

Freecode 接入指南 · 合订版

更新日期：2026-10-13。所有示例密钥均为占位符，请使用自己的 Freecode API 密钥。

目录

- [接入概览](#)
- [快速开始](#)
- [CC Switch](#)
- [Claude Code](#)
- [Codex CLI](#)
- [OpenAI SDK](#)
- [Anthropic SDK](#)
- [分组链使用指南](#)
- [常见问题](#)

Freecode 接入指南

更新日期：2026-10-13

Freecode 提供统一的 AI 模型接口。你可以通过命令行编程工具、配置切换工具，或 Python / JavaScript SDK 接入。

先准备这三项

- 1. 登录 [Freecode](#)，在 [API 密钥](#) 页面创建自己的密钥。
- 2. 在 [模型广场](#) 查看目标模型、可用分组和价格，为密钥选择包含该模型的分组。
- 3. 在 [钱包](#) 确认余额，同时检查密钥额度、有效期和模型限制。

界面默认使用繁体中文：API 密钥显示为「API 金鑰」，模型广场显示为「模型廣場」。本文示例中的 `sk-your-freecode-api-key` 都是占位符，使用时替换成自己的完整密钥。

地址速查

用途	填写内容
网站 / 控制台入口	<code>https://api.freecode.cc</code>
OpenAI SDK、Codex 的 API Base URL	<code>https://api.freecode.cc/v1</code>
Claude Code、Anthropic SDK 的 Base URL	<code>https://api.freecode.cc</code>
模型发现接口	GET <code>https://api.freecode.cc/v1/models</code>
OpenAI Responses 请求	POST <code>https://api.freecode.cc/v1/responses</code>
OpenAI Chat Completions 请求	POST <code>https://api.freecode.cc/v1/chat/completions</code>
Anthropic Messages 请求	POST <code>https://api.freecode.cc/v1/messages</code>

客户端配置中的 Base URL 不要填写完整的 `/responses`、`/messages` 或 `/chat/completions` 地址，客户端会自行追加接口路径。

按使用方式阅读

你想做什么	文档
创建密钥，发出第一个请求	01 · 快速开始
从 Freecode 一键导入 CC Switch	03 · CC Switch

你想做什么	文档
配置 Claude Code	04 · Claude Code
配置 Codex CLI	05 · Codex CLI
在 Python / JavaScript 中调用 OpenAI 格式接口	06 · OpenAI SDK
在 Python / JavaScript 中调用 Claude 格式接口	07 · Anthropic SDK
给多个密钥设置分组优先顺序和备用分组	分组链使用指南
遇到鉴权、分组、地址或模型问题	常见问题

示例模型

本文使用 `gpt-6.1-sol`、`gpt-4o`、`claude-sonnet-4-6`、`claude-opus-4-6`、`claude-haiku-4-5`。这些名称已在整理时的线上模型广场核对。模型能否调用，还取决于你所选密钥的分组、模型限制和对应接口；配置时复制完整模型 ID，保留大小写和后缀。

分组与分组链

首次接入建议先选择一个包含目标模型的分组，跑通后再按需要配置分组链。

在 API 密钥页打开「分組鏈」，新建链并选择分组，按希望尝试的顺序排列、保存；再编辑目标 API 密钥，在分组选择中选中这条链。客户端仍使用同一把密钥和相同 Base URL，不需要把分组名写进请求。

分组链复用你设定的分组顺序进行路由与重试，不保证所有类型的错误都会切换。请在使用日志中核对实际使用的模型、分组和费用。

完整步骤、切换规则及修改和删除的影响，见 [分组链使用指南](#)。

使用与分享

每个工具使用独立密钥，方便查看用量、撤销或调整额度。「无限额度」仅表示不额外设置密钥额度上限，调用仍使用账户余额并正常计费。

本目录提供分篇 Markdown 和合订版；PDF 可直接发给接入方。不要在文档、截图或公开仓库中填入真实密钥。遇到问题时只提供时间、模型、工具版本、HTTP 状态和 request ID。

