Что такое ProxyAPI

ProxyAPI — это универсальное решение для доступа к API ведущих сервисов в области искусственного интеллекта, таких как OpenAI, Anthropic и Gemini. Сервис выступает в роли посредника: запросы отправляются на наш сервер, который перенаправляет их через цепочку прокси-серверов в Европе. Мы получаем ответ от целевого сервиса и возвращаем его вам.

Для работы с ProxyAPI не требуется создавать аккаунты в сторонних системах — управление и оплата происходит через ваш личный кабинет ProxyAPI, а запросы нужно отправлять на наши серверы.

Для использования ProxyAPI необходимо получить ключ. Для этого необходимо зарегистрироваться на сайте и перейти в раздел Ключи API.

Обратите внимание, что увидеть ключ целиком можно только один раз, сразу после создания.

Для любых запросов к ProxyAPI универсальным способом авторизации является передача ключа в заголовке:

```
Authorization: Bearer <КЛЮЧ>
```

Однако, в целях поддержки официальных библиотек мы также принимаем другие формы авторизации, например заголовок `x-api-key` для Anthropic или параметр `key` строки запроса для Gemini. В любом случае, использовать нужно именно наш ключ.

Для того, чтобы использовать сервис ProxyAPI, запросы необходимо отправлять по адресу `https://api.proxyapi.ru`

В зависимости от провайдера, к которому идет обращение, после домена также необходимо указывать соответствующий идентификатор, например: `https://api.proxyapi.ru/openai/v1`. Об этом можно найти более подробную информацию в соответствующем разделе документации.

При прямом обращении к провайдеру ключ ProxyAPI работать не будет.

Если задать современным языковым моделям вопрос вроде «Какая ты модель?», нередко можно получить неожиданный и даже неверный ответ. Например, GPT-5 может уверенно заявить, что она — GPT-4o или просто GPT-4. Это классический пример галлюцинации — ситуации, когда модель генерирует информацию, не соответствующую действительности.

Причина кроется в особенностях обучения. Языковые модели обучаются на больших объемах текстовых данных, собранных до их выпуска. На момент обучения GPT-5, самой модели GPT-5 ещё не существовало — в тренировочных данных присутствовала только информация о предыдущих версиях.

Когда пользователь спрашивает модель о её версии, она не «знает» свою фактическую идентичность. Вместо этого модель опирается на вероятностные связи в обученных данных, выдавая ответ, который, по её «мнению», звучит правдоподобно. В результате она может называть более раннюю версию, известную ей из обучения.

Разработчики API это прекрасно понимают. Именно поэтому любой серьёзный провайдер в ответе API всегда указывает имя модели, которая использовалась для генерации. Этот параметр является единственным надёжным источником истины о том, какая модель фактически выполнила запрос.

Если требуется точно определить модель, нужно ориентироваться исключительно на поле, которое как правило называется `model`, в API-ответе, а не на текстовый ответ самой модели на вопрос «Кто ты?».

OpenAI-совместимый API

Введение

В ProxyAPI поддерживаются оригинальные API самых популярных LLM провайдеров, таких как OpenAI, Anthropic и Gemini. Это сделано для того, чтобы гарантировать доступ ко всем инструментам и особому функционалу, предоставляемому провайдерами искусственного интеллекта.

Однако и у универсального API есть свои преимущества, главным из которых является легкость переключения между моделями разных провайдеров без необходимости использовать разные SDK. В связи с этим, наряду с поддержкой оригинальных API, ProxyAPI также поддерживает универсальный API на основе спецификации OpenAI.

OpenAI-совместимый API — это прокси-шлюз, принимающий запросы в формате OpenAI API и автоматически перенаправляющий их к различным LLM-провайдерам. На входе и выходе вы всегда работаете с унифицированным протоколом OpenAI, а конвертация форматов и маршрутизация выполняются на нашей стороне.

1. **Клиент** отправляет запрос к одному из эндпоинтов `/v1/*` так же, как если бы он обращался к OpenAI API.
2. **Роутер** определяет целевого провайдера из поля `model` (например, `anthropic/claude-sonnet-4-20250514`).
3. **Адаптер** переводит тело запроса в нативный формат выбранного провайдера.
4. **Ответ** от LLM приводится к стандартному формату OpenAI и отправляется клиенту.

