# 小艾引擎 API 开发文档

**版本**: v1.0 **更新日期**: 2026-02-26 **基础地址**: `https://api1.szaiai.com`

---

## 1. 概述

小艾引擎 API 提供强大的大语言模型能力，支持对话补全、流式输出和文本向量化。接口兼容主流 SDK，开发者可快速集成。

### 接入信息

| 项目 | 值 |
| --- | --- |
| 基础地址 | `https://api1.szaiai.com` |
| 认证方式 | API Key（由管理员分配） |
| 请求格式 | JSON |
| 字符编码 | UTF-8 |

---

## 2. 认证

所有请求必须携带 API Key，支持两种方式：

**方式一：X-Api-Key 请求头**

```
X-Api-Key: sk-gw-your-key-here
```

**方式二：Bearer Token（兼容 OpenAI SDK）**

```
Authorization: Bearer sk-gw-your-key-here
```

---

## 3. API 接口

### 3.1 对话补全（Messages）

创建一轮对话，模型根据消息上下文生成回复。

**请求**

```
POST /v1/messages
Content-Type: application/json
```

**请求参数**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| model | string | 是 | 模型名称，填写管理员提供的值（如 `xiaoai-chat-v1`） |
| messages | array | 是 | 对话消息列表 |

| | | | |
|---|---|---|---|
| max_tokens | integer | 是 | 最大生成 token 数（建议 1024-4096） |
| stream | boolean | 否 | 是否启用流式输出，默认 false |
| temperature | number | 否 | 采样温度 0-1，默认 1。越低越确定 |
| top_p | number | 否 | 核采样概率 0-1 |
| stop_sequences | array | 否 | 停止序列列表 |
| system | string | 否 | 系统提示词，设定模型角色和行为 |

**messages 格式**

```
[
  {"role": "user", "content": "你好，请介绍一下你自己"},
  {"role": "assistant", "content": "你好！我是小艾引擎..."},
  {"role": "user", "content": "你能做什么？"}
]
```

支持的 role：

- `user` — 用户消息
- `assistant` — 助手消息（用于多轮对话上下文）

**响应示例**

```
{
  "model": "xiaoai-chat-v1",
  "id": "msg-a1b2c3d4e5f6...",
  "type": "message",
  "role": "assistant",
  "content": [
    {
      "type": "text",
      "text": "你好！我是小艾引擎，一个强大的AI助手..."
    }
  ],
  "stop_reason": "end_turn",
  "usage": {
    "input_tokens": 15,
    "output_tokens": 42
  }
}
```

**响应字段说明**

| 字段 | 说明 |
|---|---|
| model | 模型名称 |
| id | 请求唯一标识 |

| | |
|---|---|
| content | 回复内容数组，每项包含 type 和 text |
| stop_reason | 停止原因：end_turn（正常结束）、max_tokens（达到上限）、stop_sequence |
| usage.input_tokens | 输入消耗的 token 数 |
| usage.output_tokens | 输出消耗的 token 数 |

---

### 3.2 对话补全（Chat Completions，OpenAI 兼容）

兼容 OpenAI Chat Completions 格式，方便从 OpenAI SDK 迁移。

**请求**

```
POST /v1/chat/completions
Content-Type: application/json
```

**请求参数**

| 参数 | 类型 | 必填 | 说明 |
|---|---|---|---|
| model | string | 是 | 模型名称 |
| messages | array | 是 | 对话消息列表 |
| max_tokens | integer | 否 | 最大生成 token 数 |
| stream | boolean | 否 | 是否流式输出 |
| temperature | number | 否 | 采样温度 0-2 |

**响应示例**

```
{
  "id": "cmpl-a1b2c3d4e5f6...",
  "object": "chat.completion",
  "model": "xiaoai-chat-v1",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "你好！有什么可以帮助你的吗？"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 10,
    "completion_tokens": 15,
    "total_tokens": 25
  }
}
```

### 3.3 文本向量化（Embeddings）

将文本转换为向量表示，用于语义搜索、聚类等场景。

**请求**

```
POST /v1/embeddings
Content-Type: application/json
```

**请求参数**

| 参数 | 类型 | 必填 | 说明 |
| --- | --- | --- | --- |
| model | string | 是 | 模型名称 |
| input | string 或 array | 是 | 待向量化的文本（或文本数组） |
| encoding_format | string | 否 | 返回格式：float（默认）或 base64 |

**响应示例**

```
{
  "object": "list",
  "model": "xiaoai-embed-v1",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [0.0023, -0.0091, 0.0152, ...]
    }
  ],
  "usage": {
    "prompt_tokens": 8,
    "total_tokens": 8
  }
}
```

---

## 4. 流式输出（Streaming）

流式模式下，模型逐步返回生成内容，适合实时展示场景。设置 `stream: true` 即可启用。

响应格式为 Server-Sent Events (SSE)，每行以 `data:` 开头：

```
event: message_start
data: {"type":"message_start","message":{"model":"xiaoai-chat-v1",...}}

event: content_block_delta
data: {"type":"content_block_delta","delta":{"type":"text_delta","text":"你好"}}

event: content_block_delta
data: {"type":"content_block_delta","delta":{"type":"text_delta","text":"! "}}
```

```
event: message_stop
data: {"type":"message_stop"}
```

