实时音视频

SDK 参考

文档版本 01

发布日期 2024-07-02

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 使用前必读

1.1 主要功能

SparkRTC主要包含基本房间功能和跨房功能，各端主要功能框架，如图1-1所示。

说明：图1-1中仅展示各端的统一功能，独属功能详见各端SDK指导。

图 1-1 功能框架

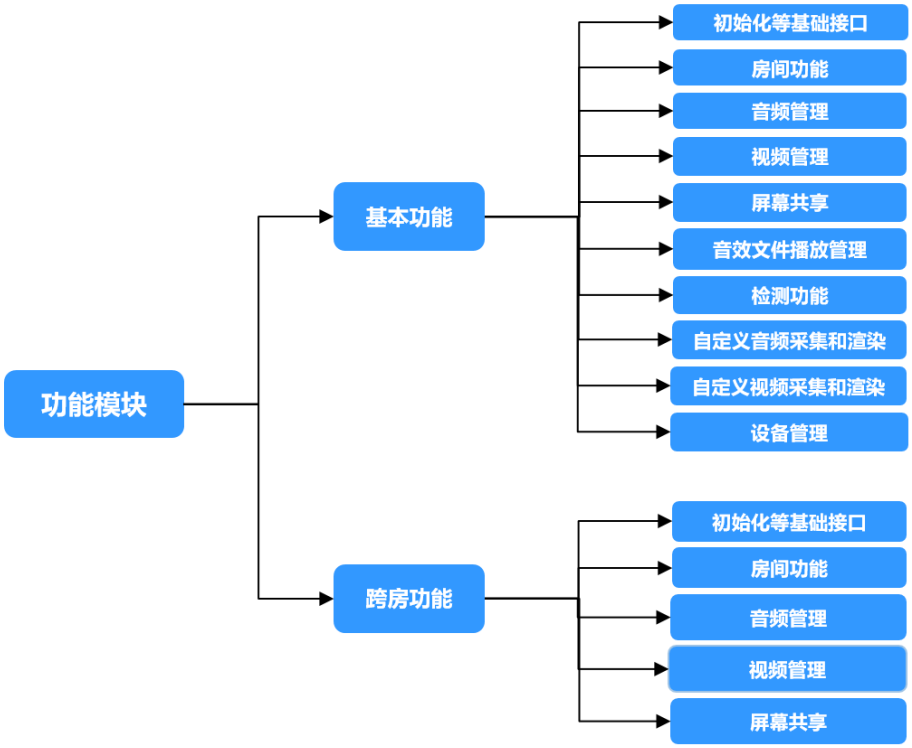


表 1-1 功能说明

类别	功能分类	功能说明
基本功能	初始化等基础接口	主要功能包括创建/销毁RTC引擎、设置日志保存位置等。
	房间功能	主要功能包括进入/离开房间操作、设置角色、创建跨房引擎等。
	音频管理	主要功能包括是否采集/发送本地音频流、是否接收远端音频流、调整录制/播放音量值、设置远端音频模式等。
	视频管理	主要功能包括创建本地/远端窗口视图和其他参数设置、是否接收远端视频流、镜像、摄像头等。
	屏幕共享	主要功能包括是否开始/停止订阅辅流、设置辅流渲染模式/角度等。
	音效文件播放管理	主要功能包括开始/停止/暂停/恢复播放音频或音效文件、音频/音效音量相关设置。
	检测功能	主要功能包括启动/关闭入会前网络检测。
	自定义音频采集和渲染	主要功能包括是否开启音频自采集、推送外部音频数据。
	自定义视频采集和渲染	主要功能包括是否开启视频自采集、推送外部视频数据、是否开启视频流自渲染。
	设备管理	主要功能包括切换摄像头、声音播放模式等。
跨房功能	跨房间连麦，指主播的媒体流可以同时转发进多个房间频道，实现主播跨频道与其他主播实时互动的场景。房间中的所有主播可以看见彼此，房间中的观众可以看到所有主播。同一时间最多只能同时跨4个房间，每个跨房房间的ID必须互不相同；同一时间只能以一个joiner角色加入某一个房间。如果本端在其他房间里的角色是joiner，则需要将本端在该房间内的player角色切换为joiner角色后再以joiner角色跨入其他房间。以player角色跨房后只能收流不能发流，以joiner角色跨房后既能收流也能发流。	

1.2 文档基本使用技巧

- **技巧1：基本使用逻辑说明**

  用时序图展示各端接口使用顺序，单击相应接口可以快速查看相关接口使用方法。

- **技巧2：接口总览说明**

  包括SparkRTC接口总体功能流程图和接口列表，根据功能分类可以快速查找具体功能单个接口，方便接口对接。

- **技巧3：单个接口使用须知**

  接口使用时注意使用的时机、参数说明。“注意”是强调每个接口使用的注意事项并带有调用该接口的相关回调，对接时需要仔细阅读。

1.3 常见问题分析解决办法

- **问题1：调用setVideoEncoderConfig接口时，为什么有些参数直接报参数设置错误?**

  设置分辨率时请使用华为SDK系统推荐的码表才能设置成功。

- **问题2：有哪些原因会导致跨房不成功?**
  - 可能原因1：同一时间不同房间最多只有一个Joiner角色才能跨房成功。
  - 可能原因2：同一时间最多只能跨4个房间，跨房对应的房间ID必须互不相同。

- **问题3：使用远端音频模式为HRTC_REMOTE_AUDIO_SUBSCRIBED时，如何设置才能默认听不到远端用户的声音?**

  HRTC_REMOTE_AUDIO_SUBSCRIBED为自主订阅，需要用户手动调用订阅。在加入房间（joinRoom）时调用带有HRTCJoinParam类的方法，创建该类实例后autoSubscribeAudio属性设置为false进入房间后则听不到远端用户的声音，需要手动调用muteRemoteAudio根据uid单个用户订阅才能听到声音。

- **问题3：为什么onVideoStats、onAudioStatus、onAuxiliaryStreamStatsNotify回调触发时程序崩溃?**

  回调函数的入参localStats和remoteStats指针有可能为空，需要先判断不为空再使用，否则可能引发空指针错误。

- **问题4：为什么本端听筒能听到自己的声音?**

  调用muteRemoteAudio时，参数设置为自己的uid就会发生此类情况。

- **问题5：setExternalAudioCapture（音频自采集）、setExternalVideoCapture（视频自采集）能在房间内开启吗?**

  不能，需要在加入房间前调用。

2 SDK 概述

华为云实时音视频服务软件开发工具包是对SparkRTC服务提供的REST API进行的封装，以简化用户的开发工作。用户直接调用SparkRTC SDK提供的接口函数即可实现使用SparkRTC服务业务能力的目的。

相关开发包请**提交工单**联系华为云技术客服获取。

同时，针对不同平台的SDK提供了集成和接口参考。详细介绍了SDK的集成操作、接口参数定义和代码示例。SparkRTC提供了以下主流平台SDK供开发者使用。

客户端	集成SDK	接口参考
Web	**Web SDK集成**	**Web SDK接口参考**

3 Web SDK

3.1 浏览器适配

本章节介绍Web SDK支持的浏览器类型、版本以及使用限制。

表 3-1 浏览器适配详情

操作系统类型	浏览器类型	浏览器版本	SDK版本约束	下行（播放）	上行（上麦）	屏幕分享
Windows	Chrome浏览器	67+	• v1.10.0及以上版本 • v2.0.0及以上版本	支持	支持	支持（Chrome 73+版本）
	QQ浏览器（极速内核）	10.4+				不支持
	360安全浏览器（极速模式）	12				支持
	微信内嵌浏览器	-	v2.0.0及以上版本	支持	不支持	不支持
	企业微信内嵌浏览器	-				
	Firefox浏览器	90+	v2.0.1及以上版本	支持	支持	支持
	Edge浏览器	80+	v2.0.2及以上版本	支持	支持	支持
	搜狗浏览器（高速模式）	11+				

操作系统类型	浏览器类型	浏览器版本	SDK版本约束	下行（播放）	上行（上麦）	屏幕分享
	Opera浏览器	54+				支持（Opera 60+ 版本）
macOS	Chrome浏览器	67+	• v1.10.0及以上版本 • v2.0.0及以上版本	支持	支持	支持（Chrome 73+ 版本）
	微信内嵌浏览器	-	v2.0.0及以上版本	支持	不支持	不支持
	企业微信内嵌浏览器	-				
	Safari浏览器	11+	v2.0.1及以上版本	支持	支持	支持（Safari 13+ 版本）
	Firefox浏览器	90+				支持
	Edge浏览器	80+				
	Opera浏览器	56+	v2.0.2及以上版本	支持	支持	支持（Opera 60+ 版本）
Android	微信内嵌浏览器（TBS 内核）	-	• v1.10.0及以上版本 • v2.0.0及以上版本	支持	支持	不支持
	微信内嵌浏览器（XWEB内核）	-				
	企业微信内嵌浏览器	-				
	移动版Chrome浏览器	-				
	移动版QQ浏览器	12+				

操作系统类型	浏览器类型	浏览器版本	SDK版本约束	下行（播放）	上行（上麦）	屏幕分享
	移动版华为浏览器	11.0.8+				
	移动版UC浏览器	-	-	不支持	不支持	不支持
iOS	微信内嵌浏览器	iOS 14.3+ 微信6.5+版本	• v1.10.0及以上版本 • v2.0.0及以上版本	支持	支持	不支持
	移动版Chrome浏览器	iOS 14.3+	v2.0.0及以上版本	支持	支持	不支持
	企业微信内嵌浏览器	-			不支持	
	移动版Safari浏览器	11+	v2.0.1及以上版本	支持	支持	不支持

表 3-2 浏览器使用限制

浏览器类型	使用限制
Chrome浏览器	• 在移动端浏览器上，**getSpeakers**接口只能获取到default音频输出设备。 • 在移动端浏览器上，不支持采集120p及以下的分辨率。 • 在华为移动端设备上，Chrome浏览器（包括华为自带浏览器）支持WebRTC的版本为91+。 • 在Mac Chrome浏览器上使用屏幕分享前，需确保已在“设置 > 安全性与隐私 > 隐私 > 屏幕录制”中打开Chrome屏幕录制授权。

浏览器类型	使用限制
Safari浏览器	• 在Safari 11及12上，需要在建链之前调用navigator.mediaDevices.getUserMedia接口（调用sdk接口**createStream**创建本地流）获取媒体权限，否则媒体无法交互（媒体链路无法建立）。 • 华为Native SDK推流，在Safari 11及12浏览器中选看，存在绿屏现象。 • Safari 13的用户可能听不到远端用户的声音。 • iOS Safari 14.2和macOS Safari 14.0.1上音频可能断断续续。 • Safari 15.1发布流时会发生异常，导致页面崩溃。 • Safari不支持获取输出设备信息，因此，不支持**getSpeakers**和**setAudioOutput**接口。 • Safari不支持调用**addTrack**和**removeTrack**接口。 • Safari浏览器不能多次调用本地流采集接口，否则会引起采集黑屏，需在调用音视频采集接口前关闭前一次采集。 • Safari不支持大小流。 • iOS移动端浏览器不支持混音功能。 • Mac设备Safari 12插耳机的场景下，开启混音后，远端听不到任何声音。 • 须在选看远端用户音频前调用navigator.mediaDevices.getUserMedia接口，否则无法听到声音和无法获取音频音量值。 • macOS Ventura系统，Safari 16.1使用屏幕分享时，会出现本地屏幕共享流黑屏。
Firefox浏览器	• Firefox只支持30fps视频帧率。 • Apple M1芯片的Mac设备上Firefox不支持H.264编解码。 • Firefox不支持获取输出设备信息，因此，不支持**getSpeakers**和**setAudioOutput**接口。 • Firefox 97+浏览器推流场景，其他端选看存在黑屏、卡顿兼容问题，正在紧急修复中。请暂时使用其他浏览器。
Opera浏览器	在华为移动端设备上，Opera浏览器支持WebRTC的版本为64+。
其他浏览器	由于各设备厂家的浏览器内核、webview、版本等因素，移动端浏览器对WebRTC的支持度不一，除可以使用**表3-1**中列举的移动端浏览器类型外，还可以集成使用Native SDK（Andriod / iOS）。

⚠ 注意

- 已集成Web SDK 1.0+版本（2.0+版本不涉及）的用户，请尽快升级至1.10.0+版本，否则在Chrome 96+浏览器上有可能出现无法使用的情况。
- Web SDK 2.0+版本是目前的主力构建版本，承载新功能及体验优化，建议您优先集成使用。Web SDK 1.0+版本仅做存量用户的维护，不再构建新的功能。
- Web SDK 1.0+和Web SDK 2.0+业务上不能互通，集成时需要注意。
- Safari浏览器上的使用限制和已知问题较多，建议使用兼容性较好的Chrome浏览器或者集成Native SDK。

3.2 开发前准备

前提条件

已**提交工单**获取SDK包。

环境要求

- 编译工具推荐安装Microsoft Visual Studio Code 1.43.2或以上版本。
- 如果客户端用Node.js开发， 推荐安装14.19.1或以上版本。
- 支持的浏览器详情请参见**Web浏览器适配详情**。
- 如果客户端用TypeScript开发，TypeScript的版本不低于3.8.3。
- 由于浏览器安全策略限制，仅支持通过**https://域名**的方式访问，或者直接在本地搭建服务器，通过**localhost:端口**访问，否则无法获取摄像头及麦克风的权限。

SDK 集成

步骤1 将获取的SDK压缩包放置在自己项目的“sdk”目录下。

步骤2 在项目代码中引入“hrtc”。

- 如果您通过<script>方式引入华为WebRTC SDK，则通过访问HRTC获取导出的模块：

```
<script src='./sdk/hrtc.js'>
    console.log(HRTC.VERSION)
</script>
```

- 如果您直接引用华为WebRTC SDK静态JS文件，则通过以下方式访问：

```
import HRTC from '../../sdk/hrtc'
console.log(HRTC.VERSION)
```

- 如果您通过npm模块化的方式引入华为WebRTC SDK，首先要安装hrtc模块，在package.json的开发依赖里引入hrtc ，如："hrtc": "./sdk/RtcSdk_Web_*.*.*.***.tar.gz"。在终端执行安装命令（版本号按实际替换）： npm install ，然后通过以下方式访问：

```
import HRTC from 'hrtc'
console.log(HRTC.VERSION)
```

----结束

3.3 SDK 使用

步骤1 检测浏览器是否兼容 SDK。具体接口详情请参见**checkSystemRequirements**。

```
async isBrowserSupport() {
  let check = false
  try {
    check = await HRTC.checkSystemRequirements()
    console.warn('browser isSupport: ' + check)
  } catch (error) {
    console.error(`check browser isSupport error: ${error.getCode()} - ${error.getMsg()}`)
    if (error.getCode() !== 90100025) {
      console.error(`browser Support part ability of RTC`)
      check = true
    }
  }
  return check
}
```

