

版本：V1.3  
发行：2020 年 12 月

# RTC SDK

## 使 用 说 明 书

---

## 版本说明

版本	修订说明	修订时间	修订人员
1.3.0	增加 switchVideoSource 切换和 getVideoSource 获取视频源接口 增加和共享屏幕相关的四个参数， screenWidth 、 screenHeight 、 screenFrameRate 和 screenBandWidth 增加 videoSourceChanged 视频源变化事件	2020/12/10	皮慧斌
1.2.0	增加和共享桌面相关的两个参数， displayShareAudioOption 和 cancelCallWhenShareCanceled	2020/9/10	皮慧斌
1.1.0	增加 version 版本号接口	2020/9/8	皮慧斌
1.0.0	第一版	2020/9/7	皮慧斌

# 1 概览

## 1.1 SDK 简述

RTC SDK 是一套在 Web 上使用的音视频通话 SDK，Web 应用软件通过集成该 SDK，可以快速赋予该应用软件音视频通话能力。

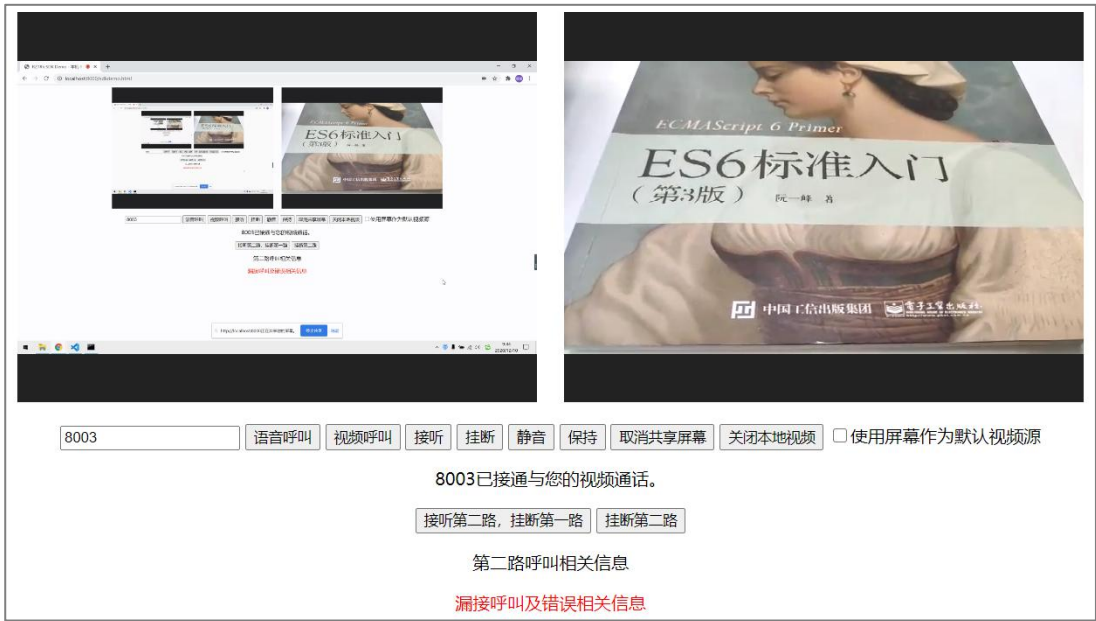


图 1-1 SDK DEMO 界面

## 1.2 SDK 功能接口

本套 SDK 提供以下功能接口：

- ✓ 双向语音通话
- ✓ 双向视频通话
- ✓ 通话中媒体操作，包括：
  - 切换摄像头/共享屏幕
  - 关闭/打开本地视频
  - 静音/取消静音
  - 保持/解保持

说明：在 Chrome 中此功能会造成音视频延迟，请慎用。

- ✓ 第二路来电提示

---

说明：不支持多路通话，接听第二路来电会将当前通话挂断。

## 1.3 SDK 运行条件

本套 SDK 需要运行于 Chrome 浏览器中，同时需要服务器侧支持 SIP 和 WEBRTC 协议。

## 1.4 文件清单

SDK 包含以下五个文件及本使用说明文档：

文件名	说明
rztrtcsdk.min.js	SDK 经压缩后的 js 文件
RTCSDK.d.ts	SDK typescript 类型声明文件
sdkdemo.html	SDK DEMO 文件
ringtone.wav	被叫振铃音文件
ringback.wav	主叫回铃音文件