- **Единый SDK.** Можно использовать официальный OpenAI SDK или любую совместимую библиотеку без изменений кода.
- **Гибкая маршрутизация.** Легко переключайтесь между моделями разных вендоров одной строкой.

Базовый адрес для OpenAI-совместимого API:

```
https://openai.api.proxyapi.ru/v1
```

HTTP	Путь	Назначение
GET	/v1/models	Список моделей и их метаданные
POST	/v1/chat/completions	Генерация текста
POST	/v1/embeddings	Векторные представления текста
POST	/v1/images/generations	Генерация изображений
POST	/v1/images/edits	Редактирование изображений
POST	/v1/audio/transcriptions	Расшифровка аудио (Speech-to-Text)
POST	/v1/audio/speech	Генерация речи (Text-to-Speech)
POST	/v1/responses	Новый API на замену /v1/chat/completions

Все маршруты полностью совместимы с официальной OpenAI спецификацией, включая форматы запросов, ответов и ошибок.

Имя модели имеет вид <провайдер>/<название-модели>. В случае с OpenRouter необходимо указывать openrouter/<провайдер>/<название-модели>

Провайдер	Пример идентификатора
OpenAI	openai/gpt-4o
Anthropic	anthropic/claude-sonnet-4-20250514
Gemini	gemini/gemini-2.0-flash
OpenRouter	openrouter/deepseek/deepseek-chat-v3.1

Список поддерживаемых моделей можно посмотреть в разделах **Модели** для каждого провайдера в данной документации.

```
curlpythonnode.js
from openai import OpenAI

client = OpenAI(
    api_key="<КЛЮЧ>",
    base_url="https://openai.api.proxyapi.ru/v1",
)

chat_completion = client.chat.completions.create(
    model="anthropic/claude-sonnet-4-20250514",
    messages=[
        {
            "role": "user",
            "content": "Привет!"
        }
    ]
)
```

Модели OpenAI

Мы обеспечиваем поддержку всех актуальных моделей OpenAI и оперативно интегрируем новые. OpenAI является одним из лидеров в области искусственного интеллекта, предлагая передовые модели, такие как GPT, DALL-E и Whisper.

При обращении к моделям можно не указывать точную версию (snapshot), достаточно указать имя всей серии. Запрос автоматически будет направлен на самую актуальную и надежную версию. Например, при обращении к модели gpt-4o запрос автоматически будет направлен

к модели `gpt-4o-2024-08-06` — самой актуальной на время написания этой статьи. Версии моделей, которые на данный момент являются алиасом по умолчанию для всей серии выделены **жирным** в списке ниже.

Модель	Контекстное окно	Кэш	Ввод	Вывод	Лимиты (RPM / TPM)*
gpt-4o-search-preview >> gpt-4o-search-preview-2025-03-11	128 000	Нет	Текст/ изображение	Текст	1 тыс / 3 млн
gpt-4o-mini-search-preview >> gpt-4o-mini-search-preview-2025-03-11	128 000	Нет	Текст/ изображение	Текст	30 тыс / 150 млн