步骤2 创建客户端。具体接口详情请参见**createClient**。

```
let config = { appId,domain,countryCode }
let client = HRTC.createClient(config)
```

- **domain**：string[128]类型，服务器域名。该参数在SDK 1.0+版本中必填，SDK 2.0+版本中非必填。
- **appId**：string[128]类型，必填。应用ID，只有App ID相同的应用程序才能进入同一个房间进行互通。
- **countryCode**：string[2]类型，可选。国家码，如：CN表示中国大陆，US表示美国，HK表示中国香港。countryCode值的填写具体请参见**国家码对照表**。

说明

domain和**appId**请**提交工单**获取。

步骤3 加入房间。具体接口详情请参见**join**。

```
let option = { userId: userId, userName: userName, signature: signature, ctime: ctime, role: role }
async joinRoom() {
  try{
    await client.join(roomId, option)
    console.log('join room success')
  } catch(error){
    console.log('join room fail',error)
  }
}
```

- **userId**：必选，string[64] 类型，本端用户唯一标识。
- **userName**：可选，string[256]类型，用户昵称，该昵称为UTF-8编码。
- **signature**：必选，string[512]类型，鉴权签名字串，应用开发者需要向远端服务器获取鉴权签名。

  说明

  远端服务器需要您自行部署，具体请参见**接入鉴权**。
- **ctime**：必选，string类型，签名UTC时间戳，单位秒。
- **role**：必选，number类型，用户角色，可以标识媒体方向，取值如下：
  - 0：joiner（发布并观看）。
  - 2：player（只观看不发布）。

- **roomId**：必选，string[64]类型，房间ID，房间唯一标识。

加入房间成功后，对端会收到“peer-join”事件。

步骤4 创建本地流并发布。具体接口详情请参见**createStream**、**initialize**、**addResolution**、**publish**、**play**。

```
let stream = HRTC.createStream({ audio:true,microphoneId:xxx,video:true,cameraId:xxx })


stream.initialize().then(() => {
  stream.addResolution('90p_1')  //可选，如果要开启双流可以添加另外一个分辨率的视频

  stream.play(elementId,{ muted:true })  //播放本地流
  client.publish(stream)
})
```

- **audio**：可选，boolean类型，指定是否采集主流的音频。默认值为false。
- **video**：可选，boolean类型，指定是否采集主流的视频，主流即摄像头的流。默认值为false。
- **microphoneId**：可选，string类型，在audio为true的时候有效，表示采集音频的源麦克风设备Id。如果不传，系统自动设置默认值。
- **cameraId**：可选，string类型，在video为true的时候有效，表示采集视频的摄像头设备Id。如果不传，系统自动设置默认值。

步骤5 当收到服务器发送的“stream-added”事件通知时，可以订阅远端媒体。具体接口详情请参见**stream-added**、**subscribe**、**getStreamInfo**。

```
client.on('stream-added', (event) => {
  const stream = event.stream
  client.subscribe(stream,{ video:true, audio:true })
})
```

说明

```
如果双流场景：
client.on('stream-added', (event) => {
  const stream = event.stream
  const streamInfo = stream.getStreamInfo()  //获取流的分辨率等信息
  const resolutionIds = streamInfo.videoProfiles.map((profile) => profile.resolutionId)  // App 根据自己的业务场景，选择分辨率
  client.subscribe(stream,{ video:true, audio:true, resolutionIds:resolutionIds })  // 订阅音频以及所选择的分辨率的视频
  })
```

订阅完成之后，本端会收到“stream-subscribed”事件通知，可设置对端窗口，播放对端音视频。具体接口详情请参见**stream-subscribed**、**play**。

```
client.on('stream-subscribed', (event) => {
   const stream = event.stream
   stream.play(elementId, { objectFit: 'contain', muted: true, resolutionId: resolutionId })
})
```

- **elementId**：HTML 标签ID。
- 播放选项参数：
  - **objectFit**：可选，string类型，取值包括contain、cover和fill，默认值：主流：cover，辅流：contain。
  - **muted**：可选，boolean类型，true表示静音，false表示不静音，默认值为false。
  - **resolutionId**: 可选，string类型，指定要播放的分辨率的视频。默认为分辨率最高的视频。

若是不想观看对端，则可取消订阅对端音视频。具体接口详情请参见**unsubscribe**。

```
client.unsubscribe(stream)
```

步骤6 当远端离开房间，本端会收到“peer-leave”事件通知，清理远端用户的资源。具体接口详情请参见**peer-leave**。

```
client.on('peer-leave', (event) => {
  // just do something…
})
```

event.userId：对端用户标识，通过监听“peer-leave”事件获得。

远端用户退出，本端同时会收到“stream-removed”事件通知，可在事件处理函数中，关掉视频窗口。具体接口详情请参见**stream-removed**。

```
client.on('stream-removed', (event) => {
  event.stream.close()
})
```

通过stream对象调用**close**方法，该方法会移除之前用“play”创建的 video 标签元素并关闭摄像头、麦克风。

步骤7 本端离开房间。具体接口详情请参见**leave**。

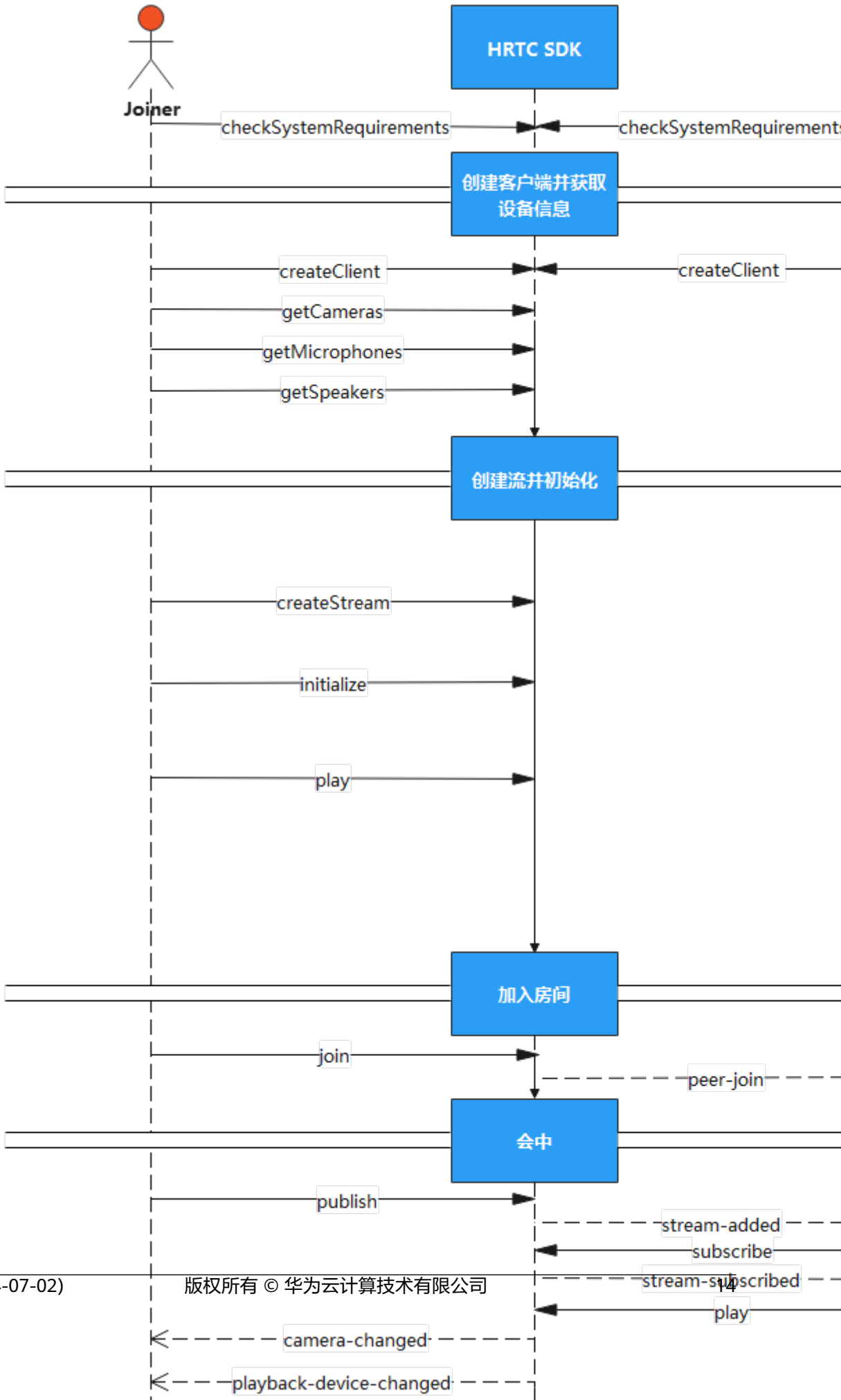
```
client.leave()
```

当音视频通话结束时，调用此接口离开房间。

至此，音视频通话基本流程可以成功运行。

----结束

3.4 基本使用逻辑



1. 创建新的项目工程，导入 SDK 后，需要创建客户端并获取本地音视频设备信息。
2. 创建本地流并初始化。
3. 当用户加入房间后，将通过回调的方式通知房间内的其他用户，收到用户加入的回调后，可以对音视频流进行订阅、取消订阅等其它操作。
4. 在会中，也可以对本地录音或播放设备等进行配置。
5. 用户离开房间后，房间内其他用户会收到该用户离开的回调信息，离开房间后，需销毁对应资源。

说明

在**时序图**中，单击相应接口名称可快速跳转到相应接口位置查看其使用方法。

3.5 接口参考

3.5.1 主入口（HRTC）

本章节介绍了Web SDK的HRTC接口详情。

表 3-3 主入口接口

接口	描述
checkSystemRequirements	检测浏览器是否兼容SparkRTC Web SDK。 **须知** 如果Web SDK版本在2.0.2到2.0.9.300之间，需要在2023年10月29日前更新至2.0.9.301及之后的版本，否则会导致checkSystemRequirements接口不可用。影响范围为微信浏览器。
VERSION	获取SparkRTC Web SDK版本。
getDevices	获取媒体输入输出设备列表。
getCameras	获取摄像头设备列表。
getMicrophones	获取麦克风设备列表。
getSpeakers	获取扬声器设备列表。
isScreenShareSupported	检查是否支持屏幕共享。
createClient	创建一个实时音视频通话的客户端对象。一个Client对应于一个房间。
createStream	创建一个本地流对象。流分为主流和辅流。主流指从摄像头和麦克风采集的音视频流，辅流是从共享屏幕采集的音视频流。
setLogLevel	设置日志级别。

checkSystemRequirements

须知

如果Web SDK版本在2.0.2到2.0.9.300之间，需要在2023年10月29日前更新至2.0.9.301及之后的版本，否则会导致checkSystemRequirements接口不可用。影响范围为微信浏览器。

```
(static) checkSystemRequirements(strictCheckBrowser: boolean): Promise<boolean>
```

【 功能说明 】

检测浏览器是否兼容SparkRTC Web SDK。

【 请求参数 】

strictCheckBrowser：可选，boolean类型，默认值为true 。true表示使用白名单放通支持的移动端浏览器（白名单由华为侧统一配置管理），false表示不使用白名单，该参数为2.0.2版本新增。

【 返回参数 】

Promise<boolean>：返回一个Promise对象，true表示浏览器兼容SparkRTC Web SDK，如果不兼容，则返回对应Error异常。

VERSION

```
VERSION
```

【 功能说明 】

获取 SparkRTC Web SDK 版本。

【 请求参数 】

无

【 返回参数 】

string：SDK当前版本号。

getDevices

```
(static) getDevices(): Promise<MediaDeviceInfo[]>
```

【 功能说明 】

获取媒体输入输出设备列表。在用户未授权摄像头或麦克风访问权限之前，“label”及“deviceId”可能为空。因此，建议在用户授权访问后，再调用此接口获取设备列表。

授权浏览器的摄像头/麦克风访问权限的方法，请参见**授权浏览器摄像头/麦克风访问权限的方法**。

【 请求参数 】

无

【 返回参数 】

Promise<MediaDeviceInfo[]>：媒体输入输出设备列表。**MediaDeviceInfo**为Web API基本接口。

getCameras

```
(static) getCameras(): Promise<MediaDeviceInfo[]>
```

【 功能说明 】

获取摄像头设备列表。在用户未授权摄像头访问权限之前，“label”及“deviceId”可能为空。因此建议在用户授权访问后，再调用此接口获取设备列表。

授权浏览器的摄像头/麦克风访问权限的方法，请参见**授权浏览器摄像头/麦克风访问权限的方法**。

【 请求参数 】

无

【 返回参数 】

Promise<MediaDeviceInfo[]>：摄像头设备列表。**MediaDeviceInfo** 为Web API基本接口。

getMicrophones

```
(static) getMicrophones(): Promise<MediaDeviceInfo[]>
```

【 功能说明 】

获取麦克风设备列表。在用户未授权麦克风访问权限之前，“label”及“deviceId”可能为空。因此建议在用户授权访问后，再调用此接口获取设备列表。

授权浏览器的摄像头/麦克风访问权限的方法，请参见**授权浏览器摄像头/麦克风访问权限的方法**。

【 请求参数 】

无

【 返回参数 】

Promise<MediaDeviceInfo[]>：麦克风设备列表。**MediaDeviceInfo**为Web API基本接口。

getSpeakers

```
(static) getSpeakers(): Promise<MediaDeviceInfo[]>
```

【 功能说明 】

获取扬声器设备列表。

【 请求参数 】

无

【 返回参数 】

Promise<MediaDeviceInfo[]>：扬声器设备列表。**MediaDeviceInfo**为Web API基本接口。

isScreenShareSupported

```
(static) isScreenShareSupported(): boolean
```

【 功能说明 】

检查浏览器是否支持屏幕共享。

【 请求参数 】

无

【 返回参数 】

boolean：true表示支持，false表示不支持。

createClient

```
(static)createClient(config: ClientConfig): Client
```

【 功能说明 】

创建一个实时音视频通话的客户端对象。一个客户端对象只能同时加入一个房间。可以创建多个客户端对象同时加入多个房间。

【 请求参数 】

config：必选，ClientConfig类型, 客户端对象配置信息。

ClientConfig定义为：{

- appId：必选，string[128]类型，应用ID，只有App ID相同的应用程序才能进入同一个房间进行互动。应用的appId请**提交工单**获取。
- domain：可选，string[128]类型，服务器的域名。需要与注册到SparkRTC平台的合法企业域名保持一致，该参数在SDK 1.0+版本中必填，SDK 2.0+版本中非必填。
- countryCode: 可选，string[2]类型，需要满足**ISO 3166-1**的2位字母的国家码要求。表示业务接入点的国家码，SDK会根据设置将业务接入到对应区域的服务，如：CN表示中国大陆，US表示美国，HK表示中国香港。countryCode值的填写具体请参见**国家码对照表**。该参数为2.0.3版本新增，且为必选参数，从2.0.7版本开始，修改为可选参数。