---

## 5. 错误处理

**错误响应格式**

```
{
  "error": {
    "type": "error_type",
    "message": "错误详细描述"
  }
}
```

**常见错误码**

| HTTP 状态码 | 错误类型 | 说明 |
| --- | --- | --- |
| 400 | invalid_request_error | 请求参数错误（如 JSON 格式无效） |
| 401 | authentication_error | API Key 无效或缺失 |
| 403 | permission_error | 权限不足（如 Key 已禁用或过期） |
| 429 | rate_limit_error | 请求频率超限，请降低调用频率 |
| 502 | upstream_error | 服务暂时不可用，请稍后重试 |

**建议的重试策略**：遇到 429 或 502 错误时，采用指数退避（1s, 2s, 4s...）重试，最多重试 3 次。

---

## 6. 代码示例

### 6.1 Python — 对话补全

```
import anthropic

client = anthropic.Anthropic(
    api_key="sk-gw-your-key-here",
    base_url="https://api1.szaiai.com"
)

response = client.messages.create(
    model="xiaoai-chat-v1",
    max_tokens=1024,
    messages=[
        {"role": "user", "content": "请用三句话介绍人工智能"}
    ]
)
```

```
print(response.content[0].text)
```

### 6.2 Python — 流式输出

```
import anthropic

client = anthropic.Anthropic(
    api_key="sk-gw-your-key-here",
    base_url="https://api1.szaiai.com"
)

with client.messages.stream(
    model="xiaoai-chat-v1",
    max_tokens=1024,
    messages=[{"role": "user", "content": "写一首关于春天的诗"}]
) as stream:
    for text in stream.text_stream:
        print(text, end="", flush=True)
```

### 6.3 Python — OpenAI 兼容模式

```
from openai import OpenAI

client = OpenAI(
    api_key="sk-gw-your-key-here",
    base_url="https://api1.szaiai.com/v1"
)

response = client.chat.completions.create(
    model="xiaoai-chat-v1",
    messages=[
        {"role": "user", "content": "什么是机器学习？"}
    ]
)

print(response.choices[0].message.content)
```

### 6.4 Node.js

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

const client = new Anthropic({
  apiKey: "sk-gw-your-key-here",
  baseURL: "https://api1.szaiai.com",
});

const response = await client.messages.create({
```

```
  model: "xiaoai-chat-v1",
  max_tokens: 1024,
  messages: [{ role: "user", content: "你好" }],
});

console.log(response.content[0].text);
```

### 6.5 cURL

```
curl https://api1.szaiai.com/v1/messages \
  -H "Content-Type: application/json" \
  -H "X-Api-Key: sk-gw-your-key-here" \
  -d '{
    "model": "xiaoai-chat-v1",
    "max_tokens": 1024,
    "messages": [
      {"role": "user", "content": "你好"}
    ]
  }'
```

### 6.6 文本向量化

```
from openai import OpenAI

client = OpenAI(
    api_key="sk-gw-your-key-here",
    base_url="https://api1.szaiai.com/v1"
)

response = client.embeddings.create(
    model="xiaoai-embed-v1",
    input="人工智能是计算机科学的一个分支"
)

print(f"向量维度: {len(response.data[0].embedding)}")
```

---

## 7. 多轮对话

实现多轮对话需要将历史消息传入 messages 数组：

```
messages = [
    {"role": "user", "content": "我叫小明"},
    {"role": "assistant", "content": "你好小明！很高兴认识你。"},
    {"role": "user", "content": "我叫什么名字？"}
]

response = client.messages.create(
    model="xiaoai-chat-v1",
```

```
    max_tokens=256,
    messages=messages
)
# 模型会回答"你叫小明"
```

---

## 8. 系统提示词

通过 system 参数设定模型的角色和行为：

```
response = client.messages.create(
    model="xiaoai-chat-v1",
    max_tokens=1024,
    system="你是一位专业的法律顾问，用通俗易懂的语言解答法律问题。",
    messages=[
        {"role": "user", "content": "租房合同到期后房东不退押金怎么办？"}
    ]
)
```

---

## 9. 使用限制

| 限制项 | 默认值 | 说明 |
| --- | --- | --- |
| 请求频率 | 60 次/分钟 | 可按需调整，联系管理员 |
| 单次请求体大小 | 50 MB | 支持大文档、图片 base64 |
| max_tokens 上限 | 取决于模型 | 通常 4096-8192 |

---

## 10. SDK 安装

小艾引擎 API 兼容主流 SDK，选择以下任一方式安装：

**Python — Messages 接口**

```
pip install anthropic
```

**Python — Chat Completions / Embeddings 接口**

```
pip install openai
```

**Node.js — Messages 接口**

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

**Node.js — Chat Completions / Embeddings 接口**

```
npm install openai
```

---

## 11. 常见问题

**Q: model 参数填什么？** A: 填写管理员分配给您的模型名称。对话接口填 `xiaoai-chat-v1`，向量化接口填 `xiaoai-embed-v1`。具体以管理员提供为准。

**Q: 支持图片输入吗？** A: 支持。在 content 中使用 image 类型的内容块，传入 base64 编码的图片即可。

**Q: 流式输出和普通模式有什么区别？** A: 流式模式下模型边生成边返回，首个 token 到达更快，适合聊天界面实时显示。普通模式等全部生成完毕后一次性返回。

**Q: 如何计算费用？** A: 按 input_tokens + output_tokens 计费，具体单价请联系管理员。

---

> ***技术支持**: 如有问题请联系管理员获取帮助。*