Модель	Контекстное окно	Кэш	Ввод	Вывод	Лимиты (RPM / TPM)*
gpt-5.2 >> gpt-5.1-2025-11-13	400 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5.2-chat-latest >> gpt-5.2-chat-latest	128 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5.1 >> gpt-5.1-2025-11-13	400 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5.1-chat-latest >> gpt-5.1-chat-latest	128 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5.1-codex-max >> gpt-5.1-codex-max	400 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5.1-codex >> gpt-5.1-codex	400 000	Да	Текст/ изображение	Текст	15 тыс / 10 млн
gpt-5.1-codex-mini >> gpt-5.1-codex-mini	400 000	Да	Текст/ изображение	Текст	30 тыс / 180 млн
gpt-5 >> gpt-5-2025-08-07	400 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5-chat-latest >> gpt-5-chat-latest	128 000	Да	Текст/ изображение	Текст	15 тыс / 40 млн
gpt-5-mini >> gpt-5-mini-2025-08-07	400 000	Да	Текст/ изображение	Текст	30 тыс / 180 млн
gpt-5-nano >> gpt-5-nano-2025-08-07	400 000	Да	Текст/ изображение	Текст	30 тыс / 180 млн
gpt-5-codex >> gpt-5-codex	400 000	Да	Текст/ изображение	Текст	15 тыс / 10 млн
gpt-4.1 >> gpt-4.1-2025-04-14	1 000 000	Да	Текст/ изображение	Текст	10 тыс / 30 млн
gpt-4.1-mini >> gpt-4.1-mini-2025-04-14	1 000 000	Да	Текст/ изображение	Текст	30 тыс / 150 млн
gpt-4.1-nano >> gpt-4.1-nano-2025-04-14	1 000 000	Да	Текст/ изображение	Текст	30 тыс / 150 млн

Модель	Контекстное окно	Кэш	Ввод	Вывод	Лимиты (RPM / TPM)*
o4-mini >> **o4-mini-2025-04-16**	200 000	Да	Текст/ изображение	Текст	30 тыс / 150 млн
o3-pro >> **o3-pro-2025-06-10**	200 000	Нет	Текст/ изображение	Текст	10 тыс / 30 млн
o3 >> **o3-2025-04-16**	200 000	Да	Текст/ изображение	Текст	10 тыс / 30 млн
o3-mini >> **o3-mini-2025-01-31**	200 000	Да	Текст	Текст	30 тыс / 150 млн
o1-pro >> **o1-pro-2025-03-19**	200 000	Нет	Текст/ изображение	Текст	10 тыс / 30 млн
o1 >> **o1-2024-12-17**	200 000	Да	Текст/ изображение	Текст	10 тыс / 30 млн
codex-mini-latest >> **codex-mini-latest**	200 000	Да	Текст/ изображение	Текст	30 тыс / 150 млн

Большинство моделей доступно только в Responses API.

Модели `gpt-oss-20b` и `gpt-oss-120b` доступны через OpenRouter.

Модель	Контекстное окно	Кэш	Ввод	Вывод	Лимиты (RPM / TPM)*
computer-use-preview >> **computer-use-preview-2025-03-11**	8 192	Нет	Текст/ изображение	Текст	3 тыс / 20 млн

Модель	Контекстное окно	Ввод	Вывод	Лимиты (RPM / TPM)*
gpt-4o >> **gpt-4o-2024-08-06** Другие версии: gpt-4o-2024-11-20 gpt-4o-2024-05-13	128 000	Да	Текст/ изображение	Текст
gpt-4o-mini >> **gpt-4o-mini-2024-07-18**	128 000	Да	Текст/ изображение	Текст
gpt-4o-audio-preview >> **gpt-4o-audio-preview-2025-06-03** Другие версии: gpt-4o-audio-preview-2024-12-17 gpt-4o-audio-preview-2024-10-01	128 000	Нет	Текст/аудио	Текст/аудио
gpt-4o-mini-audio-preview >> **gpt-4o-mini-audio-preview-2024-12-17**	128 000	Нет	Текст/аудио	Текст/аудио
gpt-4-turbo	128 000	Текст/	Текст	10 тыс / 2 млн

Модель	Контекстное окно	Ввод	Вывод	Лимиты (RPM / TPM)*
>> **gpt-4-turbo-2024-04-09**		изображение		
gpt-3.5-turbo >> **gpt-3.5-turbo-0125** Другие версии: gpt-3.5-turbo-1106	16 385	Текст	Текст	10 тыс / 50 млн

Модель	Лимиты (RPM)*
gpt-image-1.5	250
gpt-image-1	250
gpt-image-1-mini	250
dall-e-3	10 тыс
dall-e-2	10 тыс

Модель	Лимиты (RPM)*
sora-2	375
sora-2-pro	150

Модель	Лимиты (RPM)*
gpt-4o-transcribe	10 тыс
gpt-4o-transcribe-diarize	10 тыс
gpt-4o-mini-transcribe	10 тыс
whisper-1	10 тыс

Модель	Лимиты (RPM)*
gpt-4o-mini-tts	10 тыс
tts-1	10 тыс
tts-1-hd	10 тыс

Преобразование текста в векторный формат.