}

【 返回参数 】

Client：客户端对象。

createStream

```
(static) createStream(config: StreamConfig): Stream
```

【 功能说明 】

创建本地流对象。

【 请求参数 】

config：必选，StreamConfig类型，指定创建流的参数。

StreamConfig 定义为：{

- screen：可选，boolean类型，如果为true表示该流对象采集的是辅流音视频。辅流即共享屏幕的流。默认值为false，即该流对象采集的是主流音视频。
- video：可选，boolean类型，指定是否采集主流的视频，主流即摄像头的流。默认值为false。
- audio：可选，boolean类型，指定是否采集主流的音频。默认值为false。screen为false的时候该参数有效。
- microphoneId：可选，string类型，在audio为true的时候有效，表示采集音频的源麦克风设备Id。如果不传，系统自动设置默认值。
- cameraId：可选，string类型，在video为true的时候有效，表示采集视频的摄像头设备Id。如果不传，系统自动设置默认值。
- facingMode：可选，string类型，在video为true的时候有效，user表示前置摄像头，environment表示后置摄像头。
- screenAudio：可选，boolean类型，是否包含屏幕共享背景音。默认值为false。该功能仅支持Windows平台Chrome浏览器 74及以上版本。该字段为1.4.0版本新增。
- audioSource：可选，MediaStreamTrack类型，表示输入音轨对象。通过指定Track设置音频。**MediaStreamTrack**为Web API基本接口。
- videoSource：可选，MediaStreamTrack类型，表示输入视轨对象。通过指定Track设置主流视频。**MediaStreamTrack**为Web API基本接口。
- mirror：可选，boolean类型，表示主流的本地视频是否镜像。默认值为false。
- userId：可选，string类型，表示该流归属的userId。

}

【 返回参数 】

Stream：流对象。

⚠ 注意

- 在采集主流有两种方式：
  - 通过“audioSource”和“videoSource”设置音频和视频主流。这种模式下不支持大小流。
  - 通过“audio/microphoneId”和“video/cameraId/facingMode”设置音频和视频主流。
- 如果未指定任何的音频源和视频源，则创建的流对象不包含音频流和视频流，无法播放。
- 如果采集视频，同一个Stream对象不能同时采集主流和辅流。
- 如果需要包含屏幕共享背景音，需要设置screen和screenAudio均为true，该参数仅在1.4.0 及以上版本生效。

setLogLevel

```
(static) setLogLevel(level: LogLevel): void
```

【 功能说明 】

设置日志输出等级。默认输出info等级日志。

【请求参数】

level：必选，LogLevel类型，设置日志级别。

LogLevel表示日志级别，枚举取值如下：

- none：string类型，表示关闭SDK日志打印。
- error：string类型，表示开启SDK错误日志级别。
- warn：string类型，表示开启SDK警告日志级别。
- info：string类型，表示开启SDK信息日志级别。
- debug：string类型，表示开启SDK调试日志级别。

【返回参数】

无

3.5.2 客户端对象（Client）

本章节介绍了Web SDK的Client接口详情。

表 3-4 Client 接口

接口	描述
join	加入房间。调用该接口让用户加入指定房间，进行音频/视频通话。
leave	离开房间。用户结束通话后须调用该接口离开房间。
publish	加入房间后，发布本地流。
unpublish	取消发布本地流。
subscribe	订阅远端音视频媒体流。
unsubscribe	取消订阅远端音视频媒体。
batchSubscribe	批量订阅远端用户的视频媒体流。
switchRole	切换用户角色。
on	注册客户端对象事件回调接口。
off	取消注册客户端对象事件回调接口。
getConnectionState	获取客户端连接状态。
getTransportStats	获取当前网络传输状况统计数据。
getLocalAudioStats	获取本地音频统计数据。
getLocalVideoStats	获取本地视频统计数据。
getRemoteAudioStats	获取远端音频统计数据。

接口	描述
getRemoteVideoStats	获取远端视频统计数据。
enableTopThreeAudioMode	设置是否开启音频TopN模式。
setVolume4TopThree	设置音频TopN模式的音量值。该接口为1.4.0版本新增。
muteAudio4TopThree	开启/禁用音频TopN模式的所有音轨。该接口为1.4.0版本新增。
enableStreamStateDetection	开启/关闭视频码流状态探测功能。该接口为1.4.0版本新增。
changeUserName	修改用户昵称。该接口为1.5.0版本新增。
startLiveStreaming	启动直播流旁推到CDN。
updateLiveStreaming	更新旁路推流的合流参数。
stopLiveStreaming	停止直播流旁推到CDN。
setProxyServer	设置企业内部部署代理反向服务器。该接口为2.0.3版本新增。
setTurnServer	设置代理服务器配置信息。该接口为2.0.3版本新增。
enableRtcStats	设置是否开启RTC音视频流统计信息事件。该接口为2.0.3版本新增。
setNetworkBandwidth	设置媒体最大带宽。该接口为2.0.5版本新增。
renewSignature	更新签名。该接口为2.0.8版本新增。

join

```
async join(roomId: string, options: JoinConfig): Promise<void>
```

【功能说明】

加入房间。该接口让用户加入一个房间，进行音频或视频通话。

【请求参数】

- roomId：房间ID，房间唯一标识符。必选，string[64]类型。
- options：加入房间配置。必选，JoinConfig类型。
  JoinConfig 定义为：{
  - userId：用户标识，userId需要保证应用内唯一。必选，string[64]类型。
  - userName：用户昵称。可选，string[256]类型。
  - signature：签名，签名的具体生成方法请参见接入鉴权。必选，string[512]类型。

- ctime：必选，鉴权签名时间戳。string类型，UTC时间戳，单位秒。
- role：必选，number类型，用户角色，可以标识媒体方向。role的枚举值包括：
  - 0： 表示joiner，能够发送音视频和接收音视频。
  - 2： 表示player，只接收别人的音视频，不发送自己的音视频媒体。

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

- roomId支持的字符包括：a-z、A-Z、0-9、连接符 '-'、下划线 '_'。
- userId支持的字符包括：a-z、A-Z、0-9、连接符 '-'、下划线 '_'。

leave

```
async leave(): Promise<void>
```

【功能说明】

离开房间，用户结束通话后须调用该接口离开房间。

【请求参数】

无

【返回参数】

Promise<void>：返回一个Promise对象。

publish

```
async publish(stream: Stream, option?: PublishOption): Promise<void>
```

【功能说明】

加入房间并创建本地流后，可以调用该接口发布本地流。只有角色为“joiner”才能发布本地流。

【请求参数】

- stream：必选，Stream类型，本地流对象。
- option：可选，PublishOption类型，表示是否主动发布视频流，如果不传，则默认不主动发布视频流。

PublishOption类型定义如下：

{

autoPushVideo：可选，boolean类型，表示是否主动发布视频流，默认为false。

}

【返回参数】

Promise<void>：返回一个Promise对象。

unpublish

```
async unpublish(stream: Stream): Promise<void>
```

【功能说明】

取消发布本地流。

【请求参数】

stream：必选，Stream类型，本地流对象。

【返回参数】

Promise<void>：返回一个Promise对象。

subscribe

```
async subscribe(stream: RemoteStream, option?: SubscribeOption): Promise<void>
```

【功能说明】

订阅远端音视频媒体流。订阅远端媒体流成功后，会收到Client.on('**stream-subscribed**') 事件通知。然后可以对流进行播放。

【请求参数】

- stream：必选，RemoteStream类型，远端流对象，通过**stream-added**事件获得。
- option：可选，SubscribeOption类型，表示订阅视频或音频的选项，如果不传，则订阅音频和分辨率最高的视频。

SubscribeOption 类型定义如下：

{

- video：可选，boolean 类型，表示是否订阅视频，默认为 false。
- audio：可选，boolean 类型，表示是否订阅音频，默认为 false。
- resolutionIds：可选。string[]类型，video为true的时候有效。表示要订阅的分辨率Id的视频。resolutionId可通过接口**getStreamInfo** 获取。如果不传 resolutionIds，则默认订阅分辨率最高的一个视频。该参数为1.8.0 版本新增。

}

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

SubscribeOption中“video”和“audio”不能同时为“false”。

unsubscribe

```
async unsubscribe(stream: Stream,option?: SubscribeOption): Promise<void>
```

【功能说明】

取消订阅远端音视频媒体流。

【请求参数】

- stream：必选，RemoteStream类型，远端流对象，通过**stream-added**事件获得。
- option：可选。表示取消订阅的选项，如果不传，则取消订阅音频和所有分辨率的视频，为SubscribeOption类型。

  SubscribeOption定义如下：{
  - video：可选，boolean类型，表示是否取消订阅视频，默认为false。
  - audio：可选，boolean类型，表示是否取消订阅音频，默认为false。
  - resolutionIds：可选，string[]类型，在video为true的时候有效，表示取消订阅的分辨率Id的视频。resolutionId通过接口**getStreamInfo** 获取。如果resolutionIds不传，则取消订阅所有分辨率的视频。

  }

【返回参数】

Promise<void>：返回一个Promise对象。

batchSubscribe

```
async batchSubscribe(subscribeInfos: SubscribeParam[]): Promise<void>
```

【功能说明】

批量订阅多个用户的视频。订阅远端媒体流成功后，对每个远端用户都会收到Client.on('**stream-subscribed**')事件通知。然后可以对订阅成功的流进行播放。

【请求参数】

subscribeInfos：必选，SubscribeParam[]类型，表示要订阅的全量远端用户的视频信息。

SubscribeParam类型定义如下：{

- userId：必选，string类型，表示要订阅的用户ID。
- resolutionIds：可选。string[]类型，表示要订阅的用户的视频的分辨率。如果不传resolutionIds，则默认订阅分辨率最高的一个视频。
- minResolution：可选，ResolutionType类型，表示如果开启视频分辨率自适应升降档后，最小的分辨率档位。ResolutionType包括的字符串枚举值包括FHD，HD，SD，LD。

}

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

- 该接口订阅远端用户的视频，不订阅远端用户的音频。
- 不同于**subscribe** 接口的增量订阅方式，该接口订阅的入参是要订阅的全量用户列表。

switchRole

```
async switchRole(role: number,authorization: Authorization): Promise<void>
```

【功能说明】

切换用户角色，本方法用于加入房间成功之后修改角色。

【请求参数】

- role：必选，number类型。
  - 0：表示joiner，能够发送音视频和接受音视频。
  - 2：表示player，只接受别人的音视频，不发送自己的音视频媒体。
- authorization：必选，类型为Authorization，表示鉴权信息。该字段为2.0.0版本新增。Authorization的属性包括：
  - signature：必选，string[512]类型，鉴权签名字串，具体生成方法请参见**接入鉴权**。
  - ctime：必选，鉴权签名时间戳。string类型，UTC时间戳，单位秒。

【返回参数】

Promise<void>：返回一个Promise对象。

on

```
on(event: string, handler: function): void
```

【功能说明】

注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。具体请参见**ClientEvent**。
- handler：必选，function类型，事件处理方法。

【返回参数】

无

off

```
off(event: string, handler: function): void
```

【功能说明】

取消注册客户端对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。具体请参见**ClientEvent**。
- handler：必选，function类型，事件处理方法。

【返回参数】

无

getConnectionState

```
getConnectionState(): ConnectionState
```

【 功能说明 】

获取客户端连接状态。

【 请求参数 】

无

【 返回参数 】

ConnectionState：webSocket的连接状态，string类型，取值如下：

- CONNECTING：连接建立中。
- CONNECTED：连接已建立。
- RECONNECTING：重新连接中。
- DISCONNECTED：连接已断开。

getTransportStats

```
getTransportStats(): Promise<TransportStats>
```

【 功能说明 】

获取当前网络传输状况统计数据，该方法需要**publish**后调用。双流场景，默认获取主流最高分辨率视频的统计值。

【 请求参数 】

无

【 返回参数 】

返回参数为TransportStats类型，TransportStats当前支持的属性包括：{

- bytesSent：number类型，表示已发送字节数。
- bytesReceived：number类型，表示已接收字节数。
- sendBitrate：number类型，表示当前出流码率，单位kbps。
- recvBitrate：number类型，表示当前入流码率，单位kbps。
- rtt：number类型，表示SDK到边缘服务器的RTT (Round-Trip Time)，单位毫秒。

  }

getLocalAudioStats

```
getLocalAudioStats(): Promise<Map<string, LocalAudioStats>>
```

【 功能说明 】

获取本地已发送的音频统计数据。

【 请求参数 】

无

【 返回参数 】

- string：表示userId。
- LocalAudioStats：表示本地音频统计指标，包含如下属性：{

  - bytesSent：number类型。表示已发送字节数。
  - packetsSent：number类型。表示已发送包数。
    }

getLocalVideoStats

```
getLocalVideoStats(): Promise<Map<string, AllLocalVideoStats>>
```

【功能说明】

获取本地已发送的视频统计数据。

【请求参数】

无

【返回参数】

- string：表示userId。
- AllLocalVideoStats：本地视频统计指标。包含如下属性：{
  - mainStream：LocalVideoStats[]类型，本地主流视频的统计数据。
  - subStream：LocalVideoStats类型，本地辅流视频的统计数据。
    }

  LocalVideoStats包含如下属性：{
  - bytesSent：number类型，已发送字节数。
  - packetsSent：number类型，已发送包数。
  - framesEncoded：number类型，已编码帧数。
  - framesSent：number类型，已发送帧数。
  - frameWidth：number类型，视频宽度。
  - frameHeight：number类型，视频高度。
    }

⚠ 注意

getLocalVideoStats和getLocalAudioStats接口仅在本地流发布且被远端订阅后才可调用。

getRemoteAudioStats

```
getRemoteAudioStats(): Promise<Map<string, RemoteAudioStats>>
```

【功能说明】

获取远端音频统计数据。

【请求参数】

无

【返回参数】

- string：表示userId。

- RemoteAudioStats：远端音频统计指标。包含如下属性：
  - bytesReceived：number类型，已接收字节数。
  - packetsReceived：number类型，已接收包数。
  - packetsLost：number类型，丢包数。

getRemoteVideoStats

```
getRemoteVideoStats(): Promise<Map<string, AllRemoteVideoStats>>
```

【功能说明】

获取远端视频统计数据。

【请求参数】

无

【返回参数】

- string：表示userId。
- AllRemoteVideoStats：远端视频统计指标。包含如下属性：{
  - mainStream：RemoteVideoStats[]类型，远端主流（包括大小流）的统计数据。
  - subStream：RemoteVideoStats类型，远端辅流的统计数据。

  }

  RemoteVideoStats 包含如下属性：{
  - bytesReceived：number类型，已接收字节数。
  - packetsReceived：number类型，已接收包数。
  - packetsLost：number类型，丢包数。
  - framesDecoded：number类型，已解码帧数。
  - frameWidth：number类型，视频宽度。
  - frameHeight：number类型，视频高度。