以上五个文件在 rztrtcsdk 模块 sdk 目录下。

SDK 安装命令：

```
npm install rztrtcsdk
```

## 1.5 SDK DEMO

SDK DEMO 是使用本套 SDK 开发音视频通信功能的示例代码，通过运行和修改这个 DEMO，同时参考本文档中具体 API 使用说明，即可快速掌握使用本套 SDK 进行开发的方法。

### 1.5.1 如何运行 DEMO

#### 第一步：修改参数

修改 sdkdemo.html 中的创建 sdk 实例的参数，在 new RZTRtcSDK.RTCSDK 这一行（在大约第 48 行），这五个参数分别表示，

号码、密码、地址、端口、用户显示名

其他的重要参数如 rztppc、sessionExpires、发送视频的宽/高/帧率/码率等等，请参考本文档中的[设置参数](#)章节。

---

## 第二步：启动 http 服务器

在 sdkdemo.html 和 rztrtcsdk.min.js 所在目录下启动 http 服务器，可使用下面两种方法中的一种，

1. 安装有 python3

```
python -m http.server
```

2. 安装有 node

```
npm install http-server -g
```

```
http-server -p 8000 -c-1
```

## 第三步：打开 DEMO

在 Chrome 浏览器中输入 `http://localhost:8000/sdkdemo.html`。

## 2 API 说明

请在阅读 API 说明时，特别留意**红色文字**。

### 2.1 SDK 创建及设置

#### 2.1.1 创建 SDK 实例

声明：

```
constructor(account: string, password: string, url: string, port: number, displayName?: string);
```

创建 SDK 实例需要如下五个参数：

account, 用户名

password, 密码

url, 地址，可以是 IP 也可以是域名

port, wss 端口

displayName, 用户显示名

#### 2.1.2 设置参数 - setParams

声明：

---

```
setParams(paramSet: any): void;
```

paramSet 参数的说明如下：

```
sdk.setParams({  
    // 远端语音或视频标签，必须  
    remoteTag: remoteTag,  
    // 被叫振铃音，必须  
    ringRingUrl: 'ringtone.wav',  
    // 重要!!! 能缩短听到看到通话对端的时间，只适用于 RZT 并需要设置为 true  
    // 如不是 RZT 服务器，请置为 false 或不设置该值  
    rztpcc: true,  
    // 重要!!! 设置 session timer 的数值，RZT 需要设置为 240  
    sessionExpires: 240,  
    // 本地视频窗口标签，可选  
    localVideoTag: localVideoTag,  
    // 控制本地发送的视频的宽、高和帧率，可选  
    width: 640, height: 360, frameRate: 30,  
    // 控制本地发送的视频的码率，可选  
    bandWidth: 800,  
    // 控制共享屏幕时发送的视频的宽、高和帧率，可选  
    screenWidth: 1280, screenHeight: 720, screenFrameRate: 20,  
    // 控制共享屏幕时发送的视频的码率，可选  
    screenBandWidth: 1200,  
    // 共享屏幕时显示分享音频选项，共享屏幕默认屏蔽了这个选项  
    // displayShareAudioOption: true,  
    // 调用 call 或 answer 时使用屏幕作为视频源，但用户却最终取消了共享屏幕，这时候  
    // 仍然会呼出或接听（使用摄像头作为视频源）  
    // 下面这个参数置为 true 后，会改变这种默认行为，会取消呼出或挂断来电  
    // cancelCallWhenShareCanceled: true,  
    // 自定义 SIP USER AGENT  
    // userAgent: 'xxx',
```