Модель	Лимиты (RPM / TPM)*
text-embedding-3-small	10 тыс / 10 млн
text-embedding-3-large	10 тыс / 10 млн
text-embedding-ada-002	10 тыс / 10 млн

---

* RPM - requests per minute - запросов в минуту / TPM - tokens per minute - токенов в минуту

Поддержка API от OpenAI

Responses API

11 марта 2025 года компания OpenAI представила Responses API — новый инструмент для разработчиков, объединяющий простоту Chat Completions API с возможностями Assistants API. Он включает встроенные функции, такие как веб-поиск для получения актуальной информации из

интернета, работа с файлами и инструмент Computer Use, позволяющий агентам выполнять задачи на компьютере, имитируя действия пользователя. OpenAI планирует заменить Assistants API и Chat Completions API на Responses API к середине 2026 года, учитывая отзывы разработчиков и стремясь улучшить функциональность.

ProxyAPI, в свою очередь, начинает внедрять поддержку Responses API поэтапно. На данный момент поддерживаются все виды запросов и инструментов, кроме работы с файлами и векторными хранилищами. Как и в случае с Assistants API, этот функционал со временем станет доступен в подписке ProxyAPI Pro.

Генерация текста OpenAI API

Все методы и форматы запросов и ответов в ProxyAPI идентичны оригинальным от OpenAI. Таким образом, официальные SDK полностью совместимы с ProxyAPI.

Для запросов к OpenAI в качестве пути к API необходимо использовать:

https://api.proxyapi.ru/openai/v1

Пример

Пример простого запроса генерации текста.

Responses API

python

```
from openai import OpenAI

client = OpenAI(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/openai/v1",
)

response = client.responses.create(
    model="gpt-4o",
```

```
    input="Привет!"
)
```

Chat Completions API

curl

python

node.js

```
from openai import OpenAI

client = OpenAI(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/openai/v1",
)

chat_completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {
            "role": "user",
            "content": "Привет!"
        }
    ]
)
```

Генерация текста OpenAI API с веб-поиском

Web search — это инструмент OpenAI API, который позволяет моделям искусственного интеллекта выполнять поиск в интернете для получения актуальной информации перед генерацией ответа.

Объем поискового контекста

Параметр search_context_size определяет объем контекста, получаемого из веб-поиска, для формирования ответа модели. Он регулирует количество токенов, извлекаемых из веб-страниц, что влияет на полноту и детализацию ответа. Изменение этого параметра позволяет

контролировать, насколько обширным будет контекст, используемый моделью при генерации ответа.

Возможные значения:

low — малый объем контекста

medium — средний объем контекста (по умолчанию)

high — большой объем контекста

Местоположение пользователя

Параметр user_location позволяет задать местоположение пользователя для веб-поиска. Он используется для улучшения релевантности результатов поиска и предоставления более точных и актуальных ответов.

Параметры:

type — тип местоположения (всегда approximate)

country — двухбуквенный ISO-код страны (например, RU для России)

city — город

region — регион

Тарификация

Потребление токенов тарифицируется как обычно для выбранной модели. Каждый поиск считается отдельно, стоимость фиксированная и зависит от выбранного объема контекста. Цены здесь.

Responses API

В новой Responses API поиск доступен в качестве инструмента, который можно включать или не включать в параметры запроса. Таким образом, можно контролировать, для каких запросов поиск будет осуществлен, а для каких — нет.

Пример:

python

```
from openai import OpenAI

client = OpenAI(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/openai/v1",
)

response = client.responses.create(
    model="gpt-4o",
    tools=[{
        "type": "web_search",
        "search_context_size": "low",
        "user_location": {
            "type": "approximate",
            "country": "RU",
            "city": "Moscow",
            "region": "Moscow"
        }
    }],
    input="Какая сегодня погода?"
)
```

Chat Completions API

В Chat Completions API поиск доступен через обращение к специальным расширениям основных моделей: gpt-4o-search-preview и gpt-4o-mini-search-preview. При запросе к этим моделям, поиск выполняется обязательно, то есть нельзя включать или выключать его для каких-то определенных запросов.