  }

⚠ 注意

Firefox浏览器上，无法获取到远端视频的分辨率数据。

enableTopThreeAudioMode

```
enableTopThreeAudioMode(enable: boolean): boolean
```

【功能说明】

设置入会前是否开启音频TopN 模式（最大三方模式）。

【请求参数】

enable：必选，boolean类型，true表示开启音频TopN 模式，false表示不开启。默认为false。

【返回参数】

boolean类型，true表示开启成功，false表示开启失败。

⚠ 注意

该接口需在加入房间前设置，1.2.0版本新增。

setVolume4TopThree

```
setVolume4TopThree(volume: number): void
```

【功能说明】

开启音频TopN模式（最大三方模式）后，设置音频的音量值。

【请求参数】

volume：必选，number类型，取值范围为[0,100]，音频的音量值。

【返回参数】

无

⚠ 注意

该接口需要在**enableTopThreeAudioMode**后设置，1.4.0版本新增。

muteAudio4TopThree

```
muteAudio4TopThree(enable: boolean): void
```

【功能说明】

开启音频TopN模式（最大三方模式）后，开启/禁用音频TopN模式的音轨。

【请求参数】

enable：必选，boolean类型，true表示禁用音频TopN模式的音轨，false表示开启音频TopN模式的音轨。默认为false。

【返回参数】

无

⚠ 注意

该接口需要在**enableTopThreeAudioMode**后设置，1.4.0版本新增。

enableStreamStateDetection

```
async enableStreamStateDetection(enable: boolean, interval: number): Promise<void>
```

【功能说明】

开启/关闭视频码流状态探测功能，开启后可探测房间内任意其他已订阅用户是否处于视频无码流的状态，若有用户处于视频无码流状态，可收到**stream-interrupted**事件，若有用户视频码流恢复，可收到**stream-recovered**事件。

【 请求参数 】

- enable：必选，boolean类型，true表示开启视频码流状态探测，false表示关闭视频码流状态探测。默认为false。
- interval：必选，number类型，单位：秒 ，取值范围为[1,60]。视频无码流状态的判断时间。推荐设置为3秒。

【 返回参数 】

Promise<void>：返回一个Promise对象。

⚠ 注意

该接口需在加入房间后设置，1.4.0版本新增。

changeUserName

```
async changeUserName(userName: string): Promise<boolean>
```

【 功能说明 】

Joiner角色的用户修改用户昵称，修改成功后，房间内其他用户会收到**remote-user-name-changed**事件，而Player角色的用户修改后不会通知房间内的其他用户。

【 请求参数 】

userName：必选，string[256]类型，新的用户昵称。

【 返回参数 】

Promise<boolean>：返回一个Promise对象，true表示修改成功，false表示修改失败。

⚠ 注意

该接口需在加入房间后设置，1.5.0版本新增。

startLiveStreaming

```
async startLiveStreaming(taskId: string, urls: string[], publishConfig: PublishConfig, userConfigs: UserConfig[]): Promise<void>
```

【 功能说明 】

启动直播流旁推到CDN。

【 请求参数 】

- urls：必选，string[]类型，旁推的CDN Urls。
- taskId：必选， string类型，表示旁推任务Id。由APP生成和管理。

- publishConfig：必选，PublishConfig类型，表示旁推音视频配置。PublishConfig定义如下：{
  - width：必选，number类型，表示推流视频的总宽度，默认值360，单位为像素。
  - height：必选，number类型，表示推流视频的总高度，默认值640，单位为像素。
  - videoBitrate：必选，number类型，表示推流视频的码率，单位为Kbps，默认值为400 Kbps。
  - videoFramerate：必选，number类型，表示推流视频的帧率，单位为fps，取值范围为[0, 60]，默认值为15 fps。
  - videoGop：必选，number类型，表示旁路直播的输出视频的GOP，单位为帧。默认值为30帧。
  - audioSampleRate：必选，number类型，表示旁路直播的输出音频的采样率16000/32000/48000，默认值为32000。
  - audioBitrate：必选，number类型，用于旁路直播的输出音频的码率。单位为Kbps，默认值为48。
  - audioChannels：必选，number类型，用于旁路直播的输出音频的声道数，取值范围为[1, 5]，默认值为2。
  - template：可选，number类型，0表示悬浮，1表示九宫格，2表示屏幕分享，99表示自定义模板，默认值为0。

    }
- userConfigs：必选，UserConfig[]类型。 表示需要旁推的每个用户的配置。UserConfig定义如下：{
  - userId：必选，string类型，表示流用户Id。
  - audio：必选，boolean类型，表示是否旁推音频。
  - resolutionIds：必选，string[]类型，表示要旁推的主流或者辅流的视频分辨率ID。
  - layouts：可选，ResolutionLayout[]类型 ，在template为自定义模板有效，表示自定义用户流画面布局。ResolutionLayout定义如下：{
    - resolutionId：必选，string类型，表示主流或者辅流的视频分辨率ID。
    - subWidth：可选，number类型，表示画面在输出时的宽度，单位为像素值，默认值为36。
    - subHeight：可选，number类型，表示画面在输出时的高度，单位为像素值，默认值为64。
    - localX：可选，number类型，表示该画面在输出时的X偏移，单位为像素值，localX与subWidth之和不能超过混流输出的总宽度，默认值为0。
    - localY：可选，number类型，表示该画面在输出时的Y偏移，单位为像素值，localY与subHeight之和不能超过混流输出的总高度，默认值为0。
    - renderMode：可选，number类型，表示该画面在输出时的显示模式，0表示裁剪，1表示缩放。默认值为0。

- alpha：可选，number类型，表示该用户视频图像在输出视频图像中的透明度，取值范围为0~100。0表示该用户视频图像完全透明，100表示该用户视频图像完全不透明。默认值为100。
- subBackGroundColor：可选，number类型，表示混流-输出流背景色，取值是十进制整数。默认为黑色。
- zorder：可选，number类型，标志图层。

}
}

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

- 该方法只有角色为joiner的用户才能使用该功能，2.0.1版本新增。
- 请确保在publish流成功之后调用本方法。

updateLiveStreaming

```
async updateLiveStreaming(taskId: string, urls: string[], publishConfig: PublishConfig, userConfigs: UserConfig[]): Promise<void>
```

【功能说明】

更新旁路推流。

在推流状态变更时，本地会触发Client.on(“**live-streaming-updated**”) 回调。

【请求参数】

和**startLiveStreaming**相同。

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

该方法需在启动直播流旁推成功后调用，2.0.1版本新增。

stopLiveStreaming

```
async stopLiveStreaming(taskId: string): Promise<void>
```

【功能说明】

停止直播流旁推到CDN。该方法在2.0.1版本新增。

【请求参数】

taskId：必选，string类型，旁路推流的任务Id。

【返回参数】

Promise<void>：返回一个Promise对象。

enableRtcStats

```
async enableRtcStats(enable: boolean, interval: number): Promise<void>
```

【功能说明】

设置是否开启RTC音视频流统计信息事件。该接口为2.0.3版本新增。

【请求参数】

enable：必选，boolean类型，表示是否开启RTC音视频流统计信息事件，true表示开启，false表示不开启。

interval：必选，number类型，设置RTC音视频流统计信息事件间隔时间，单位毫秒。当enable为true时有效。

【返回参数】

Promise<void>：返回一个Promise对象。

setProxyServer

```
setProxyServer(server: string): void
```

【功能说明】

设置信令代理服务器。用于企业内部部署反向代理服务器（如 nginx ）的场景。该方法为2.0.3版本新增。

【请求参数】

server：必选，string类型，反向代理服务器的列表。一个代理服务器的格式，比如：http://ip:port / https://domain:port。

【返回参数】

无

⚠ 注意

setProxyServer和setTurnServer方法需在join之前调用。

setTurnServer

```
setTurnServer(turnServerConfig: TurnServerConfig): void
```

【功能说明】

设置代理服务器配置信息。用于企业内部部署反向代理服务器（如nginx）的场景。该方法为2.0.3版本新增。

【请求参数】

turnServerConfig：代理服务器配置信息。必选，TurnServerConfig类型。

TurnServerConfig 类型定义如下：{

- turnServers：反向代理服务器地址。必选，string[]类型。
- udpPort：UDP端口。可选，number类型。
- userName：反向代理服务器用户名。可选，string类型。
- credential：反向代理服务器密码。可选，string类型。

  }

【 返回参数 】

无

setNetworkBandwidth

```
setNetworkBandwidth(bandwidthParam: NetworkBandwidth): void
```

【 功能说明 】

设置媒体最大带宽。该方法为2.0.5版本新增。

【 请求参数 】

NetworkBandwidth类型定义如下：{

maxBandwidth：必选，number类型，媒体最大总带宽，取值范围为[3072, 51200]，单位为kbps。

}

【 返回参数 】

无

renewSignature

```
renewSignature(ctime: string, signature: string): boolean
```

【 功能说明 】

更新签名。

【 请求参数 】

- ctime：签名鉴权的过期时间，是系统当前UTC时间（unix时间戳）加上鉴权过期时间（推荐2小时，最长需要小于12小时）。单位：秒。必选，string类型。
- signature：签名，签名的具体生成方法请参见**接入鉴权**。必选，string类型，string[512]类型。

【 返回参数 】

boolean：返回一个boolean值，说明更新签名是否成功。

⚠ 注意

该接口2.0.8版本新增。

3.5.3 客户端事件通知（ClientEvent）

本章节介绍了Web SDK的ClientEvent事件。

表 3-5 ClientEvent 事件

接口	描述
peer-join	远端用户进入房间事件。
peer-leave	远端用户退出房间事件。
stream-added	远端流添加事件。
stream-removed	远端流删除事件。
stream-updated	远端流更新事件。
stream-subscribed	远端流订阅成功事件。
client-banned	用户被踢，离开房间事件。
kick-room	解散房间
Error	客户端出现错误事件。
connection-state-changed	Client连接状态变更事件。
mute-audio	远端流禁用音频事件。
unmute-audio	远端流启用音频事件。
mute-video	远端流禁用视频事件。
unmute-video	远端流启用视频事件。
log-upload-result	日志上传结果事件。
signature-expired	签名过期事件。该事件为2.0.8版本新增。
camera-changed	摄像头设备变更事件。
recording-device-changed	录音设备变更事件。
playback-device-changed	播放设备变更事件。
network-quality	网络上下行质量报告事件。
stream-interrupted	远端流处于无码流状态事件。该事件为1.4.0版本新增。
stream-recovered	远端流码流恢复事件。该事件为1.4.0版本新增。
volume-indicator	音频TopN（即最大三方）模式下，当前音量最大的用户提示事件。该事件为1.5.0版本新增。
remote-user-name-changed	远端用户昵称变更事件。该事件为1.5.0版本新增。
live-streaming-updated	更新旁路推流的事件。该事件为2.0.0版本新增。
rtc-stats	音视频流数据统计事件。该事件为2.0.3版本新增。

> **⚠ 注意**
>
> 事件注册监听应在业务结束时取消注册，否则注册监听事件累积会有内存泄漏风险。

peer-join

【事件说明】

远端用户加入房间事件，远端用户加入房间后会收到该事件通知。

【回调参数】

peerJoinEvent：必选，PeerJoin类型，用户信息。

PeerJoin定义为：{

- userId：必选，string[64]类型，用户标识。
- userName：可选，string[256]类型，用户昵称。

}

peer-leave

【事件说明】

远端用户离开房间事件，远端用户离开房间后会收到该事件通知。

【回调参数】

peerLeaveEvent：必选，PeerLeaveInfo 类型，用户离开信息。

PeerLeaveInfo 定义为：{

- userId：必选，string[64]类型，用户标识。
- userName：可选，string[256]类型，用户昵称。
- reason：可选，HRTCLeaveReason类型。

}

HRTCLeaveReason定义为：{

- code：number类型, 离开原因枚举，取值如**表2 离开房间原因**所示。
- msg：string类型，原因描述。

}

表 3-6 离开房间原因

枚举值	描述
0	用户主动离开
1	服务器异常
2	sfu服务故障

枚举值	描述
3	服务不可达503
4	内部错误
5	被踢出房间
6	签名过期
7	重连超时
8	网络检测，UI不需要关注该错误码，不对外体现
9	用户移除
10	房间解散

stream-added

【事件说明】

远端流添加事件，当远端用户发流成功后会收到该事件通知。

【回调参数】

stream：必选，RemoteStream类型，远端流对象。

stream-removed

【事件说明】

远端流移除事件，当远端用户取消发流或者原来已经发布的远端用户退出房间后会收到该事件通知。

【回调参数】

stream：必选，RemoteStream类型，远端流对象。

stream-updated

【事件说明】

远端流更新事件，当远端用户的流发生变化，例如添加或者移除音视频轨，视频轨规格变化时会收到该事件通知。

【回调参数】

stream：必选，RemoteStream类型，远端流对象。

stream-subscribed

【事件说明】

远端流订阅成功事件，当订阅远端流成功后会收到该事件通知。

【回调参数】

stream：必选，RemoteStream类型，远端流对象。

client-banned

【事件说明】

用户被踢下线事件，当用户以相同的userId 在其他Client加入相同的房间后，被踢的Client会收到该事件通知。

【回调参数】

clientBannedEvent：必选，ClientBanInfo类型，

ClientBanInfo定义为：{

- userId：必选，string[64]类型，被踢用户标识。
- reason：必选，string类型，原因描述。

}

kick-room

【事件说明】

解散房间，由调用的服务端接口触发。

【回调参数】

roomId：必选，string[64]类型，房间Id。

Error

【事件说明】

客户端错误事件，当出现不可恢复的错误后，Client会上报该事件通知。

【回调参数】

errorInfo：必选，ErrorInfo类型，错误信息。

ErrorInfo 定义为：{

- errorCode：必选，string类型，错误码。
- errorMsg：必选，string类型，错误描述。

}

connection-state-changed

【事件说明】

Client连接状态变更事件，Client连接状态变更会收到该事件通知。

【回调参数】

- ConnectionStateInfoEvent：{
  - prevState：必选，ConnectionState类型，变更前状态。
  - curState：必选，ConnectionState类型，变更后状态。