01 · Freecode 快速开始

目标：创建自己的 API 密钥，并通过 Freecode 发出一个实际请求。

1. 登录与准备余额

打开 <https://api.freecode.cc> 注册并登录；如果注册入口要求邀请码，按页面提示获取。进入 [钱包](#)，确认账户有可用余额。

2. 创建 API 密钥

打开 [API 密钥](#)，点击创建按钮。

- 名称：例如 my-first-app。
- 分组：在 [模型广场](#) 查找 gpt-6.1-sol，选择包含该模型、允许你使用的分组。
- 额度与有效期：按实际需要设置。
- 模型限制：如启用限制，把准备调用的模型加入允许列表。

保存后复制完整密钥，包括 sk- 前缀。密钥选中的分组决定路由，不能只凭模型在广场中出现就判断这把密钥一定可用。

3. 设置环境变量

以下命令在 macOS / Linux / WSL 的 Bash 或 Zsh 中运行：

```
export FREECODE_API_KEY="sk-your-freecode-api-key"
export FREECODE_MODEL="gpt-6.1-sol"
```

Windows PowerShell：

```
$env:FREECODE_API_KEY = "sk-your-freecode-api-key"
$env:FREECODE_MODEL = "gpt-6.1-sol"
```

这些设置只对当前终端及其启动的程序生效。示例密钥需要替换，模型名可换成该密钥有权使用的模型。

4. 查看该密钥的模型目录

macOS / Linux / WSL：

```
curl -sS https://api.freecode.cc/v1/models \
```

```
-H "Authorization: Bearer ${FREECODE_API_KEY}"
```

PowerShell 使用 `curl.exe`，避免旧版本 PowerShell 的命令别名影响：

```
curl.exe -sS "https://api.freecode.cc/v1/models" `
-H "Authorization: Bearer $env:FREECODE_API_KEY"
```

这个接口用于发现模型；返回模型列表不代表一次实际生成已经成功。

5. 发出第一个 Responses 请求

macOS / Linux / WSL：

```
curl -sS --max-time 120 https://api.freecode.cc/v1/responses `
-H "Authorization: Bearer ${FREECODE_API_KEY}" `
-H "Content-Type: application/json" `
--data '{"model":"'${FREECODE_MODEL}'","input":"'请用一句话打个招呼`
',"store":false}'
```

Windows PowerShell：

```
$body = @{
    model = $env:FREECODE_MODEL
    input = "请用一句话打个招呼"
    store = $false
} | ConvertTo-Json

Invoke-RestMethod `
-Method Post `
-Uri "https://api.freecode.cc/v1/responses" `
-Headers @{ Authorization = "Bearer $env:FREECODE_API_KEY" } `
-ContentType "application/json; charset=utf-8" `
-Body ([System.Text.Encoding]::UTF8.GetBytes($body))
```

成功时返回 Responses 对象，回答通常在 `output` 数组的文本内容中。随后打开 [使用日志](#)，核对本次模型、实际分组、用量和费用。

6. 可选：使用 Chat Completions

如果所选模型及分组支持 Chat Completions，可使用下面的示例。这里使用模型广场中可见的 `gpt-4o`，请为密钥选择对应分组。

```
curl -sS --max-time 120 https://api.freecode.cc/v1/chat/completions \
-H "Authorization: Bearer ${FREECODE_API_KEY}" \
-H "Content-Type: application/json" \
--data '{
  "model": "gpt-4o",
  "messages": [{"role": "user", "content": "请用一句话打个招呼"}]
}'
```

站内 [Playground](#) 可以验证当前账号的对话能力；要验证某把 API 密钥的分组和限制，应执行上面的实际请求。

接下来按工具选择 [CC Switch](#)、[Claude Code](#)、[Codex CLI](#) 或 [SDK](#)。报错时查看 [常见问题](#)。

参考：[原快速开始指南](#)。本文地址、模型和菜单按 [Freecode](#) 整理。

03 · 使用 CC Switch 接入 Freecode

CC Switch 用于管理编程工具的供应商配置。Freecode 的 API 密钥页提供 CC Switch 导入入口，目前可选择 Claude、Codex 和 Gemini。本文以 GPT 和 Claude 接入为例。