---

```
// 自定义 STUN/TURN 设置 , RTCConfiguration ,
https://www.w3.org/TR/webrtc/#dom-rtcconfiguration
// 这个参数设置后, 前面的 rztpcc 将失效

// pcc: {
//   iceServers : [ {
//     urls      : [ 'turn:xxx.xxx.xxx.xxx' ],
//     username   : 'uuu_your_username_uuu',
//     credential : 'ppp_your_password_ppp'
//   } ],
//   iceTransportPolicy : 'relay',
//   rtcpMuxPolicy       : 'require'
// },
// 下面这两个参数, 对于 FreeSWITCH, 可不用设置, 如果回铃音在 183 early media
// 中提供了
// 主叫回铃音
// ringBackUrl: 'ringback.wav',
// 即使主叫收到了 183 仍然使用自己的主叫回铃音
// SC183StillRingBack: true,
// 禁用 session timer, 这个参数设置后, 前面的 sessionExpires 无效
// sessionTimer: false,
});
```

**注意：不能单独设置某一参数，每次调用 `setParams` 函数，都需要带上所有要设置的参数。**

### 2.1.3 设置事件处理回调 - `eventHandler`

声明：

`eventHandler`: any

```
sdk.eventHandler = function(event, data) { ... }
```

`event` 是事件的类型，`data` 是事件的数据。

具体事件的类型和数据，都在本文档的事件处理章节中详细说明。

---

#### 2.1.4 启动 SDK - start

声明:

```
start(): void;
```

#### 2.1.5 停止 SDK - stop

声明:

```
stop(): void;
```

#### 2.1.6 SDK 版本号 - version

声明:

```
version(): string;
```

#### 2.1.7 打开调试信息 - enableDebug

声明:

```
enableDebug(): void;
```

#### 2.1.8 关闭调试信息 - disableDebug

声明:

```
disableDebug(): void;
```

## 2.2 事件处理

在阅读本章节之前，请先了解[设置事件处理回调](#)部分。

#### 2.2.1 注册成功 - 'registered'

此事件没有任何数据。



---

### 2.2.2 来电 - 'incomingCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

remoteUser, 对端的用户名 (显示名)

remoteUserId, 对端的号码

callType, 呼叫类型, 为'audio'或'video', 表示语音或视频

**注意:** 可以使用此处的 **remoteUser** (该值从信令层面获得), 来显示对端的名称; 也可以通过 **remoteUserId** 在客户自己的系统中反查得到对端的名称, 有可能这种方法更适用一些。

**注意:** 来电的时候给用户提示, 请不要使用 **window** 对象的 **alert/confirm/prompt** 等函数, 这些函数会阻塞后台 **JavaScript** 脚本执行, 引起不能播放振铃音等各种异常。

### 2.2.3 对端振铃 - 'remoteRingRing'

事件数据:

```
const { remoteUser, remoteUserId, callType, statusCode } = data;
```

statusCode, 远端回铃码, 为 180 或 183

### 2.2.4 对端接听 - 'remoteAnswerCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

### 2.2.5 对端挂断 - 'remoteHangupCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

### 2.2.6 本地接听 - 'localAnswerCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

---

### 2.2.7 本地挂断 - 'localHangupCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

### 2.2.8 呼叫建立失败 - 'callEstablishFailed'

事件数据:

```
const { originator, cause } = data;
```

originator, 事件来源, 为'local'或'remote', 表示本地或远端

cause, 原因, 字符串

### 2.2.9 漏接呼叫 - 'missCall'

事件数据:

```
const { remoteUser, remoteUserId, callType } = data;
```

**注意:** 由于本 SDK 不支持多路通话, 当 SDK 不能处理新来电时, 会产生漏接呼叫。

### 2.2.10 错误 - 'failed'

事件数据:

```
const { type, reason } = data;
```

type, 类型, 可能为'accountError'或'connectionError'或'registrationFailed'或'disconnected'