}

连接状态ConnectionState的取值如下所示：

- CONNECTING：连接建立中。
- CONNECTED：连接已连接。
- RECONNECTING：重新连接中。
- DISCONNECTED：连接已断开。

mute-audio

【事件说明】

远端用户禁用音频通知事件，当远端用户禁用音频后，流接收端会收到该事件通知。

【回调参数】

mediaStatus：必选，MediaStatusNotifyInfo类型。

MediaStatusNotifyInfo包含如下属性：

- roomId：必选，string[64]类型。
- userId：必选，string[64]类型。
- status：必选，MediaStatusAction类型。MediaStatusAction枚举值定义如下所示：
  - 1：媒资状态可用。
  - 2：媒资状态不可用。
- reason：必选，MediaStatusReason类型。MediaStatusReason枚举值定义如下所示：
  - 0：媒体离线。
  - 1：媒体静音。
  - 2：媒体不静音。

unmute-audio

【事件说明】

远端用户启用音频通知事件，当远端用户启用音频后，流接收端会收到该事件通知。

【回调参数】

mediaStatus：必选，**MediaStatusNotifyInfo**类型。

mute-video

【事件说明】

远端用户禁用视频通知事件，当远端用户禁用视频后，流接收端会收到该事件通知。

【回调参数】

mediaStatus：必选，**MediaStatusNotifyInfo**类型。

unmute-video

【事件说明】

远端用户启用视频通知事件，当远端用户启用视频后，流接收端会收到该事件通知。

【回调参数】

mediaStatus：必选，**MediaStatusNotifyInfo**类型。

log-upload-result

【事件说明】

日志上传结果事件。

【回调参数】

status：必选，number类型，日志上传结果，200表示成功，其余表示失败。

signature-expired

【事件说明】

签名过期事件。

【回调参数】

errorInfo：错误信息。必选，ErrorInfo类型。

ErrorInfo 定义为：{

- errorCode：错误码。必选，string类型。
- errorMsg：必选，string类型。取值如下面的取值示例中所示。

}

取值示例，如下所示：

- 签名过期：{
  errorCode: '90100030'
  errorMsg: 'signature expired'
  }
- 签名无效：{
  errorCode: '90100031'
  errorMsg: 'signature invalid'
  }

注意

监听到签名过期事件后可通过错误码区分是签名无效还是签名过期，签名过期后可通过**renewSignature**接口更新签名。

camera-changed

【事件说明】

本地摄像头设备变更事件，当插、拔本地摄像头时触发。

【回调参数】

DeviceChangedEvent：必选，DeviceChangedInfo类型，设备变更详情。

DeviceChangedInfo 定义为：{

- deviceId：必选，string类型，设备deviceId。
- state：必选，DeviceChangeMode类型，DeviceChangeMode枚举值定义如下：
  - ADD：设备添加。
  - REMOVE：设备移除。

}

⚠ 注意

视频采集设备，拔插后需要应用层进行相应的处理，如：拔除摄像头的时候是否切换其他视频采集设备重新采集；插入摄像头的时候是否使用新插入设备进行重新采集。

recording-device-changed

【事件说明】

本地录音设备变更事件，当本地录音设备变更时触发。

【回调参数】

DeviceChangedEvent：必选，DeviceChangedInfo类型，设备变更详情。
DeviceChangedInfo定义参考**camera-changed**中的说明。

⚠ 注意

麦克风采集设备，拔插后需要应用层进行相应的处理，如：拔除麦克风的时候是否切换其他麦克风采集设备重新采集；插入麦克风的时候是否使用新插入设备进行重新采集。

playback-device-changed

【事件说明】

本地音频播放设备变更事件，当本地音频播放设备变更时触发。

【回调参数】

DeviceChangedEvent：必选，DeviceChangedInfo类型，设备变更详情。
DeviceChangedInfo定义参考**camera-changed**中的说明。

network-quality

【事件说明】

网络上下行质量报告事件，用户加入房间后，SDK在网络质量变化的时候会触发一次该事件，报告用户的本地网络上下行质量情况。

【回调参数】

NetworkQualityEvent：必选， NetworkQualityInfo类型，网络上下行质量详情。

NetworkQualityInfo定义为：{

- uplinkNetworkQuality：必选，number类型，上行网络质量。枚举值如下：
  - 0：质量未知。
  - 1：质量极好。
  - 2：用户主观感觉和极好差不多，但码率可能略低于极好。
  - 3：用户主观感受有瑕疵但不影响沟通。
  - 4：勉强能沟通但不顺畅。
  - 5：网络质量非常差，基本不能沟通。
  - 6：网络连接断开，完全无法沟通。
- downlinkNetworkQuality：必选，number类型，下行网络质量。枚举值如下：
  - 0：质量未知。
  - 1：质量极好。
  - 2：用户主观感觉和极好差不多，但码率可能略低于极好。
  - 3：用户主观感受有瑕疵但不影响沟通。
  - 4：勉强能沟通但不顺畅。
  - 5：网络质量非常差，基本不能沟通。
  - 6：网络连接断开，完全无法沟通。

}

stream-interrupted

【事件说明】

远端流的码流中断事件。中断表示在 **enableStreamStateDetection** 接口参数 interval 设置的统计周期内未接收到有效视频帧。该事件为 1.4.0 版本新增。

【回调参数】

streamInterruptedEvent：必选，UserList[]类型，已订阅且无视频码流的用户列表。

UserList定义为：{

- userId：必选，string类型，用户Id。
- isScreen：必选，boolean类型，true表示对应用户的辅流，false表示对应用户的主流。

}

stream-recovered

【事件说明】

远端流的码流恢复事件。

【回调参数】

streamRecoveredEvent：必选，UserList[]类型，已订阅且视频码流恢复的用户列表。UserList参考**stream-interrupted**中的定义。该事件为1.4.0版本新增。

volume-indicator

【事件说明】

音频TopN模式下，房间中当前音量最大的用户提示事件。

【回调参数】

userVolumeInfos：必选， UserVolumeInfo[]类型。

UserVolumeInfo定义为：{

- user_id：必选，string类型，用户Id。
- volume：可选，number类型，取值范围为[0,100]。

}

⚠ 注意

该事件仅在音频TopN模式下生效，1.5.0版本新增。

remote-user-name-changed

【事件说明】

远端用户昵称变更事件。该事件为1.5.0版本新增。

【回调参数】

userNameChangedEvent：必选， UserNameInfo类型。

UserNameInfo定义为：{

- roomId：必选，string[64]类型，房间Id。
- userId：必选，string[64]类型，用户Id。
- userName：必选， string[256]类型，修改后的用户昵称。

}

live-streaming-updated

【事件说明】

旁路推流状态更新事件。该事件为2.0.1版本新增。

【回调参数】

urlStatus：UrlStatus[]类型，所有CDN推流状态更新。UrlStatus定义如下：

- url：必选，string类型，表示CDN推流URL

- status：必选，number类型，表示当前推流状态。
  - 0：初始化。
  - 1：链接正常且有流。
  - 2：链接正常但无流。
  - 3：异常重试。
  - 4：处理失败。
- errorCode：可选，number类型，表示详细的失败原因，支持的枚举值如下：
  - 0： 正常。
  - 1： 内部错误。
  - 2： 地址解析失败。
  - 3： 连接失败。
  - 4： RTMP握手失败。
  - 5： 内存错误。
  - 6： 参数错误。
  - 7： 重试失败。
  - 8： 响应超时失败。

rtc-stats

【事件说明】

音视频流数据统计事件。该事件为2.0.3版本新增。

【回调参数】

rtcStatsInfo：必选， rtcStatsInfo[]类型。

rtcStatsInfo定义为：{

- userName：必选，string类型，用户昵称。
- isRemote：必选，boolean类型，是否为远端流，true表示远端流，false表示本地流。
- streamType：必选，ContentType类型，流类型。ContentType类型的枚举值如下：{
  - main：string类型，表示主流。
  - middle1：string类型，表示主流，middle1~middle4码流依次降低。
  - middle2：string类型，表示主流，middle1~middle4码流依次降低。
  - middle3：string类型，表示主流，middle1~middle4码流依次降低。
  - middle4：string类型，表示主流，middle1~middle4码流依次降低。
  - slides：string类型，表示主流视频分辨率最小的流。
  - desktop：string类型，表示共享流。

  }
- mediaType：必选，MediaType类型，媒体类型，音频或视频。
- bitrate：必选，number类型，音视频流码率，单位为kbps。
- frameRate：必选，number类型，视频帧率，单位为帧/秒。

- rtt：必选，number类型，表示SDK到边缘服务器的RTT (Round-Trip Time)，单位毫秒。只有本地流才有rtt 值。
- jitter：必选，number类型，音视频流抖动值。
- pktLossRate：必选，number类型，音视频流丢包率。

}

3.5.4 流对象（Stream）

本章节介绍了Web SDK的Stream接口详情。

表 3-7 Stream 接口

接口	描述
play	播放该音视频流。
stop	停止播放视频流。
resume	恢复播放音视频。
close	关闭音视频。
muteAudio	禁用音频轨道。
muteVideo	禁用视频轨道。
unmuteAudio	启用音频轨道。
unmuteVideo	启用视频轨道。
getId	获取Stream唯一标识ID。
getUserId	获取Stream所属的用户ID。
setAudioOutput	设置音频输出设备。
setAudioVolume	设置音频音量大小。
getAudioLevel	获取实时音频音量级别。
hasAudio	流是否包含音频轨道。
hasVideo	流是否包含视频轨道。
getAudioTrack	获取流的音频轨道。
getVideoTrack	获取流的视频轨道。
getType	获取流类型。
on	注册流对象事件回调接口。
off	取消注册流对象事件回调接口。
getStreamInfo	获取流信息。

play

```
async play(elementId: string, options?: Options): Promise<void>
```

【功能说明】

播放音视频流。该方法会自动创建<audio>、<video>标签，并在指定的标签上播放音频和视频，同时该标签会被添加到页面中名为 “elementId ”的div容器下。

【请求参数】

- elementId：必选，string类型，HTML 标签ID。
- options：可选，Options类型，播放选项。

  Options类型定义为：{
  - objectFit：可选，string类型，远端共享流的默认值为contain，其他流默认值为cover。支持的枚举值如下：
    - contain：优先保证视频内容全部显示。视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
    - cover：优先保证视窗被填满。视频尺寸等比缩放，直至整个视窗被视频填满。如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
    - fill：视频内容完全填充视窗。如果视频的宽高比与视窗不相匹配，那么视频将被拉伸以适应视窗。
  - muted：可选，boolean类型，true表示静音，false表示不静音。默认值为false。
  - resolutionId：可选，string类型。双流场景，指定播放的分辨率Id的视频，如果不指定，默认选择分辨率最高的视频。该参数为1.8.0版本新增。
  - }

【返回参数】

Promise<void>：返回一个Promise对象。

⚠ 注意

- 由于浏览器自动播放策略的限制，在play()返回错误后需要引导用户通过手动触发页面控件后，调用**resume**接口恢复播放。
- 本地流播放通常需要设置muted参数为true（静音），以防播放出来的声音也被麦克风采集到，造成回音的效果。
- 在App上，一个resolution对应于一个音视频播放窗口，Stream中的音频对所有的resolution是公共的。

stop

```
stop(option?: StopOption): void
```

【功能说明】

停止播放音视频流。

【 请求参数 】

option：可选，StopOption类型，表示停止播放的选项。如果不传，则停止该流的音频和所有分辨率的视频。

StopOption类型定义如下：{

- audio：可选，boolean类型。表示是否停止音频。默认为false。
- video：可选，boolean类型。表示是否停止视频。默认为false。
- resolutionIds：可选，string[] 型。在video为true的时候有效。指定停止播放的分辨率Id 的视频，如果不传resolutionIds，则默认停止所有分辨率的视频。

}

【 返回参数 】

无

resume

```
async resume(option?: ResumeOption): Promise<void>
```

【 功能说明 】

恢复播放音视频。场景说明如下：

- 在某些版本浏览器上，移动传入play()的div容器可能会导致音视频播放器进入PAUSED状态，此时需要调用该接口恢复播放。
- 由于浏览器自动播放策略的限制，在play()返回错误后需要引导用户通过手动方式调用该接口恢复播放。

【 请求参数 】

option：可选，表示恢复播放的选项，ResumeOption类型，如果不传则恢复播放该流的音频和所有分辨率的视频。

ResumeOption类型定义为：{

- audio：可选，boolean类型。是否resume音频。默认为false。
- video：可选，boolean类型。是否resume视频。默认为false。
- resolutionIds：可选，string[]类型，在video为true的时候有效。指定恢复播放的分辨率Id的视频，如果不传resolutionIds，默认恢复所有分辨率的视频。

}

【 返回参数 】

Promise<void>：返回一个Promise对象。

close

```
close(option?: CloseOption): void
```

【 功能说明 】

关闭音视频的播放。对于本地流，该方法在关闭播放的同时会关闭音视频采集，释放设备资源占用。

【 请求参数 】

option：可选，CloseOption类型，表示关闭音视频的选项。如果option不填写，则关闭音频和所有分辨率的视频。

CloseOption类型定义如下：{

- audio：可选，boolean类型。表示是否关闭音频。默认为false。
- video：可选，boolean类型。表示是否关闭视频。默认为false。
- resolutionIds：可选，string[]类型。在video为true的时候有效。指定关闭播放的分辨率Id的视频，如果不传resolutionIds，默认关闭所有分辨率的视频。

}

【 返回参数 】

无

muteAudio

```
muteAudio(): boolean
```

【 功能说明 】

禁用音频轨道。

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示禁用音频轨道成功，false表示禁用音频轨道失败。

muteVideo

```
muteVideo(): boolean
```

【 功能说明 】

禁用视频轨道。

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示禁用视频轨道成功，false表示禁用视频轨道失败。

unmuteAudio

```
unmuteAudio(): boolean
```

【 功能说明 】

启用音频轨道。

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示启用音频轨道成功，false表示启用音频轨道失败。

unmuteVideo

```
unmuteVideo(): boolean
```

【 功能说明 】

启用视频轨道。

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示启用视频轨道成功，false表示启用视频轨道失败。

getId

```
getId(): string
```

【 功能说明 】

获取Stream唯一标识ID。发布流中需包含视频流，发布后才能获取到有效的ID。

【 请求参数 】

无

【 返回参数 】

string类型，Stream唯一标识ID。

getUserId

```
getUserId(): string
```

【 功能说明 】

获取Stream所属的用户ID。对于本地流，如果在**createStream**的时候入参StreamConfig中没有设置该参数，则返回undefined。

【 请求参数 】

无

【 返回参数 】

string类型，Stream所属的用户ID。

setAudioOutput

```
setAudioOutput(deviceId: string): Promise<void>
```

【 功能说明 】

设置音频输出设备。

【 请求参数 】

deviceId：必选，string类型，音频输出设备的设备ID。

【 返回参数 】

Promise<void>：返回一个Promise对象。

setAudioVolume

```
setAudioVolume(volume: number): void
```

【 功能说明 】

设置音频音量大小。

【 请求参数 】

volume：必选，number类型，音量大小，取值范围为[0,100]。

【 返回参数 】

无

getAudioLevel

```
getAudioLevel(): number
```

【 功能说明 】

获取实时音量级别。

【 请求参数 】

无

【 返回参数 】

number类型，返回值在(0, 1)之间，通常该值大于0.1认为用户正在说话。

hasAudio

```
hasAudio(): boolean
```

【 功能说明 】

该Stream是否包含音频轨道。

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示包含音频轨道，false表示不包含音频轨道。

hasVideo

```
hasVideo(): boolean
```

【 功能说明 】

该Stream 是否包含视频轨道

【 请求参数 】

无

【 返回参数 】

boolean类型，true表示包含视频轨道，false表示不包含视频轨道。

getAudioTrack

```
getAudioTrack(): MediaStreamTrack
```

【功能说明】

获取音频轨道

【请求参数】

无

【返回参数】

MediaStreamTrack类型。MediaStreamTrack类型详情可参见**MediaStreamTrack**。

getVideoTrack

```
getVideoTrack(resolutionId?:string): MediaStreamTrack
```

【功能说明】

获取视频轨道。

【请求参数】

resolutionId：可选，string类型。指定分辨率Id，如果不指定，默认选择分辨率最高的视频。

【返回参数】

MediaStreamTrack 类型。MediaStreamTrack类型详情可参见**MediaStreamTrack**。

getType

```
getType(): string
```

【功能说明】

获取流类型。用于判断一个流是主流还是辅流，辅流通常是一个屏幕共享流。

【请求参数】

无

【返回参数】

string类型，本地流：local，远端主流：main，远端辅流：auxiliary。

on

```
on(event: string, handler: function): void
```

【功能说明】

注册流对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。详细事件列表请参见**RTCStreamEvent**。
- handler：必选，function类型，事件处理方法。