1. 安装

从 [CC Switch 官方发布页](#) 下载当前稳定版，安装并启动一次，使系统注册 `ccswitch://` 协议。

2. 从 Freecode 导入

- 1. 登录 [Freecode](#)。
 - 2. 打开 [API 密钥](#)，为目标工具创建独立密钥，并选择包含目标模型的分组。
 - 3. 在该密钥所在行打开操作菜单，选择「CC Switch」。
 - 4. 选择 Claude 或 Codex，填写供应商名称，例如 Freecode Claude 或 Freecode Codex。
 - 5. 选择主模型；Claude 可进一步选择 Haiku、Sonnet 和 Opus 模型。
 - 6. 点击「Open CC Switch / 開啟 CC Switch」，允许浏览器打开应用。
 - 7. 在 CC Switch 的确认界面核对名称、地址、密钥和模型，然后完成导入并确认当前启用的供应商。
- 推荐示例：Codex 选择 `gpt-6.1-sol`；Claude 主模型选择 `claude-sonnet-4-6`。模型选择框可能列出账号可见模型，使用前仍需确认当前这把密钥的分组有权调用。

3. 核对地址

Freecode 导入器会按工具设置地址：

应用	地址	主要接口
Codex	<code>https://api.freecode.cc/v1</code>	Responses
Claude Code	<code>https://api.freecode.cc</code>	Anthropic Messages

如果需要手动添加供应商，也使用上述地址、自己的 Freecode 密钥和对应模型。不要在 Base URL 中填入完整接口路径。

Claude 的可选槽位可填写：

槽位	示例模型
主模型 / Sonnet	<code>claude-sonnet-4-6</code>
Opus	<code>claude-opus-4-6</code>

槽位	示例模型
Haiku	claude-haiku-4-5

4. 验证

完全退出并重新打开对应 CLI，开始新会话，发送一句话。收到真实回复后，在 [Freecode 使用日志](#) 核对调用记录。

Claude Code 使用 `/status` 检查配置；Codex 检查当前模型和供应商。若你同时使用 CC Switch 和手动配置文件，要确认最终生效的是哪一套，避免互相覆盖。

常见问题

- 点击后没有打开应用：先启动 CC Switch，确认浏览器允许打开外部应用。
- 导入后没有生效：在 CC Switch 确认已启用 Freecode 供应商，然后重新打开 CLI。
- 模型不可用：核对密钥分组、模型限制和精确 ID。
- 需要切回原服务：在 CC Switch 重新启用原供应商。

导入链接中包含真实密钥，不要转发链接或截图。其他错误见 [常见问题](#)。

参考：[原 CC Switch 指南](#)、[CC Switch 官方说明](#)。

04 · Claude Code 接入 Freecode

Claude Code 使用 Anthropic Messages 接口。配置 Freecode 时填写站点根地址，不加 /v1。

1. 安装

按 [Claude Code 官方安装指南](#) 安装当前版本。安装后检查：

```
claude --version
```

已有 Node.js 环境、需要使用 npm 安装方式的用户，也可选择：

```
npm install -g @anthropic-ai/claude-code
```

官方优先推荐原生安装方式，具体要求以官方指南为准。

2. 配置 settings.json

配置文件位置：

系统	路径
macOS / Linux / WSL	~/.claude/settings.json
Windows	%USERPROFILE%\\.claude\settings.json

文件存在时合并 env 中的相关项，保留原有权限、插件等其他设置。以下配置可直接作为新文件模板：

```
{
  "env": {
    "ANTHROPIC_BASE_URL": "https://api.freecode.cc",
    "ANTHROPIC_AUTH_TOKEN": "sk-your-freecode-api-key",
    "ANTHROPIC_MODEL": "claude-sonnet-4-6",
    "ANTHROPIC_DEFAULT_SONNET_MODEL": "claude-sonnet-4-6",
    "ANTHROPIC_DEFAULT_OPUS_MODEL": "claude-opus-4-6",
    "ANTHROPIC_DEFAULT_HAIKU_MODEL": "claude-haiku-4-5"
  }
}
```

替换密钥，并确认其分组可以使用配置的各个模型。主会话之外，Claude Code 也可能调用其他槽位执行后台任务，因此不要给槽位填写该密钥无权使用的模型。

ANTHROPIC_MODEL 是主模型；ANTHROPIC_DEFAULT_*_MODEL 控制相关模型别名。不要把 Base URL 改成 /v1/messages，客户端会自己拼接路径。

