# 保利威AI答疑助手API接口文档

## 目录



## 接口概览

| 接口名称 | 接口地址 | 请求方式 | 说明 |
| --- | --- | --- | --- |
| 获取Token | `https://api.polyv.net/live/v3/common/token/get-ai-token` | POST | 获取AI答疑助手的访问令牌 |
| AI聊天问答 | `https://api.polyv.net/ai/v1/chat/question` | GET | 向AI助手发送问题并流式获取回答 |

## 调用流程

1. **获取Token**
   - 使用应用信息(appId、secretKey)调用获取Token接口
   - 生成签名并提交请求
   - 获取返回的token
2. **使用Token进行AI聊天**
   - 使用上一步获取的token调用AI聊天接口

- 处理流式返回的数据
- 拼接返回内容获得完整回答

# 接口详情

## 获取Token接口

### 基本信息

- **接口地址**：`https://api.polyv.net/live/v3/common/token/get-ai-token`
- **请求方式**：POST
- **数据格式**：form-data

### 请求参数

| 参数名 | 类型 | 必填 | 描述 |
| --- | --- | --- | --- |
| appId | String | 是 | 客户在保利威的应用ID |
| timestamp | Long | 是 | 当前时间戳(毫秒) |
| viewerId | String | 是 | 观看者/用户唯一标识 |
| sign | String | 是 | 请求签名 |

### 返回参数

| 参数名 | 类型 | 描述 |
| --- | --- | --- |
| code | Integer | 状态码，200表示成功 |
| message | String | 响应消息 |
| status | String | 响应状态 |
| data.token | String | AI答疑助手的访问令牌 |
| data.userId | String | 用户ID |
| data.validTime | Integer | 令牌有效时间(秒) |

### 返回示例

```
{
  "code": 200,
  "message": "success",
  "status": "success",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "userId": "viewer123456",
    "validTime": 3600
  }
}
```

## AI聊天问答接口

### 基本信息

- **接口地址**：`https://api.polyv.net/ai/v1/chat/question`
- **请求方式**：GET
- **返回格式**：EventStream(流式返回)

### 请求参数

| 参数名 | 类型 | 必填 | 描述 |
| --- | --- | --- | --- |
| question | String | 是 | 用户的问题内容 |
| aiId | String | 是 | AI助手的ID |
| token | String | 是 | 通过获取Token接口获得的访问令牌 |

### 返回格式

接口使用EventStream(Server-Sent Events)格式返回数据，需要客户端支持处理此格式：

```
event: message
data: {"content":"我可以回答您关于", "done":false}
```

### 返回数据说明

| 事件类型 | 数据格式 | 描述 |
| --- | --- | --- |
| message | JSON | AI生成的部分回答内容 |
| done | JSON | 表示回答已经结束 |

### 数据字段说明

| 字段名 | 类型 | 描述 |
| --- | --- | --- |
| content | String | AI生成的回答内容 |
| done | Boolean | AI生成的回答内容是否完成 |

## 签名生成规则

生成sign参数的步骤：

1. 将除sign以外的全部参数按照参数名ASCII码从小到大排序
2. 将排序后的参数以key=value形式拼接，并用&连接
3. 在拼接后的字符串首尾加上密钥(secretKey)
4. 对拼接后的字符串进行MD5加密并转为大写

签名公式：

```
MD5(secretKey + 排序并拼接后的参数字符串 + secretKey).toUpperCase()
```

# 代码示例

## Python完整调用流程

```
import time
import hashlib
import json
import requests
import sseclient  # 需要安装: pip install sseclient-py

POLYV_CONFIG = {
    "app_id": '保利威平台appId',
    "app_secret": '保利威平台appSecret'
}

# 第一步：获取Token
def get_ai_token(viewer_id):
 """获取聊天Token的接口"""

    # 构建请求参数
    if viewerId is None:
        print("viewerId is None")
        viewerId = f"polyvWebscriptViewerId-{uuid.uuid4()}"

    print("最后的viewerId", viewerId)
    form_data = {
        "appId": POLYV_CONFIG["app_id"],
        "timestamp": int(time.time() * 1000),
        "viewerId": viewerId
    }

    # 添加签名
    form_data["sign"] = create_api_sign(POLYV_CONFIG["app_secret"], form_data)

    # 发送请求到保利威API
    response = requests.post(
        "https://api.polyv.net/live/v3/common/token/get-ai-token",
        data=form_data
    )

    print(response.json())
    token = response.json().get('data')
    return token
# 第二步：使用token调用AI聊天接口
def chat_with_ai(token, ai_id, question):
    """向AI助手发送问题并获取回答"""
    # 构建请求URL
    url = f"https://api.polyv.net/ai/v1/chat/question?question={question}&aiId={ai_id}&token={token}"

    # 发送请求
    headers = {
        "Accept": "text/event-stream",
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/134.0.0.0 Safari/537.36"
    }

    response = requests.get(url, headers=headers)
    print(response)
    client = sseclient.SSEClient(response)

    # 处理EventStream响应
    answer = ""
    try:
        for event in client.events():
            data = json.loads(event.data)
            print(data)
            content = data.get("content", "")
            done = data.get("done")
            if not done:
                print(f'过程数据：===>{content}')
                answer += content
                # 处理回答片段
            else:
                answer += content
                print("回答完成")
                break
        return answer
    except Exception as e:
        print(f"处理响应出错：{e}")
        raise

# 使用示例
if __name__ == "__main__":
    # 配置参数
    APP_ID = "你的appId"
    SECRET_KEY = "你的secretKey"
    VIEWER_ID = "用户唯一标识"
    AI_ID = "1159"
    QUESTION = "你好，请介绍一下自己"

    try:
        # 注意：实际应用中token获取应在后端进行
        token = get_ai_token(APP_ID, SECRET_KEY, VIEWER_ID)
        print(f"获取token成功：{token}")

        # 使用token进行聊天
        answer = chat_with_ai(token, AI_ID, QUESTION)
        print(f"完整回答：{answer}")
    except Exception as e:
        print(f"错误：{e}")
```

# 常见问题

1. **Token安全问题**
    - 获取Token的接口必须在服务器端调用，避免将secretKey暴露在前端

- 可以在自己的后端服务中封装一个获取Token的接口供前端调用

2. **Token有效期**
   - Token有效期通常为3600秒(1小时)
   - 建议在Token过期前自动更新，可以设置定时任务在过期前5-10分钟更新Token
3. **EventStream处理**
   - 不同前端框架处理EventStream的方式可能不同
   - 需要正确拼接所有message事件中的content字段以获取完整回答
   - 当收到type为"done"的事件时，表示本次回答已经结束
4. **错误处理**
   - 建议添加超时处理机制，避免网络问题导致请求一直挂起
   - 添加错误重试机制，在网络波动时能够自动重新连接
5. **请求限制**
   - 接口调用频率限制：单个appId每秒最多500次请求
   - 确保合理控制请求频率，避免触发限流