【返回参数】

无

off

```
off(event: string, handler: function): void
```

【功能说明】

取消注册流对象事件回调接口。

【请求参数】

- event：必选，string类型，事件名称。详细事件列表请参见**RTCStreamEvent**。
- handler：必选，function类型，事件处理方法。

【返回参数】

无

getStreamInfo

```
getStreamInfo(): StreamInfo
```

【功能说明】

获取已经初始化的本地流，或者收到的远端流的的信息。

【请求参数】

无

【返回参数】

StreamInfo 类型定义如下：{

- videoProfiles：RTCVideoProfileInfo[]类型。
- audioProfile：RTCAudioProfile类型。

}

RTCVideoProfileInfo类型定义如下：{

- resolutionId：string类型，表示分辨率Id。
- hasTrack：boolean类型，表示该分辨率的视频是否具备可播放的track。
- width：number类型，分辨率的宽度，单位为像素。
- height：number类型，分辨率的高度，单位为像素。
- frameRate：number类型，视频帧率，单位为帧/秒。
- minBitrate：number类型，视频最小码率，单位为bps。
- maxBitrate：number类型，视频最大码率，单位为bps。

}

RTCAudioProfile类型定义如下：{

- sampleRate：number类型，表示音频采样率。
- channelCount：number类型，表示音频声道数。
- bitrate：number类型，表示音频码率，单位为bps。

```
}
```

3.5.5 本地流对象（LocalStream）

该对象继承自Stream对象，并有如下新增接口。

表 3-8 LocalStream 接口

接口	描述
initialize	本地流初始化。
setAudioProfile	设置音频流配置。
setVideoProfile	设置视频流配置。
setScreenProfile	设置辅流配置。
addAudioTrackCapture	流初始化后，流对象中没有音频track的，可通过该接口增加音频track采集。该接口为1.10.0版本新增。
addVideoTrackCapture	流初始化后，流对象中没有视频track的，通过该接口可增加视频track采集。该接口为1.10.0版本新增。
addResolution	对初始化后的本地流增加新的分辨率的视频。
removeResolution	对流删除视频指定分辨率的视频。
addTrack	为本地流对象添加音视频轨。
removeTrack	从本地流对象移除音视频轨。
replaceTrack	为本地流对象替换音视频轨。
switchDevice	切换媒体输入设备。
startAudioMixing	开始播放在线音频文件。
stopAudioMixing	停止播放在线音频文件。
pauseAudioMixing	暂停播放在线音频文件。
resumeAudioMixing	恢复播放在线音频文件。
getAudioMixingDuration	获取在线音频文件时长。
setAudioMixingVolume	设置在线音频音量大小。
setAudioMixingPosition	设置在线音频播放进度。
getAudioMixingCurrentPosition	获取在线音频播放进度。
bindScreenAudio2RelatedStream	绑定屏幕共享背景音至关联流对象。该接口为1.4.0版本新增。

initialize

```
initialize(): Promise<StreamInitializeResult>
```

【功能说明】

根据**createStream**入参StreamConfig，初始化本地音视频流对象。流只有初始化完成后才可以播放。

【请求参数】

无

【返回参数】

流初始化结果：StreamInitializeResult类型。

⚠ 注意

StreamInitializeResult表示流初始化的结果。对象继承自RtcError。提供如下方法：

- 获取所有媒体类型的初始化结果：
  ```
  getMediaCaptureResult(): MediaCaptureResult[]
  ```
- 获取指定媒体类型的初始化结果：
  ```
  getMediaCaptureResultByType(type: MediaType): MediaCaptureResult
  ```

MediaCaptureResult类型定义如下：

{

- type：必选，表示媒体类型，MediaType类型，MediaType枚举值包括audio或者video。
- track：可选，表示对应媒体类型如果初始化成功后的 track，MediaStreamTrack类型。
- error：可选，表示对应媒体类型如果初始化失败的错误，RtcError类型。

}

如果初始化音视频采集失败，则结果中会返回对应的错误信息。错误码范围90000001，90100017~90100020，具体请参考**表3-13**。

setAudioProfile

```
setAudioProfile(profile: string|RTCAudioProfile): void
```

【功能说明】

设置音频流采样率、声道和码率等配置。如果未调用该接口设置，则SDK设置为默认值standard。

【请求参数】

profile：必选，string类型或RTCAudioProfile类型。 若为string类型，则相关的采样率、声道数、码率如**表3-9**所示。 若为RTCAudioProfile类型，则需要根据实际情况进行配置，推荐使用已定义的profile，输入不合法时，默认使用standard。

表 3-9 profile 对应的采样率、声道数和码率

profile	采样率(千赫兹/KHZ)	声道数	码率（Kbps）
low	16	1	24

profile	采样率(千赫兹/KHZ)	声道数	码率（Kbps）
standard	48	1	40
high	48	1	128

RTCAudioProfile类型定义为：{

- sampleRate：可选，number类型，采样率。
- channelCount：可选，number类型，音轨数。
- bitrate：可选，number类型，码率，单位为bps。

}

【返回参数】

无

注意

该方法需要在调用**initialize**之前调用。

需注意码表中单位为kbps，接口传参的单位为bps，实际调用接口时需进行转换。

setVideoProfile

```
setVideoProfile(profile: string|RTCVideoProfile,resolutionId?: string): void
```

【功能说明】

设置视频主流配置，如分辨率、帧率和码率等。如果未调用该接口设置，则SDK设置默认值为360p_2。如果该流已经发布，则该流会自动重新发布到远端。

【请求参数】

- profile：必选，string类型或RTCVideoProfile类型。 若为string类型，则相关的分辨率、帧率和码率如**表3-10**。 若为RTCVideoProfile类型，则需要根据实际情况进行配置，推荐使用已定义的“profile”。输入不合法时，默认使用360p_2。

表 3-10 profile 对应的分辨率、帧率和码率

profile	分辨率	帧率	最小码率（kbps）	最大码率（kbps）
90p_1	160 x 90	15	64	110
90p_2	120 x 90	15	64	110
120p_1	160 x 120	15	64	120
120p_2	120 x 120	15	64	110
180p_1	320 x 180	15	80	320

profile	分辨率	帧率	最小码率（kbps）	最大码率（kbps）
180p_2	240 x 180	15	80	170
180p_3	180 x 180	15	64	130
240p_1	320 x 240	15	100	400
240p_2	240 x 240	15	80	320
270p_1	480 x 270	15	160	600
300p_1	400 x 300	15	200	500
360p_1	640 x 360	15	200	800
360p_2	480 x 360	15	200	700
360p_3	360 x 360	15	150	600
450p_1	800 x 450	15	300	950
480p_1	640 x 480	15	250	900
480p_2	480 x 480	15	200	800
540p_1	960 x 540	15	400	1000
630p_1	1120 x 630	15	450	1150
720p_1	1280 x 720	15	500	1500
720p_2	960 x 720	15	450	1100
1080p_1	1920 x 1080	15	600	2000
1080p_2	1440 x 1080	15	550	1700

- resolutionId：可选，string类型。在双流场景下，指定要设置的分辨率Id的视频，如果不指定，默认选择最高的分辨率的视频。

RTCVideoProfile类型定义为：{

- width：可选，number类型，分辨率的宽度，单位为像素。
- height：可选，number类型，分辨率的高度，单位为像素。
- frameRate：可选，number类型，视频帧率，单位为帧/秒。
- minBitrate：可选，number类型，视频最小码率，单位为bps。
- maxBitrate：可选，number类型，视频最大码率，单位为bps。

}

【返回参数】

无

⚠ 注意

- 需注意码表中单位为kbps，接口传参的单位为bps，实际调用接口时需进行转换。
- 自定义采集的流（使用videoSource创建的Stream）不支持动态调用此接口，仅支持摄像头采集的流调用。
- 由于设备采集能力、系统性能以及浏览器的限制，视频分辨率，帧率，码率的实际值不一定能够完全匹配设定值，这种情况下浏览器会自动调整分辨率，尽可能匹配设定值，具体分辨率以实际采集到的分辨率为准。
- 能否采集1080p及以上的分辨率取决于采集设备能力以及系统性能，iOS Safari不支持采集1080p。
- 码表中的1080p分辨率为2.0.0版本新增，1.0版本如需采集1080P分辨率，请使用自定义参数的方式设置。
- Firefox不支持自定义帧率（默认为30fps）。
- 如果上表的Profile不符合您的要求，用户可以根据自身业务需求自定义Profile。

setScreenProfile

```
setScreenProfile(profile: string|RTCScreenProfile): void
```

【功能说明】

设置辅流配置，包括分辨率、帧率和码率等。如果未调用该接口设置，则SDK设置默认值为720p。

【请求参数】

profile：必选，string类型或RTCScreenProfile类型。若为string类型，则相关分辨率、帧率、码率如表3-11所示。如果为RTCScreenProfile，则需要根据实际情况进行配置，推荐使用已定义的“profile”。输入不合法时，默认使用720p。

表 3-11 profile 对应的分辨率、帧率和码率

profile	分辨率	帧率	码率（kbps）
720p	1280 x 720	15	1200
1080p	1920 x 1080	15	2000

RTCScreenProfile类型为: {

- width：可选，number类型，分辨率的宽度，单位为像素。
- height：可选，number类型，分辨率的高度，单位为像素。
- frameRate：可选，number类型，视频帧率，单位为帧/秒。
- bitrate：可选，number类型，视频码率，单位为bps。

}

【返回参数】

无

⚠ 注意

该方法需要在调用**initialize**之前调用。

需注意码表中单位为kbps，接口传参的单位为bps，实际调用接口时需进行转换。

addAudioTrackCapture

```
addAudioTrackCapture(microphoneId?: string): promise<MediaStreamTrack>
```

【功能说明】

已经初始化的本地流对象，如果未初始化音频或者音频初始化失败，可以调用该接口对本地流增加音频的track采集。若该本地流已经被发布，则该流会自动重新发布到远端。该接口为1.10.0版本新增。

【请求参数】

microphoneId：可选，string类型，指定采集的麦克风设备Id。如果不传，SDK采用**createStream**入参StreamConfig中指定的microphoneId。

【返回参数】

MediaStreamTrack类型。表示增加成功的track。

⚠ 注意

在通过**removeTrack**移除音频的track后，可以通过该接口重新添加音频的track。

addVideoTrackCapture

```
addVideoTrackCapture(option?: VideoCaptureOption): promise<MediaStreamTrack>
```

【功能说明】

已经初始化的本地流对象，如果未初始化视频或者视频初始化失败，可以调用该接口对本地流增加视频的track采集。若该本地流已经被发布，则该流会自动重新发布到远端。该接口为1.10.0版本新增。

【请求参数】

option：可选，指定采集参数。VideoCaptureOption类型。

VideoCaptureOption定义如下：{

- cameraId：可选，string类型。指定摄像头设备Id。对于Android设备：user表示前置摄像头，environment表示后置摄像头。如果不传，SDK采用**createStream**入参StreamConfig中指定的cameraId和facingMode 。
- resolutionId：可选，string类型。指定要设置的分辨率Id的视频。如果不传，SDK默认选择分辨率最高的视频。

}

【返回参数】

MediaStreamTrack类型。表示增加成功的track。

⚠ 注意

在通过**removeTrack**移除视频的track后，可以通过该接口重新添加视频的track。

addResolution

```
addResolution(profile: string|RTCVideoProfile,audio?: boolean): promise<RTCVideoProfileInfo>
```

【 功能说明 】

为已经初始化的本地流对象添加指定分辨率的视频。若该本地流已经被发布，则该流会自动重新发布到远端。

【 请求参数 】

- profile：必选，string类型，表示添加的分辨率视频的参数信息。RTCVideoProfile类型说明参考**setVideoProfile**部分的说明。
- audio：可选，boolean类型，表示是否创建音频，true表示创建，false表示不创建。Stream中的音频对该stream中的所有的resolution是公共的。如果创建流的配置开启了音频但是没有音频track，则默认创建。

【 返回参数 】

RTCVideoProfileInfo类型。增加成功后resolution的profile信息。RTCVideoProfileInfo定义参考**getStreamInfo**接口定义。

如果失败，则返回StreamInitializeResult。StreamInitializeResult参考**initialize**接口的定义。

⚠ 注意

- 当前对一个LocalStream最多可支持2个分辨率。
- 当开启大小流的场景下，流的分辨率设置接口**setVideoProfile**使用会存在限制（两条流的分辨率大者，决定了可调整的最大分辨率上限），建议谨慎使用。

removeResolution

```
removeResolution(resolutionId: string): promise<void>
```

【 功能说明 】

对本地流对象删除视频分辨率。若该本地流已经被发布，则该流会自动重新发布到远端。

【 请求参数 】

resolutionId：必选。string类型，指定要删除的视频的分辨率Id。

【 返回参数 】

Promise<void>：返回一个Promise对象。

addTrack

```
addTrack(track: MediaStreamTrack,resolutionId?: string ): promise<void>
```

【 功能说明 】

为初始化后的本地流对象添加音视频轨。若该本地流已经被发布，则该流会自动重新发布到远端。

【 请求参数 】

- track：必选，MediaStreamTrack类型。指定要添加的track。
- resolutionId：可选，string类型。在双流主流场景下，指定分辨率Id的视频，如果不指定，默认选择分辨率最高的视频增加 track。

【 返回参数 】

Promise<void>：返回一个Promise对象。

⚠ 注意

- 如果分辨率的视频track已经存在，则不能重复添加。
- 如果需要更新旁路推流等操作，需要等待异步操作完成

removeTrack

```
removeTrack(track: MediaStreamTrack): promise<void>
```

【 功能说明 】

从本地流对象移除音视频轨。若该本地流已经被发布，则该流会自动重新发布到远端。

【 请求参数 】

track：必选，MediaStreamTrack类型。指定要移除的track。

【 返回参数 】

Promise<void>：返回一个Promise对象。

⚠ 注意

- 如果麦克风对应的音频track都移除，则SDK不会再访问该麦克风。
- 如果摄像头对应的视频track都移除，则SDK不会再访问该摄像头。
- 如果需要更新旁路推流等操作，需要等待异步操作完成

replaceTrack

```
replaceTrack(track: MediaStreamTrack, type: "audio" | "video",resolutionId?: string): promise<void>
```