Пример:

curl

python

node.js

```
from openai import OpenAI

client = OpenAI(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/openai/v1",
)

chat_completion = client.chat.completions.create(
    model="gpt-4o-search-preview",
    web_search_options={
        "search_context_size": "low",
        "user_location": {
            "type": "approximate",
            "approximate": {
                "country": "RU",
                "city": "Moscow",
                "region": "Moscow"
            }
        }
    },
    messages=[
        {
            "role": "user",
            "content": "Какая сегодня погода?"
        }
    ]
)
```

Модели Anthropic

Anthropic — это один из мировых лидеров в сфере искусственного интеллекта. Компания известна своими передовыми языковыми моделями семейства Claude, конкурирующими с

ChatGPT от OpenAI и Gemini от Google. Разработчики особенно высоко оценивают модель Claude, особенно её последнюю версию Claude 3.7 Sonnet, за её выдающиеся способности в программировании. Многие отмечают, что она превосходит другие модели ИИ в генерации кода и решении сложных задач.

Модель	Контекстное окно	Кэш	Ввод	Вывод	Лимиты (RPM / TPM)*
claude-opus-4-5-20251101	200 000	Да	Текст/ изображение	Текст	4 тыс / 2 млн
claude-opus-4-1-20250805	200 000	Да	Текст/ изображение	Текст	4 тыс / 2 млн
claude-opus-4-20250514	200 000	Да	Текст/ изображение	Текст	4 тыс / 2 млн
claude-sonnet-4-5 >> **claude-sonnet-4-5-20250929**	1 000 000	Да	Текст/ изображение	Текст	4 тыс / 2 млн
claude-sonnet-4-20250514	1 000 000	Да	Текст/ изображение	Текст	4 тыс / 2 млн
claude-3-7-sonnet-20250219	200 000	Да	Текст/ изображение	Текст	4 тыс / 280 тыс
claude-haiku-4-5 >> **claude-haiku-4-5-20251001**	200 000	Да	Текст/ изображение	Текст	4 тыс / 4 млн
claude-3-5-haiku-20241022	200 000	Да	Текст/ изображение	Текст	4 тыс / 480 тыс

---

* RPM - requests per minute - запросов в минуту / TPM - tokens per minute - токенов в минуту

Генерация текста Anthropic API

Все методы и форматы запросов и ответов, а также методы авторизации в ProxyAPI идентичны оригинальным от Anthropic. Таким образом, официальные SDK полностью совместимы с ProxyAPI.

Для запросов к Anthropic в качестве пути к API необходимо использовать:

```
https://api.proxyapi.ru/anthropic
```

Пример простого запроса генерации текста:

```
python
import anthropic

client = anthropic.Anthropic(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/anthropic",
)

message = client.messages.create(
    model="claude-3-7-sonnet-20250219",
    max_tokens=1000,
    messages=[
```

```
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "Привет!"
                }
            ]
        }
    ]
)
```

Генерация текста Anthropic API с веб-поиском

Инструмент веб-поиска предоставляет Claude прямой доступ к контенту в реальном времени, позволяя ему отвечать на вопросы с актуальной информацией, выходящей за рамки его базы знаний. Claude автоматически указывает источники из результатов поиска в своих ответах.