reason, 原因, 字符串

**'accountError', 表示账号或密码错误**

**'connectionError', 表示 SDK 初始化后 10 秒仍连接不上服务器**

**'registrationFailed', 注册失败**

'disconnected', 和服务器连接中断

### 2.2.11 视频源变化 - 'videoSourceChanged'

此事件没有任何数据。

---

可使用 `getVideoSource` 来获取最新的视频源。

#### 2.2.12 第二路来电 - 'secondCall'

事件数据：

```
const { remoteUser, remoteUserId, callType } = data;
```

#### 2.2.13 第二路对端挂断 - 'remoteSecondCallHangup'

事件数据：

```
const { remoteUser, remoteUserId, callType } = data;
```

#### 2.2.14 第二路本地挂断 - 'localSecondCallHangup'

事件数据：

```
const { remoteUser, remoteUserId, callType } = data;
```

#### 2.2.15 第二路呼叫建立失败 - 'secondCallEstablishFailed'

事件数据：

```
const { originator, cause } = data;
```

参数说明同呼叫建立失败。

## 2.3 基本呼叫操作

### 2.3.1 呼出 - call

声明：

```
call(phoneNo: string, callType: string, shareScreen?: boolean): void;
```

phoneNo, 对端号码

callType, 呼叫类型, 为'audio'或'video', 表示语音或视频

shareScreen, 是否是共享屏幕模式

---

### 2.3.2 接听 - answer

声明:

```
answer(shareScreen?: boolean): void;
```

### 2.3.3 挂断 - hangup

声明:

```
hangup(): void;
```

### 2.3.4 是否在通话中 - inCall

声明:

```
inCall(): boolean;
```

### 2.3.5 是否可以接听 - canAnswer

声明:

```
canAnswer(): boolean;
```

### 2.3.6 获取对端信息 - getCallInfo

声明:

```
getCallInfo(): {  
    remoteUser: string;  
    remoteUserId: string;  
    callType: string;  
};
```

---

## 2.4 通话中操作

### 2.4.1 是否在保持中 - isOnHold

声明:

```
isOnHold(): boolean;
```

### 2.4.2 保持 - hold

声明:

```
hold(): void;
```

### 2.4.3 解保持 - unhold

声明:

```
unhold(): void;
```

### 2.4.4 获取视频源 - getVideoSource

声明:

```
getVideoSource(): string;
```

语音通话返回空"; 视频源为摄像头时返回'camera', 为共享屏幕时返回'screen'。

### 2.4.5 切换视频源 - switchVideoSource

声明:

```
switchVideoSource(newVideoSource: string): void;
```

切换视频源为'camera'或'screen'。

### 2.4.6 获取通话中视频的状态 - getVideoStatus

声明:

```
getVideoStatus(): boolean;
```

---

返回 true 表示在发送视频（摄像头已打开），返回 false 表示没有在发送视频（摄像头已关闭）。

#### 2.4.7 切换视频 - switchVideo

声明：

```
switchVideo(): void;
```

切换视频的发送，发送或停止发送视频，也表示打开或关闭摄像头。

#### 2.4.8 获取通话中语音的状态 - getAudioStatus

声明：

```
getAudioStatus(): boolean;
```

返回 true 表示声音已开启，返回 false 表示声音已关闭。

#### 2.4.9 切换语音 - switchAudio

声明：

```
switchAudio(): void;
```

切换声音的发送，静音或取消静音。

### 2.5 第二路来电处理

#### 2.5.1 是否有第二路来电 - hasSecondCall

声明：

```
hasSecondCall(): boolean;
```

#### 2.5.2 接听第二路来电 - answerSecondCall

声明：

```
answerSecondCall(shareScreen?: boolean): void;
```

---

### 2.5.3 挂断第二路来电 - hangupSecondCall

声明:

```
hangupSecondCall(): void;
```

### 2.5.4 获取第二路来电对端信息 - getSecondCallInfo

声明:

```
getSecondCallInfo(): {  
    remoteUser: string;  
    remoteUserId: string;  
    callType: string;  
};
```

## 2.6 错误处理

请参考《[事件处理 - 错误 - 'failed'](#)》

请参考《[事件处理 - 漏接呼叫 - 'missCall'](#)》

请参考《[事件处理 - 呼叫建立失败 - 'callEstablishFailed'](#)》

请参考《[事件处理 - 第二路呼叫建立失败 - 'secondCallEstablishFailed'](#)》