3. 可选：先在当前终端验证

不修改配置文件时，可以在 macOS / Linux / WSL 当前终端设置：

```
export ANTHROPIC_BASE_URL="https://api.freecode.cc"
export ANTHROPIC_AUTH_TOKEN="sk-your-freecode-api-key"
export ANTHROPIC_MODEL="claude-sonnet-4-6"
claude
```

PowerShell：

```
$env:ANTHROPIC_BASE_URL = "https://api.freecode.cc"
$env:ANTHROPIC_AUTH_TOKEN = "sk-your-freecode-api-key"
$env:ANTHROPIC_MODEL = "claude-sonnet-4-6"
claude
```

如果原来使用其他服务，检查终端中是否还有旧的 Anthropic 认证或地址变量。使用一种明确的配置来源，并在新会话中通过 /status 确认最终配置。

4. 验证

在项目目录运行 claude，输入“用一句话说明当前项目的用途”。收到正常回复后打开 [使用日志](#)，核对模型、分组和费用。

/status 用于检查配置；只有实际回复和日志记录才能证明本次模型调用成功。也可以使用 [CC Switch](#) 管理配置。

常见问题

- 401：检查完整密钥、有效期、额度，以及是否仍在旧服务的认证变量。
- 404：Base URL 应为 https://api.freecode.cc，检查是否重复了 /v1。
- 主会话可用，后台任务失败：检查 Haiku / Sonnet / Opus 槽位对应的模型权限。
- 切换后仍走旧服务：重新打开终端与 CLI，核对 CC Switch 当前供应商。

更多排查见 [常见问题](#)。

参考：[原 Claude Code 指南](#)、[官方网关配置](#)、[官方模型配置](#)。

05 · Codex CLI 接入 Freecode

Codex 使用 Responses 接口。本文用独立的 Freecode 环境变量认证，不需要把密钥写进 config.toml。

1. 安装

已有受支持的 Node.js 环境时：

```
npm install -g @openai/codex
codex --version
```

也可按 [Codex 官方 CLI 指南](#) 选择其他安装方式。

2. 配置用户级 config.toml

系统	路径
macOS / Linux / WSL	~/.codex/config.toml
Windows	%USERPROFILE%\codex\config.toml

自定义供应商放在用户级配置文件中。已有文件时合并相关项，不要覆盖 MCP、项目或其他供应商配置，也不要重复定义同名 TOML 表。

```
#:schema https://developers.openai.com/codex/config-schema.json
model_provider = "freecode"
model = "gpt-6.1-sol"
model_reasoning_effort = "high"

[model_providers.freecode]
name = "Freecode"
base_url = "https://api.freecode.cc/v1"
wire_api = "responses"
env_key = "FREECODE_API_KEY"
requires_openai_auth = false
```

Base URL 需要带 /v1，但不要加 /responses。这里使用 env_key 读取 Freecode 密钥，所以 requires_openai_auth 设为 false；不要把另一套 auth.json 认证方式混进同一个供应商配置。

3. 设置密钥

macOS / Linux / WSL，先在当前终端设置，再从这个终端启动 Codex：

```
export FREECODE_API_KEY="sk-your-freecode-api-key"
codex
```

Windows PowerShell:

```
$env:FREECODE_API_KEY = "sk-your-freecode-api-key"
codex
```

要长期使用，可把变量配置到自己的系统环境或受保护的本机配置中；新开的终端需重新读取。IDE 中使用时，确保启动 IDE 的进程也能读取该变量。

4. 验证

在项目目录启动 Codex，核对供应商为 freecode、模型为 gpt-6.1-sol，发送一句话或让它读取项目并概括结构。收到真实回复后，在 [使用日志](#) 核对记录。

推理强度由模型实际能力决定；如果调整为其他模型，请按该模型的支持范围设置。先验证基础对话，再按实际分组支持情况启用搜索、图像等附加能力。

5. 常见问题

- 提示 FREECODE_API_KEY 未设置：在启动 Codex 的同一个终端设置变量。
- 401：检查变量里是否为完整 Freecode 密钥，以及密钥是否仍有效。
- 404：确认地址是 <https://api.freecode.cc/v1>，没有重复 /v1 或多加 /responses。
- 模型不可用：检查精确 ID、密钥分组和模型允许列表。
- 仍走其他供应商：检查用户级 model_provider，以及是否有 CC Switch 或启动参数覆盖配置。

