhausting the rate limit.
x-ratelimit-remaining-requests	59	The remaining number of requests that are permitted before exhausting the rate limit.
x-ratelimit-remaining-tokens	149984	The remaining number of tokens that are permitted before exhausting the rate limit.
x-ratelimit-reset-requests	1s	The time until the rate limit (based on requests) resets to its initial state.
x-ratelimit-reset-tokens	6m0s	The time until the rate limit (based on tokens) resets to its initial state.

## What happens if I hit a rate limit error?

Rate limit errors look like this:

```
Rate limit reached for default-text-davinci-002 in organization org-{id} on requests per min. Limit: 20.000000 / min. Current: 24.000000 / min.
```

If you hit a rate limit, it means you've made too many requests in a short period of time, and the API is refusing to fulfill further requests until a specified amount of time has passed.

## Rate limits vs max\_tokens

Each [model we offer](#) has a limited number of tokens that can be passed in as input when making a request. You cannot increase the maximum number of tokens a model takes in. For example, if you are using `text-ada-001`, the maximum number of tokens you can send to this model is 2,048 tokens per request.

## Error Mitigation

## What are some steps I can take to mitigate this?

The OpenAI Cookbook has a [Python notebook](#) that explains how to avoid rate limit errors, as well an example [Python script](#) for staying under rate limits while batch processing API requests.

You should also exercise caution when providing programmatic access, bulk processing features, and automated social media posting - consider only enabling these for trusted customers.

To protect against automated and high-volume misuse, set a usage limit for individual users within a specified time frame (daily, weekly, or monthly). Consider implementing a hard cap or a manual review process for users who exceed the limit.

## Retrying with exponential backoff

One easy way to avoid rate limit errors is to automatically retry requests with a random exponential backoff. Retrying with exponential backoff means performing a short sleep when a rate limit error is hit, then retrying the unsuccessful request. If the request is still unsuccessful, the sleep length is increased and the process is repeated. This continues until the request is successful or until a maximum number of retries is reached. This approach has many benefits:

- Automatic retries means you can recover from rate limit errors without crashes or missing data

- Exponential backoff means that your first retries can be tried quickly, while still benefiting from longer delays if your first few retries fail

- Adding random jitter to the delay helps retries from all hitting at the same time.

Note that unsuccessful requests contribute to your per-minute limit, so continuously resending a request won't work.

Below are a few example solutions **for Python** that use exponential backoff.

### > Example #1: Using the Tenacity library

---

### > Example #2: Using the backoff library

---

## ➤ Example 3: Manual backoff implementation

---

### Reduce the `max_tokens` to match the size of your completions

Your rate limit is calculated as the maximum of `max_tokens` and the estimated number of tokens based on the character count of your request. Try to set the `max_tokens` value as close to your expected response size as possible.

### Batching requests

The OpenAI API has separate limits for requests per minute and tokens per minute.

If you're hitting the limit on requests per minute, but have available capacity on tokens per minute, you can increase your throughput by batching multiple tasks into each request. This will allow you to process more tokens per minute, especially with our smaller models.

Sending in a batch of prompts works exactly the same as a normal API call, except you pass in a list of strings to the `prompt` parameter instead of a single string.

## ➤ Example without batching

---

## ➤ Example with batching

---

**i** Warning: the response object may not return completions in the order of the prompts, so always remember to match responses back to prompts using the `index` field.

## Request Increase

### When should I consider applying for a rate limit increase?

Our default rate limits help us maximize stability and prevent abuse of our API. We increase limits to enable high-traffic applications, so the best time to apply for a rate limit

increase is when you feel that you have the necessary traffic data to support a strong case for increasing the rate limit. Large rate limit increase requests without supporting data are not likely to be approved. If you're gearing up for a product launch, please obtain the relevant data through a phased release over 10 days.

Keep in mind that rate limit increases can sometimes take 7-10 days so it makes sense to try and plan ahead and submit early if there is data to support you will reach your rate limit given your current growth numbers.

## **Will my rate limit increase request be rejected?**

A rate limit increase request is most often rejected because it lacks the data needed to justify the increase. We have provided numerical examples below that show how to best support a rate limit increase request and try our best to approve all requests that align with our safety policy and show supporting data. We are committed to enabling developers to scale and be successful with our API.

## **I've implemented exponential backoff for my text/code APIs, but I'm still hitting this error. How do I increase my rate limit?**

We understand the frustration that limited rate limits can cause, and we would love to raise the defaults for everyone. However, due to shared capacity constraints, we can only approve rate limit increases for paid customers who have demonstrated a need through our [Rate Limit Increase Request form](#). To help us evaluate your needs properly, we ask that you please provide statistics on your current usage or projections based on historic user activity in the 'Share evidence of need' section of the form. If this information is not available, we recommend a phased release approach. Start by releasing the service to a subset of users at your current rate limits, gather usage data for 10 business days, and then submit a formal rate limit increase request based on that data for our review and approval.

We will review your request and if it is approved, we will notify you of the approval within a period of 7-10 business days.

Here are some examples of how you might fill out this form:

### **> DALL-E API examples**

---

## > Language model examples