Когда вы добавляете инструмент веб-поиска в свой API-запрос:

- Claude сам решает, когда выполнять поиск, исходя из запроса.
- API выполняет поисковые запросы и предоставляет Claude полученные результаты. Этот процесс может повторяться несколько раз в рамках одного запроса.
- В конце своей сессии Claude возвращает финальный ответ с указанием использованных источников.

```
python
cimport anthropic

client = anthropic.Anthropic(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/anthropic",
)

response = client.messages.create(
    model="claude-3-7-sonnet-20250219",
    max_tokens=1024,
    messages=[
        {
            "role": "user",
            "content": "Сколько людей живет в Москве?"
        }
    ],
    tools=[{
        "type": "web_search_20250305",
        "name": "web_search",
        "max_uses": 5
    }]
)
print(response)
```

Инструмент веб-поиска поддерживает следующие параметры:

```
{
  "type": "web_search_20250305",
  "name": "web_search",

  // Необязательное поле: максимальное количество поисковых запросов
  "max_uses": 5,

  // Необязательное поле: включить результаты только для этих доменов
  "allowed_domains": ["example.com", "trusteddomain.org"],

  // Необязательное поле: исключить результаты для этих доменов
  "blocked_domains": ["untrustedsource.com"],

  // Необязательное поле: локализованные результаты поиска
  "user_location": {
    "type": "approximate",
    "city": "San Francisco",
    "region": "California",
    "country": "US",
    "timezone": "America/Los_Angeles"
  }
}
```

Параметр `max_uses` ограничивает количество выполняемых поисков. Если Claude попытается выполнить больше поисков, чем разрешено, `web_search_tool_result` вернёт ошибку с кодом `max_uses_exceeded`.

При использовании фильтров по доменам:

- Не указывайте схему HTTP/HTTPS (используйте `example.com` вместо `https://example.com`)
- Поддомены включаются автоматически (`example.com` охватывает `docs.example.com`)
- Поддерживаются подпути (`example.com/blog`)
- Вы можете использовать либо `allowed_domains`, либо `blocked_domains`, но не оба параметра в одном запросе.

Параметр `user_location` позволяет локализовать результаты поиска в зависимости от местоположения пользователя.

- `type`: Тип местоположения (должен быть `approximate`)
- `city`: Название города
- `region`: Регион или штат
- `country`: Страна
- `timezone`: IANA-идентификатор часового пояса

Веб-поиск работает с кэшированием. Чтобы включить кэширование, добавьте хотя бы одну точку останова `cache_control` в ваш запрос. Система автоматически закэширует всё до последнего блока `web_search_tool_result` при выполнении инструмента.

Для диалогов с несколькими итерациями установите точку останова `cache_control` внутри или после последнего

блока `web_search_tool_result`, чтобы повторно использовать закэшированный контент.

Например, чтобы использовать кэширование с веб-поиском в диалоге с несколькими итерациями:

```
import anthropic

client = anthropic.Anthropic(
    api_key="<КЛЮЧ>",
    base_url="https://api.proxyapi.ru/anthropic",
)

# Первый запрос с веб-поиском и точкой останова кэширования
messages = [
    {
        "role": "user",
        "content": "Какая сегодня погода?"
    }
]

response1 = client.messages.create(
    model="claude-3-7-sonnet-20250219",
    max_tokens=1024,
    messages=messages,
    tools=[{
        "type": "web_search_20250305",
        "name": "web_search",
        "user_location": {
            "type": "approximate",
            "city": "Moscow",
            "region": "Moscow",
            "country": "RU",
            "timezone": "Europe/Moscow"
        }
    }]
)

# Добавляем ответ Claude в диалог
messages.append({
    "role": "assistant",
    "content": response1.content
})

# Второй запрос с точкой останова кэширования после результатов поиска
messages.append({
    "role": "user",
    "content": "Будет ли дождь на неделе?",
    "cache_control": {"type": "ephemeral"}  # Кэшировать до этой точки
})

response2 = client.messages.create(
    model="claude-3-7-sonnet-20250219",
    max_tokens=1024,
    messages=messages,
    tools=[{
        "type": "web_search_20250305",
        "name": "web_search",
        "user_location": {
            "type": "approximate",
            "city": "Moscow",
            "region": "Moscow",
            "country": "RU",
```

```
            "timezone": "Europe/Moscow"
        }
    }]
)
# Второй ответ будет использовать кэшированные результаты поиска
# и все еще сможет выполнять новые поиски, если это необходимо
print(f"Cache read tokens: {response2.usage.get('cache_read_input_tokens', 0)}")
```