更多排查见 [常见问题](#)。

参考：[原 Codex 指南](#)、[官方高级配置](#)、[官方认证说明](#)。配置已结合实际读取的 OpenAI 官方文档核对。

06 · OpenAI SDK 接入 Freecode

使用官方 OpenAI SDK，自定义 Base URL 为 `https://api.freecode.cc/v1`。Responses 和 Chat Completions 是不同接口，应选择该模型及分组支持的格式。

1. 设置环境变量

macOS / Linux / WSL:

```
export FREECODE_API_KEY="sk-your-freecode-api-key"
export FREECODE_MODEL="gpt-6.1-sol"
```

PowerShell:

```
$env:FREECODE_API_KEY = "sk-your-freecode-api-key"
$env:FREECODE_MODEL = "gpt-6.1-sol"
```

2. Python: Responses

安装到当前 Python 环境:

```
python -m pip install --upgrade openai
```

保存为 `freecode_openai.py`:

```
import os
from openai import OpenAI

client = OpenAI(
    api_key=os.environ["FREECODE_API_KEY"],
    base_url="https://api.freecode.cc/v1",
    timeout=120.0,
)

response = client.responses.create(
    model=os.environ.get("FREECODE_MODEL", "gpt-6.1-sol"),
    input="请用一句话打个招呼",
    store=False,
)

print(response.output_text)
```

运行：

```
python freecode_openai.py
```

流式输出，将上面创建请求和打印的部分替换为：

```
stream = client.responses.create(
    model=os.environ.get("FREECODE_MODEL", "gpt-6.1-sol"),
    input="请写一首四行小诗",
    store=False,
    stream=True,
)
for event in stream:
    if event.type == "response.output_text.delta":
        print(event.delta, end="", flush=True)
print()
```

3. JavaScript: Responses

在项目目录安装：

```
npm install openai
```

保存为 freecode_openai.mjs。 .mjs 使用 ES Modules，可直接在 Node.js 中执行：

```
import OpenAI from "openai";

if (!process.env.FREECODE_API_KEY) {
    throw new Error("请先设置 FREECODE_API_KEY");
}

const client = new OpenAI({
    apiKey: process.env.FREECODE_API_KEY,
    baseURL: "https://api.freecode.cc/v1",
    timeout: 120000,
});

const response = await client.responses.create({
    model: process.env.FREECODE_MODEL ?? "gpt-6.1-sol",
    input: "请用一句话打个招呼",
    store: false,
});

console.log(response.output_text);
```

运行：

```
node freecode_openai.mjs
```

流式输出，将创建请求和打印的部分替换为：

```
const stream = await client.responses.create({
  model: process.env.FREECODE_MODEL ?? "gpt-6.1-sol",
  input: "请写一首四行小诗",
  store: false,
  stream: true,
});
for await (const event of stream) {
  if (event.type === "response.output_text.delta") {
    process.stdout.write(event.delta);
  }
}
process.stdout.write("\n");
```

4. Chat Completions

使用支持该接口的模型及分组，例如 gpt-4o。在上面的 Python client 初始化之后，可改为：

```
response = client.chat.completions.create(
  model="gpt-4o",
  messages=[{"role": "user", "content": "请用一句话打个招呼"}],
)
print(response.choices[0].message.content)
```

JavaScript 的对应写法：

```
const response = await client.chat.completions.create({
  model: "gpt-4o",
  messages: [{ role: "user", content: "请用一句话打个招呼" }],
});
console.log(response.choices[0].message.content);
```

若使用 Chat Completions 流式输出，部分分片可能没有 choices，读取前先判断，避免因空分片导致应用异常。

5. 验证与集成

收到回复后，在 [使用日志](#) 核对模型、分组和用量。LangChain 等框架若支持自定义 OpenAI Base URL，可以使用相同地址；接口和模型参数仍须按框架实际调用格式设置。

密钥仅放在服务端环境或本机环境中，不写入网页前端代码。报错见 [常见问题](#)。

参考：[原 OpenAI SDK 指南](#)、[OpenAI 官方 SDK 快速开始](#)。

07 · Anthropic SDK 接入 Freecode

使用 Anthropic Messages 协议，Base URL 填 `https://api.freecode.cc`。SDK 会自行追加 `/v1/messages`。