【 功能说明 】

为初始化后的本地流对象替换音视频轨。若该本地流已经被发布，则该流会自动重新发布到远端。

【 请求参数 】

- track：必选，MediaStreamTrack类型。
- type：必选，string类型，"audio" | "video"。
- resolutionId：可选，string类型。type为video的时候有效。指定要替换的分辨率Id的视频，如果不指定，默认选择分辨率最高的视频。

【 返回参数 】

Promise<void>：返回一个Promise对象。

switchDevice

```
switchDevice(deviceType: "audio" | "video", deviceId: string): Promise<void>
```

【 功能说明 】

切换媒体输入设备。若该本地流已经被发布，则会自动更新发布到远端的音视频流。

【 请求参数 】

- deviceType：必选，string类型，"audio" | "video"。
- deviceId：必选，string类型， 输入设备的设备ID。

【 返回参数 】

Promise<void>：返回一个Promise对象。

startAudioMixing

```
startAudioMixing(option: AudioMixOption): Promise<void>
```

【 功能说明 】

开始播放在线音频文件。

【 请求参数 】

option：必选，音频文件播放参数。参数类型为AudioMixOption。

AudioMixOption定义为：{

- filePath：必选，string类型，表示在线音频文件的下载路径。支持音频格式包括MP3、AAC以及浏览器支持的其他音频格式。
- startTime：可选，number类型，表示音频文件开始播放的时间点，默认值为0。
- replace：可选，boolean类型，如果为true表示要用音频文件替换本地音频流，默认值为false。
- loop：可选，boolean类型，如果为true表示需要无限循环播放，默认值为false。
- repeatCount：可选，number类型，循环播放次数。如果为0，表示不重复播放，默认值为0。

}

【 返回参数 】

Promise<void>：返回一个Promise对象。

⚠ 注意

- loop设置为true时，repeatCount需要设置为0。
- 混音功能相关接口仅在**publish**接口调用成功后方可调用。

stopAudioMixing

```
stopAudioMixing(): Promise<void>
```

【功能说明】

停止播放在线音频文件。

【请求参数】

无

【返回参数】

Promise<void>：返回一个Promise对象。

pauseAudioMixing

```
pauseAudioMixing(): void
```

【功能说明】

暂停播放在线音频文件。

【请求参数】

无

【返回参数】

无

resumeAudioMixing

```
resumeAudioMixing(): void
```

【功能说明】

恢复播放在线音频文件。

【请求参数】

无

【返回参数】

无

getAudioMixingDuration

```
getAudioMixingDuration(): number
```

【功能说明】

获取在线音频文件时长。单位为毫秒。

【请求参数】

无

【 返回参数 】

number类型，音频文件时长。

setAudioMixingVolume

```
setAudioMixingVolume(level: number): void
```

【 功能说明 】

设置在线音频音量大小。

【 请求参数 】

level：必选，number类型，表示音量大小。

【 返回参数 】

无

setAudioMixingPosition

```
setAudioMixingPosition(position: number): void
```

【 功能说明 】

设置在线音频播放进度。

【 请求参数 】

position：必选，number类型，表示进度，取值不能大于音频实际时长。单位为毫秒。

【 返回参数 】

无

getAudioMixingCurrentPosition

```
getAudioMixingCurrentPosition(): number
```

【 功能说明 】

获取在线音频播放进度。

【 请求参数 】

无

【 返回参数 】

number类型，当前音频播放进度。单位为毫秒。

bindScreenAudio2RelatedStream

```
bindScreenAudio2RelatedStream(bindStream: LocalStream, muteMainStreamAudio?: boolean): void
```

【 功能说明 】

绑定屏幕共享背景音至关联流对象。

【 请求参数 】

- bindStream：必选，LocalStream类型, 创建并初始化成功的本地主流对象。
- muteMainStreamAudio：可选，boolean类型, 主流音频是否静音，默认值为false。true表示主流音频静音，false表示主流音频不静音。

【返回参数】

无

⚠ 注意

- 该接口仅由通过**HRTC.createStream**创建的辅流对象调用。
- 屏幕共享背景音需要借助主流的音频通道发送，若想要订阅共享背景音，至少需要订阅主流音频。
- 弹出的共享选择窗口，需要勾选左下角的共享音频复选框，否则屏幕共享背景音将无法共享。
- 该功能仅支持Windows平台Chrome浏览器74及以上版本。

3.5.6 远端流对象（RemoteStream）

该对象继承自Stream对象。

3.5.7 流事件通知（RTCStreamEvent）

本章节介绍了Web SDK的RTCStreamEvent事件。

表 3-12 StreamEvent 事件

接口	描述
player-state-change	播放状态变更事件。
screen-sharing-stopped	共享屏幕停止事件。
audio-mixing-played	本地混音播放事件。
audio-mixing-finished	本地混音播放结束事件。

⚠ 注意

事件注册监听应在业务结束时取消注册，否则注册监听事件累积会有内存泄漏风险。

player-state-change

【事件说明】

播放状态变更事件。在播放状态变更的时候触发。

【回调参数】

event：playState类型。字段定义如下：

- type：string类型，表示播放器类型，取值为video/audio。
- id：string类型，表示流分辨率Id。
- state：string类型，表示当前播放状态。取值包括：PLAYING，STOPPED，PAUSED， NONE。
- reason：string类型，表示触发播放状态变更的原因。

screen-sharing-stopped

【事件说明】

共享屏幕停止事件。仅在本地共享屏幕停止时触发。

【回调参数】

event：string类型。表示停止共享屏幕时的流Id。

audio-mixing-played

【事件说明】

混音播放事件。仅在本地混音播放时触发。

【回调参数】

无

audio-mixing-finished

【事件说明】

混音播放结束事件。仅在本地混音结束播放时触发。

注意：手动调用**stopAudioMixing**和**pauseAudioMixing**不会触发此事件。

【回调参数】

无

3.5.8 错误码（RtcError）

getCode

```
getCode(): number
```

【功能说明】

获取错误码。

【请求参数】

无

【返回参数】

number类型，错误码值。

getMsg

```
getMsg(): string
```

【 功能说明 】

获取错误描述。

【 请求参数 】

无

【 返回参数 】

string类型，错误码描述。

3.5.9 客户端错误码

本章节介绍了Web SDK的客户端错误码RtcErrorCode的详细信息。

表 3-13 错误码说明

类成员	错误码	描述	错误原因或建议处理方式
RTC_ERR_CODE_SUCCESS	0	success	成功。
RTC_ERR_CODE_RTC_SDK_ERROR	90000001	sdk internal error	SDK内部错误，请联系技术支持。
RTC_ERR_CODE_WAIT_RSP_TIMEOUT	90000004	message response timeout	消息响应超时，请联系技术支持。
RTC_ERR_CODE_INVALID_PARAMETER	90000005	invalid parameter	参数传递错误，请参照API文档排查。
RTC_ERR_CODE_INVALID_OPERATION	90100001	illegal operation	非法操作，用户状态不正确。
RTC_ERR_CODE_NOT_SUPPORT_MEDIA_DEVICES	90100002	not support enumerate devices	浏览器不支持enumerateDevices方法。
RTC_ERR_CODE_NO_AVAILABLE_DEVICES	90100003	no available devices	没有找到可用设备，请排查设备是否就绪。
RTC_ERR_CODE_NO_AVAILABLE_VIDEO_INPUT_DEVICES	90100004	no available video input devices	没有找到可用摄像头设备，请排查视频采集设备是否就绪。
RTC_ERR_CODE_NO_AVAILABLE_AUDIO_INPUT_DEVICES	90100005	no available audio input devices	没有找到音频输入设备，请排查音频采集设备是否就绪。
RTC_ERR_CODE_NO_AVAILABLE_AUDIO_OUTPUT_DEVICES	90100006	no available audio output devices	没有找到音频输出设备。
RTC_ERR_CODE_STATUS_ERROR	90100007	room status error	房间状态不正确，检查是否入会成功。

类成员	错误码	描述	错误原因或建议处理方式
RTC_ERR_CODE_WEBSOCKET_NOT_CONNECTED	90100008	websocket connection state is not "CONNECTED"	websocket 链接未成功，检查链接情况。
RTC_ERR_CODE_WAIT_CONFIG_FAIL	90100009	wait server config fail	获取下发配置失败，请联系技术支持。
RTC_ERR_CODE_PUBLISH_RESPONSE_FAIL	90100010	publish response fail	发布流响应失败，请联系技术支持。
RTC_ERR_CODE_REGION_NOT_COVERED	90100011	current region is not covered, service unavailable	没有找到服务端地址，请联系技术支持。
RTC_ERR_CODE_WEBSOCKET_CONNECT_TIMEOUT	90100012	websocket connect timeout	websocket建链超时，请联系技术支持。
RTC_ERR_CODE_WEBSOCKET_RECONNECT_TIMEOUT	90100013	websocket reconnect timeout	websocket重连超时，请联系技术支持。
RTC_ERR_CODE_WEBSOCKET_NOT_OPEN	90100014	websocket is not open	websocket链接未打开，请联系技术支持。
RTC_ERR_CODE_WEBSOCKET_INTERRUPTED	90100015	websocket connection state is idle, interrupt operation	websocket链接被强制关闭，一般为离会或者重连。
RTC_ERR_CODE_WEBSOCKET_CONNECT_ERROR	90100016	websocket connect error	websocket监听onerror，服务端主动断链。
RTC_ERR_CODE_CAPTURE_PERMISSION_DENIED	90100017	capture failed, permission denied	采集失败，音视频设备采集权限未被授权。建议提示用户授权摄像头/麦克风访问权限。
RTC_ERR_CODE_CAPTURE_OVER_CONSTRAINED	90100018	capture failed, Constraint parameter invalid	采集失败，音视频采集设备不支持设置的采集约束。
RTC_ERR_CODE_CAPTURE_DEVICE_NOT_FOUND	90100019	capture failed, requested device not found	采集失败，设备未找到。建议在通话开始前引导用户检查通话所需的摄像头或麦克风等设备是够就绪。

类成员	错误码	描述	错误原因或建议处理方式
RTC_ERR_CODE_CAPTURE_DEVICE_NOT_READABLE	90100020	capture failed, maybe device is occupied by other application	采集失败，设备被占用，请检查使用状态。建议提示用户“暂时无法访问摄像头/麦克风，请确保当前没有其他应用请求访问摄像头/麦克风，并重试”。
RTC_ERR_CODE_PLAY_NOT_ALLOW	90100021	the user didn't interact with the document first, please trigger by gesture	不允许播放。
RTC_ERR_CODE_ROLE_NO_PERMISSION	90100022	the user role have no permission to operate	用户角色没有权限，请检查用户角色。
RTC_ERR_CODE_ANSWER_SDP_INVALID	90100023	the answer sdp is invalid	SDP协商错误，请联系技术支持。
RTC_ERR_CODE_MEDIA_UPSTREAM_UNSUPPORTED	90100024	the upstream media is not supported	浏览器不支持媒体采集。
RTC_ERR_CODE_MEDIA_NETWORK_ERROR	90100026	media connection establish failed, please switch network or try again later	媒体建链失败，请切换网络重试。
RTC_ERR_CODE_CLIENT_RELAY_ROOM_OVER_MAXNUM	90100027	relay room number over maxium number	跨房数量超过最大值。
RTC_ERR_CODE_CLIENT_RELAY_JOINER_OVER_MAXNUM	90100028	joiner already exist in relay rooms	跨房内主播人数超过最大值。
RTC_ERR_CODE_ROOM_STREAM_STATUS_PAUSED	90100029	room stream status paused	房间音视频暂停。
RTC_ERR_CODE_SIGNATURE_EXPIRED	90100030	signature expired	签名过期。
RTC_ERR_CODE_SIGNATURE_INVALID	90100031	signature invalid	签名非法。

类成员	错误码	描述	错误原因或建议处理方式
RTC_ERR_CODE_RTC_ACS	90100100	server internal exception	服务端内部错误，请联系技术支持。
RTC_ERR_CODE_RTC_CONTROL_ERROR	90100200	server internal exception	服务端内部错误，请联系技术支持。
RTC_ERR_CODE_SFU_ERROR	90100600	server internal exception	服务端内部错误，请联系技术支持。

3.5.10 服务端错误码

当SDK运行出现网络、媒体相关等错误时，SDK无法自动恢复，需要APP干预或进行用户提示。该错误码由服务端产生，通过onError返回。

表 3-14 服务端错误码

错误码	描述	错误原因
RTC.10000001	内部错误	程序或环境问题
RTC.31000000	节点不存在	程序或环境问题
RTC.31000001	session校验失败	程序或环境问题
RTC.31000003	内部异常	程序或环境问题
RTC.31000004	认证失败	用户使用问题
RTC.31000005	请重试	用户使用问题
RTC.31000006	需要时钟同步	用户使用问题
RTC.31000007	请求资源不存在	程序或环境问题
RTC.32000000	服务异常	程序或环境问题
RTC.32000001	流号已满	程序或环境问题
RTC.32000002	SFU为空	程序或环境问题
RTC.32000004	下发流信息失败	程序或环境问题
RTC.32000005	添加适配器失败	程序或环境问题
RTC.32000006	添加路由失败	程序或环境问题
RTC.32000007	获取用户信息异常	程序或环境问题
RTC.32000010	选看用户不存在	程序或环境问题
RTC.32000011	音频速率参数非法	程序或环境问题
RTC.32000012	用户列表为空	程序或环境问题
RTC.32000013	非法请求参数	程序或环境问题

