**扣子websocketapi**

除了基于 WebRTC 的实时通信方案外，扣子智能语音还推出了 WebSocket OpenAPI 方案，用于实现用户与智能体之间的实时语音通话。该方案通过 WebSocket 协议提供高效、灵活的语音交互能力，适用于多种应用场景。

**注意事项**

WebSocket OpenAPI 支持的音频编码格式如下：

* 上行：输入音频支持 PCM、G711A、G711U，上传语音文件支持 WAV 或 OGG 格式，默认格式为 WAV。
* 下行：输出音频支持 PCM 或 OPUS 格式，默认为采样率 24000 的 PCM 片段。

**准备工作**

在开始集成 WebSocket OpenAPI 之前，你需要先完成以下准备工作。

|  |  |
| --- | --- |
| **操作** | **说明** |
| 发布智能体 | 已成功搭建并发布智能体为 API 服务。搭建步骤请参见[搭建一个 AI 助手智能体](https://www.coze.cn/open/docs/guides/agent_quick_start)，发布步骤请参见[发布智能体为 API 服务](https://www.coze.cn/open/docs/guides/publish_agent_api)。 |
| 获取访问密钥 | 获取访问密钥，用于身份认证与鉴权。   * **体验或调试场景**：建议生成短期的个人访问令牌（PAT），以快速完成 Realtime SDK 的整体流程。个人访问令牌的获取方法请参见[添加个人访问令牌](https://www.coze.cn/open/docs/developer_guides/pat)。 |

**实现语音通话**

**步骤一 ：建立 WebSocket 连接**

发起 HTTP 请求时，在请求头（Header）中添加Authorization信息。Authorization的取值固定为Bearer *$Access\_Token*，用于扣子 OpenAPI 鉴权的访问密钥。将您在准备工作中获取的访问密钥替换掉 $Access\_Token 后再发起请求。

|  |  |
| --- | --- |
| **操作** | **说明** |
| 发布智能体 | 已成功搭建并发布智能体为 API 服务。搭建步骤请参见[搭建一个 AI 助手智能体](https://www.coze.cn/open/docs/guides/agent_quick_start)，发布步骤请参见[发布智能体为 API 服务](https://www.coze.cn/open/docs/guides/publish_agent_api)。 |
| 获取访问密钥 | 获取访问密钥，用于身份认证与鉴权。   * **体验或调试场景**：建议生成短期的个人访问令牌（PAT），以快速完成 Realtime SDK 的整体流程。个人访问令牌的获取方法请参见[添加个人访问令牌](https://www.coze.cn/open/docs/developer_guides/pat)。 * **线上环境**：在线上环境中，应使用 OAuth 鉴权方案。OAuth 鉴权方案的详细说明请参见[OAuth 应用管理](https://www.coze.cn/open/docs/developer_guides/oauth_apps)。 |

|  |
| --- |
| javascript import WebSocket from 'ws';    const url = `wss://ws.coze.cn/v1/audio/transcriptions?authorization=Bearer pat\_OYDacMzM3WyOWV3Dtj2bHRMymzxP\*\*\*\*`;  # 如果是流式语音对话接口，需要在 url 中带上 bot\_id  # const url = wss://ws.coze.cn/v1/chat?bot\_id=73791654286875\*\*\*&authorization=Bearer pat\_OYDacMzM3WyOWV3Dtj2bHRMymzxP\*\*\*\*`;  const ws = new WebSocket(url);    ws.on('open', function open() {   console.log('Connected to server.');  });    ws.on('message', function incoming(*message*) {   console.log(JSON.parse(message.toString()));  }); |

**步骤二：发送和接收事件**

通过 WebSocket 与扣子智能体进行实时语音交互时，需要通过 WebSocket 接口发送和接收消息。成功连接后，客户端可以发送和接收代表文本、音频、配置更新等的事件。客户端可以发送的事件消息以及从服务器接收的事件消息列表请参见[双向流式对话事件](https://www.coze.cn/open/docs/developer_guides/streaming_chat_event)。

发送和接收事件的示例代码如下：

|  |
| --- |
| python # To send a client event, serialize a dictionary to JSON  # of the proper event type  def on\_open(ws):   print("Connected to server.")     data = {   "role": "user",   "content\_type": "text",   "content": "你好呀"   }   event = {   "event\_type": "conversation.message.create",   "data": data   }   ws.send(json.dumps(event))    # Receiving messages will require parsing message payloads  # from JSON  def on\_message(ws, message):   data = json.loads(message)   print("Received event:", json.dumps(data, indent=2)) |

**事件类型**

智能语音 WebSocket 事件包括上行事件和下行事件。每个事件有 ID 和 EventType，通过 EventType 可以区分具体的事件类型，每个事件类型对应的 Payload 在 Data 中，开发者可以按需去提取需要的内容。

* **上行事件**：设备端上报给服务端的事件。应用程序需要根据扣子平台提供的事件结构，在触发事件时填充字段内容并上报事件。
* **下行事件**：服务端下发给设备端的事件，应用程序需要解析下行事件，并根据业务需求进行下一步操作。

**注意事项**

* 每个上行事件 ID 建议不要重复，故障排查场景下便于定位问题。
* 每个事件有 ID 和 EventType，通过 EventType 可以区分具体的事件类型，每个事件类型对应的 Payload 在 Data 中，开发者可以按需去提取需要的内容。

**公共参数**

智能语音 WebSocket 事件的公共参数如下：

|  |  |  |
| --- | --- | --- |
| **参数名称** | **类型** | **描述** |
| id | String | 事件 ID，也就是事件的唯一标识。由客户端或服务端生成，在故障排查场景下用于定位具体的事件，便于排查问题。 |
| event\_type | String | 事件的类型。 |
| data | JSON | 事件的详细信息，其中包含具体事件的业务字段。 |