1. 设置环境变量

macOS / Linux / WSL:

```
export FREECODE_API_KEY="sk-your-freecode-api-key"
export FREECODE_CLAUDE_MODEL="claude-sonnet-4-6"
```

PowerShell:

```
$env:FREECODE_API_KEY = "sk-your-freecode-api-key"
$env:FREECODE_CLAUDE_MODEL = "claude-sonnet-4-6"
```

为密钥选择支持目标 Claude 模型及 SDK 调用的分组。如果某个分组限制客户端类型，应按该分组的要求选择接入工具。

2. Python

安装:

```
python -m pip install --upgrade anthropic
```

保存为 `freecode_anthropic.py`:

```
import os
import anthropic

client = anthropic.Anthropic(
    api_key=os.environ["FREECODE_API_KEY"],
    base_url="https://api.freecode.cc",
    timeout=120.0,
)

message = client.messages.create(
    model=os.environ.get("FREECODE_CLAUDE_MODEL", "claude-sonnet-4-6"),
    max_tokens=1024,
```

```
    messages=[{"role": "user", "content": "请用一句话打个招呼"}],
  )
  for block in message.content:
    if block.type == "text":
      print(block.text)
```

运行：

```
python freecode_anthropic.py
```

流式输出，将创建请求和打印的部分替换为：

```
with client.messages.stream(
    model=os.environ.get("FREECODE_CLAUDE_MODEL", "claude-sonnet-4-6"),
    max_tokens=1024,
    messages=[{"role": "user", "content": "请写一首四行小诗"}],
) as stream:
    for text in stream.text_stream:
        print(text, end="", flush=True)
print()
```

3. JavaScript

安装：

```
npm install @anthropic-ai/sdk
```

保存为 freecode_anthropic.mjs：

```
import Anthropic from "@anthropic-ai/sdk";

if (!process.env.FREECODE_API_KEY) {
  throw new Error("请先设置 FREECODE_API_KEY");
}

const client = new Anthropic({
  apiKey: process.env.FREECODE_API_KEY,
  baseUrl: "https://api.freecode.cc",
  timeout: 120000,
});

const message = await client.messages.create({
  model: process.env.FREECODE_CLAUDE_MODEL ?? "claude-sonnet-4-6",
```

```
max_tokens: 1024,
messages: [{ role: "user", content: "请用一句话打个招呼" }],
});
for (const block of message.content) {
  if (block.type === "text") console.log(block.text);
}
```

运行：

```
node freecode_anthropic.mjs
```

流式输出，将创建请求和打印的部分替换为：

```
const stream = client.messages.stream({
  model: process.env.FREECODE_CLAUDE_MODEL ?? "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{ role: "user", content: "请写一首四行小诗" }],
});
for await (const event of stream) {
  if (
    event.type === "content_block_delta" &&
    event.delta.type === "text_delta"
  ) {
    process.stdout.write(event.delta.text);
  }
}
process.stdout.write("\n");
```

4. 参数与验证

- `max_tokens` 是 Messages 请求的必填参数，示例设置为 1024。
- SDK 的 `api_key` / `apiKey` 默认生成 `x-api-key` 认证头，Freecode 的 Messages 接口支持这种方式。
- Python 使用 `base_url`；JavaScript 使用 `baseUrl`。
- 回复的 `content` 可能包含多个类型的块，示例只打印文本块。

收到回复后，在 [使用日志](#) 核对模型、分组和费用。也可按 [OpenAI SDK 指南](#) 使用兼容接口，但须确认目标模型和分组支持该格式。

问题排查见 [常见问题](#)。

参考：[原 Anthropic SDK 指南](#)、[Anthropic Python SDK](#)、[Anthropic JavaScript SDK](#)。

分组链使用指南

分组链把多个分组保存为一条有顺序的备用线路。多个 API 密钥可以绑定同一条链；以后调整这条链的分组或顺序，绑定的密钥会一起使用新配置，客户端不需要换密钥或修改地址。

本文按控制台默认的繁体中文界面说明。「分組鏈」在英语界面显示为 **Group chains**。

使用前准备

- 1. 登录 [Freecode 控制台](#)，打开 [模型广场](#)，确认目标模型在哪些分组可用。
- 2. 选择你有权限使用的分组，并查看各分组的价格。分组链不会增加分组权限，也不会把一个模型转换成另一个模型。
- 3. 在 [钱包](#) 确认余额；检查密钥有效期、额度和模型限制。

建议先用单个分组跑通请求，再添加备用分组。准备作为备用的分组也应支持同一个完整模型 ID 和你使用的接口。

第一步：创建分组链

- 1. 打开 [API 密钥](#) 页面，点击顶部「分組鏈」。
- 2. 点击「新增分組鏈」，填写容易辨认的「分組鏈名稱」，例如 日常编程备用线路。
- 3. 在「新增分組」中依次选择你要使用的分组。
- 4. 在「分組順序」列表中拖动条目，或使用上移、下移按钮调整顺序。首选分组放在最上面；删除某个条目只会把它从这条链中移除。
- 5. 点击「儲存」。列表会显示链名称、分组顺序和绑定的 API 密钥数量。

可以按下面的思路排列；表中名称是示意，请从你自己的可用分组中选择：

顺序	作用	选择建议
1	首选分组 A	平时优先使用，模型和价格符合预期
2	备用分组 B	支持同一个模型，首选无法使用时作为备用
3	兜底分组 C	同样支持目标模型，并确认其价格可接受

每个账户最多创建 **10 条链**，每条链包含 **1–20 个不重复的分组**。名称不能重复，最长 100 个字符。「自动选择分组 / auto」不能作为链中的一个成员。

第二步：把 API 密钥绑定到链

- 1. 返回 API 密钥列表，编辑你要使用的密钥；也可以创建一把新密钥。
- 2. 打开「分組」下拉框，搜索并选中刚保存的**链名称**。

3. 检查密钥的模型限制和额度，然后保存。选择分组链后，系统会启用跨分组重试。
4. 在列表的「分組」列确认显示的是链名称。

只创建链、没有编辑密钥绑定它，原来的密钥仍使用原来的分组。不同工具可以使用不同密钥，并绑定同一条链。

第三步：正常调用并确认结果

客户端继续使用自己的 Freecode API 密钥、原来的接口地址和模型 ID。无需在请求正文或 HTTP Header 中添加链名称，也无需填写分组名。

客户端	地址
OpenAI SDK / Codex	https://api.freecode.cc/v1
Claude Code / Anthropic SDK	https://api.freecode.cc

配置方法见 [Codex CLI](#)、[Claude Code](#)、[OpenAI SDK](#) 和 [Anthropic SDK](#)。

发出一次实际请求后，打开 [使用日志](#)，按密钥和时间查找记录，核对实际使用的模型、分组与费用。路由预览或模型列表可见，只说明配置符合对应条件；确认可用仍需一次成功的实际调用。

系统什么时候切换分组

分组链的正常选择顺序是从上到下，同一个请求成功后就停止，不会同时向所有分组发送请求。

- 当前分组没有目标模型的可用渠道时，会继续寻找后面的分组。
- 上游请求失败时，只有服务器判定为可重试的错误，才会继续重试。系统可能先尝试同一分组内的其他渠道；达到该分组的重试限制后，再尝试后面的分组。
- 鉴权、参数、额度等在进入上游前就被拒绝的请求，不会因绑定分组链而自动恢复。明确不可重试的错误也会直接返回。
- 重试次数、重试状态码和管理员设置会影响结果。若管理员设置了会话亲和或严格会话绑定，同一会话还可能优先复用已有渠道，或限制切换。
- 链内所有分组都必须保持对你可用。某个分组被取消权限或移除倍率配置时，整条链可能在校验阶段被拒绝；请编辑链删除这个分组。

因此，分组链不保证每种错误都能切换成功，也不会自动把请求中的模型 ID 换成其他模型。

费用如何计算

按实际使用的模型和实际路由到的分组计费。链中的倍率不相加，也不会按第一组的价格锁定后续分组。

如果备用组更贵，切换后的成功请求可能产生更高费用。请先在模型广场比较各分组价格，再确定顺序，并在使用日志中确认扣费。分组链不会绕过账户余额或密钥额度限制。

修改与删除

****修改链：****在「分組鏈」弹窗中点击目标链的编辑按钮，修改名称、分组或顺序并保存。绑定这条链的所有密钥都会使用新配置，之后发出的请求按新配置处理；不需要逐个重新绑定。

****从链中删除一个分组：****只改变链的备用顺序，不会删除平台分组、渠道或 API 密钥。

****删除整条链：****会保留原来的 API 密钥，并将它们改为「自动选择分组」，清空原来的自定义分组顺序，同时启用跨分组重试。自动选择使用平台提供的可用分组顺序，可能与被删除的链不同。

如果你希望删除后仍固定使用某个分组，请先编辑相关密钥，改绑到指定分组或另一条链，再删除旧链。链列表中的密钥数量可帮助确认影响范围。

常见问题

现象	检查方法
创建了链，但调用仍走原来的分组	编辑密钥，在「分組」中选中链名称并保存
下拉框没有想要的分组	确认账户拥有该分组权限；管理员还需配置分组倍率
提示「分组链包含不可用分组」	编辑链，移除被撤销权限或已下线配置的分组
返回没有可用渠道	检查目标模型 ID、接口类型及链中是否有启用的对应渠道
失败后没有切到下一组	根据 HTTP 状态和 request ID 排查；并非每个错误都允许重试
费用与首选组不同	查看使用日志的实际分组和该组模型价格

其他接入问题见 [常见问题](#)。

Freecode 接入常见问题

先确认实际请求地址、密钥分组和精确模型 ID，再检查额度、网络 and 工具配置。

按错误排查

现象	检查内容
401 / Unauthorized	是否使用完整 Freecode 密钥；是否被禁用、过期或删除；客户端是否读到了预期环境变量
403 / 无权访问分组	密钥绑定的分组、账号可用分组，以及分组对客户端类型的要求
No available channel / 模型不可用	当前分组是否包含模型；模型白名单是否放行；对应协议是否可用
404 / Not Found	是否重复 /v1；是否把完整接口路径误填进 Base URL
429	请求速率、并发限制或上游限流；降低并发，按服务返回要求重试
余额或额度不足	钱包余额、密钥剩余额度和有效期；无限额度密钥也会消耗账户余额
超时 / stream error	DNS、终端代理、网络稳定性和请求超时；保留报错中的 request ID
某个工具可用，另一个不可用	两个工具是否使用相同地址、密钥分组、模型及接口；客户端限制是否一致

地址怎么填

工具	Base URL
OpenAI SDK / Codex	https://api.freecode.cc/v1
Anthropic SDK / Claude Code	https://api.freecode.cc

错误示例：https://api.freecode.cc/v1/v1、把 /v1/responses 作为 Codex Base URL，或把 /v1/messages 作为 Anthropic SDK Base URL。

模型目录有结果，为什么请求仍失败

/v1/models 是模型发现接口；实际生成还要经过额度、模型限制、路由和上游处理。打开 [模型广场](#) 核对精确模型 ID，再用这把密钥执行一次最小请求。

Playground 使用当前账号的站内身份，不能替代对具体 API 密钥的验证。

分组怎么选

首次使用先选一个明确包含目标模型的分组。如果需要备用顺序，在 [API 密钥页](#) 创建分组链，再编辑密钥绑定该链。客户端无需额外填写分组参数。

分组链中某个分组没有目标模型时，请按后台可用模型和路由结果检查；不要自行猜测分组名或在请求中追加未知分组参数。

切换配置后仍走旧服务

1. 检查 CC Switch 当前启用的站点。
2. 检查终端中的旧 Base URL、认证变量和启动参数。
3. 检查用户级配置文件，避免同一个供应商重复定义。
4. 完全退出 CLI，重新打开终端并开始新会话。
5. 在 [使用日志](#) 验证实际使用的模型和分组。

需要协助时提供什么

提供工具名称与版本、请求时间和时区、模型 ID、Base URL、HTTP 状态、错误消息及 request ID。不要提供完整 API Key、Authorization、Cookie 或包含真实密钥的导入链接。

如果密钥已泄露，在 API 密钥页停用或删除该密钥，创建新的密钥并更新客户端。