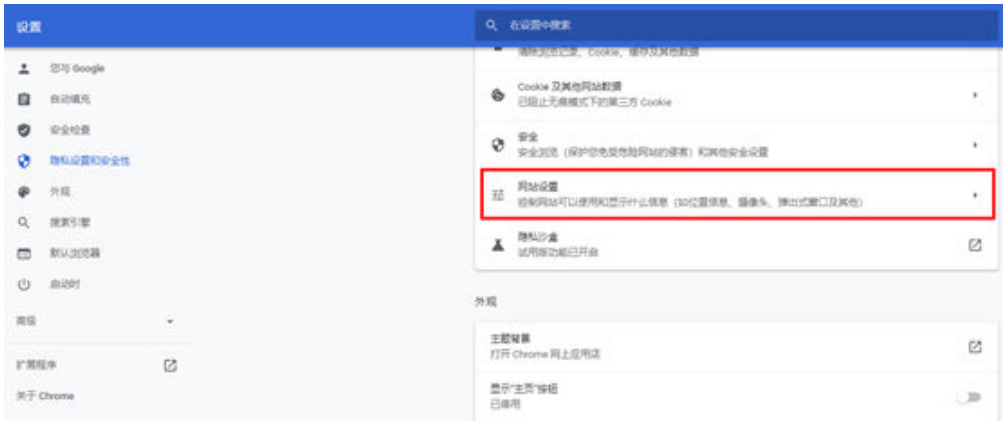
错误码	描述	错误原因
RTC.32000015	内部调用异常	程序或环境问题
RTC.32000016	内部调用异常	程序或环境问题
RTC.32000017	站点不存在	程序或环境问题
RTC.32000018	错误的加密算法	程序或环境问题
RTC.32000019	客户端媒体加密密钥base64解码失败	程序或环境问题
RTC.32000020	生成媒体加密密钥失败	程序或环境问题
RTC.32000021	下发加密信息异常	程序或环境问题
RTC.32000022	获取SFU的IP失败	程序或环境问题
RTC.32000024	内部调用异常	程序或环境问题
RTC.32000025	内部调用异常	程序或环境问题
RTC.32000028	不支持的操作	程序或环境问题
RTC.32000030	sfu资源不足	程序或环境问题
RTC.32000032	跨房数量超过上限	用户使用问题
RTC.32000033	不允许重复跨入同一房间	用户使用问题
RTC.32000034	稍后重试	程序或环境问题
RTC.33000000	服务异常	程序或环境问题
RTC.33000001	节点不存在	程序或环境问题
RTC.34000001	房间已满	用户使用问题
RTC.34000002	房间不存在	程序或环境问题
RTC.34000003	站点不存在	程序或环境问题
RTC.34000004	内部调用异常	程序或环境问题
RTC.34000006	用户不存在	用户使用问题
RTC.34000007	已存在辅流共享	用户使用问题

3.5.11 授权浏览器摄像头/麦克风访问权限的方法

谷歌浏览器

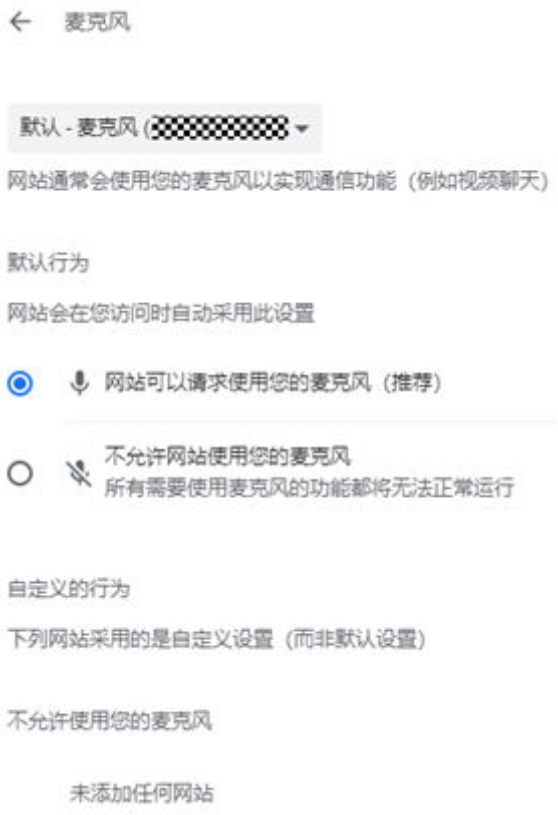
1. 打开谷歌浏览器，单击右上角设置图标。

2. 单击“设置”，打开设置页面。选择“隐私设置和安全性”，再单击“网站设置”。



3. 进入网站设置页面，分别单击“摄像头”和“麦克风”。

4. 进入摄像头、麦克风授权页面，选择网站可以请求使用您的摄像头、麦克风权限即可。

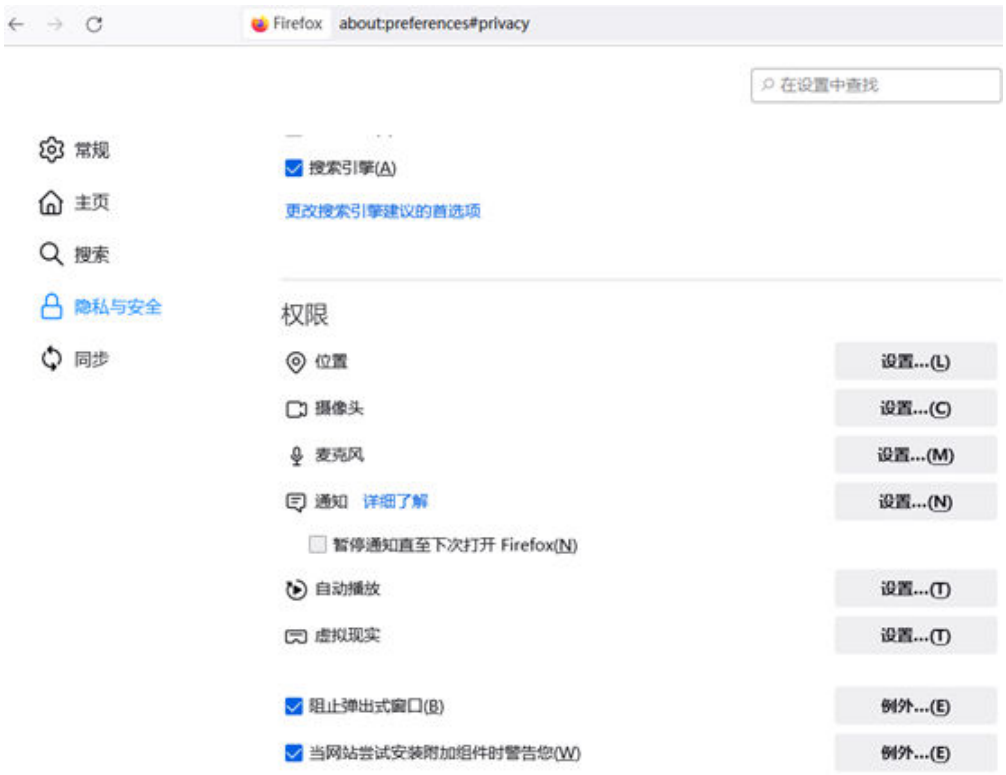


5. 完成以上设置后，浏览器在需要使用摄像头、麦克风时，就会在页面弹出设备询问框，选择“允许”即可。

火狐浏览器

1. 打开火狐浏览器，单击右上角设置图标，单击“设置”。

2. 打开设置页面，单击“隐私与安全”，找到摄像头、麦克风权限。分别单击“摄像头”和“麦克风”的“设置”。



3. 进入设置页面，将请求使用摄像头、麦克风权限的网站加入使用列表，并单击“保存更改”。

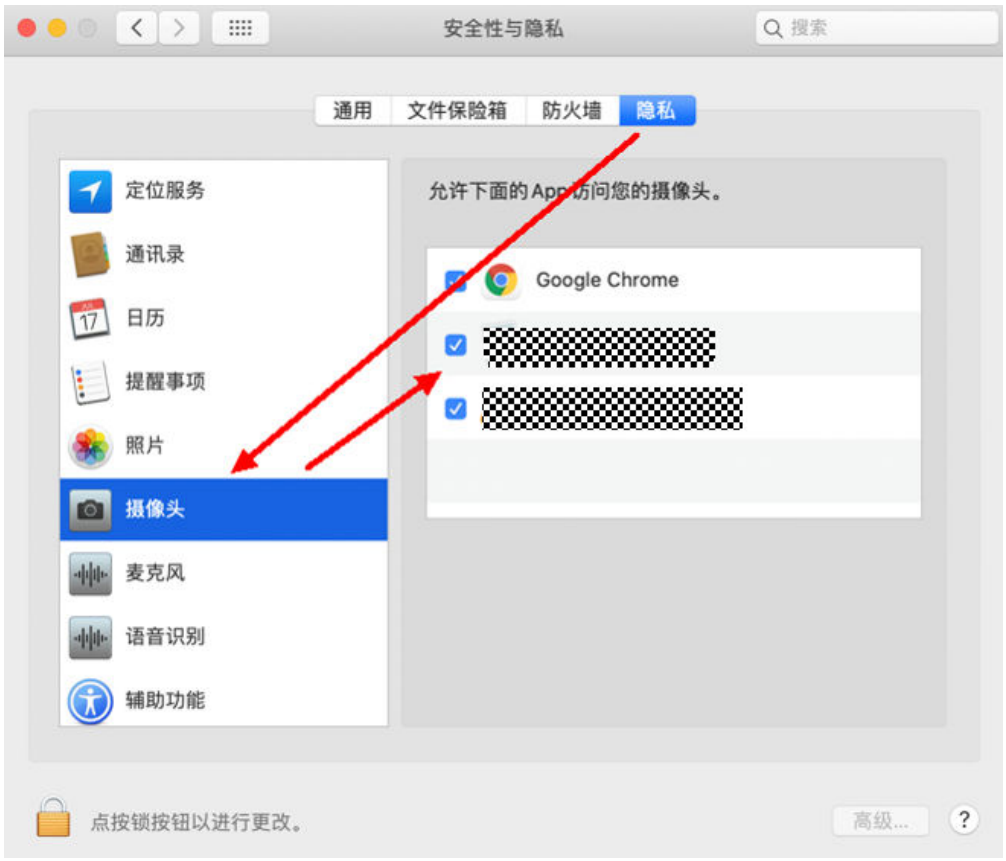
4. 完成以上设置后，浏览器在需要使用摄像头、麦克风时，就会在页面弹出设备询问框，选择“允许”即可。

Mac 系统的浏览器

1. 在程序坞中找到“系统偏好设置”并单击图标。

2. 打开后找到“安全性与隐私”并单击打开。

3. 在“安全性与隐私”页面选择“隐私”，然后单击“摄像头”/“麦克风”，将需要使用摄像头/麦克风的应用设置为允许。



3.6 常见问题

- **加入房间时，userName必须填吗?**

  非必填。userName、userId由App自定义，但可以相同。

  - userId：必填，string[64]类型，用户标识，userId需要保证应用内唯一。userId支持的字符包括：a-z、A-Z、0-9、连接符 '-'、下划线 '_'。
  - userName：选填，string[128]类型，用户昵称。

- **microphoneId跟cameraId在哪获取? 为什么是必填的?**

  分别为麦克风ID和摄像头ID ，音视频通话必须的，即在创建流的时候采集ID对应的音频、视频。

  使用**getDevices**、**getCameras**、**getMicrophones**接口可直接获取媒体输入输出、麦克风/摄像头设备ID 。

- **若摄像头不打开，是否显示默认的人像图标?**

  如果获取不到摄像头 ，但是能获取到麦克风，则视频是黑屏，音频有流。如果麦克风和摄像头都获取不到，则本地预览失败，不会显示默认的人像图标。

- **退出房间后摄像头没关，是不是需要释放摄像头?**

  退出房间不需要手动释放摄像头，会自动关闭，不再采集摄像头。

- **如何鉴权? 在什么时候鉴权?**

  具体请参见**接入鉴权**。

- **客户端以joiner角色加入房间失败。**

  客户端调用**join**时传入的角色参数不对。role是数值型，如果客户端传入的是字符串型，则会加入房间失败。

- **创建本地流失败，控制台提示Cannot read property 'getUserMedia' of undefined，无法获取到媒体源。**

  可能有以下原因：

  原因1：系统未允许应用访问摄像头等媒体源，解决方法请参见**授权浏览器摄像头/麦克风访问方法**。

  原因2：由于浏览器的策略，仅允许通过https://方式或者localhost的方式访问用户的摄像头和麦克风权限。

  原因3：检查摄像头等设备是否被其他应用占用了。

- **如果之前访问过使用Web SDK开发的App网站，又清理了该网站的权限，存在一定几率无法开启摄像头和麦克风。**

  在保证打开**授权浏览器摄像头/麦克风访问权限的方法**前提下。点开网页的左上角，将权限改为允许。如下图所示：

- **输入在线音频地址，且该地址可以在浏览器中打开，但是使用Web SDK的混音功能时，启动混音失败。**

  需要确认在线音频文件下载服务器是否支持跨域，由于浏览器的安全策略，必须要支持跨域，否则请求失败。

  注意：混音只有对端可以听到，本端听不见。

- **音频TopN 模式（音频最大三方模式）是什么意思?**

  音频TopN模式也叫音频最大三方模式。开启音频TopN模式后，本地用户不需要通过调用接口，单独订阅某个远端用户的音频，即可接收到当前房间内音量值最大的三个用户的音频。具体接口调用可参见**切换音频模式**。

- **调用setVolume4TopThree设置音频最大三方音量值是设置房间内三个最大声音的用户的音量值吗? 传参需要传一个参数还是三个参数?**

  是的。传参只需要传一个参数。

- **如果业务上App只能使用http协议，是否能够集成使用SparkRTC Web SDK ?**

  可以集成使用，但不推荐。需要用户手动关闭安全策略相关的开关。打开chrome页签，输入**chrome://flags/#unsafely-treat-insecure-origin-as-secure**，开启开关项，并把App的加载地址加入到忽略列表。

- **Firefox浏览器中使用Web SDK，加入房间失败怎么办?**

  请排查Firefox浏览器的H264插件是否安装。浏览器中输入**about:addons**，跳转到插件安装页面，查看H264插件是否安装完成，如未安装请在该页面更新安装。

- **使用Mac Chrome浏览器屏幕分享失败，提示 "NotAllowedError: Permission denied by system" 或 "NotReadableError: Could not start video source" 时怎么办?**

可能是由于未开启浏览器的屏幕录制权限导致。您可以通过在Mac的“设置 > 安全性与隐私 > 隐私 > 屏幕录制”中，打开Chrome屏幕录制授权后，重启Chrome浏览器。

3.7 修订记录

表 3-15 修订记录

修改时间	修改说明
2023-11-30	第二次正式发布 本次变更如下： **客户端对象（Client）**新增旁路推流接口：startLiveStreaming、updateLiveStreaming和stopLiveStreaming。
2022-09-30	第一次正式发布

4 接入鉴权

为保证SparkRTC的通信安全，当用户加入房间时，华为云SparkRTC服务需要对其进行接入鉴权。本章节主要介绍华为云SparkRTC接入鉴权的实现原理及鉴权签名的生成方法。

鉴权原理

华为云SparkRTC系统使用数字签名作为接入鉴权方式，需要在SDK加入房间时设置“signature”和“ctime”。“signature”为标识签名，由租户使用华为云SparkRTC提供的“app_key”，“app_id”以及当前的“room_id”，“user_id”，“ctime”，按照华为SparkRTC的**签名生成样例**自行生成。具体参数说明请参见**表4-1**。

```
//认证用的app_key和app_id硬编码至代码中或以明文形式存储会有极大风险。建议密文形式配置存储在文件或者环境变量中，使用时解密，以确保安全。本例以app_key和app_id存放至环境变量为例，运行前请先在本地环境中设置完成环境变量APP_KEY和APP_ID。
app_key =  System.getenv("APP_KEY");
app_id =  System.getenv("APP_ID");
signature = hmacSha256(app_key,(app_id + room_id + user_id + ctime))
```

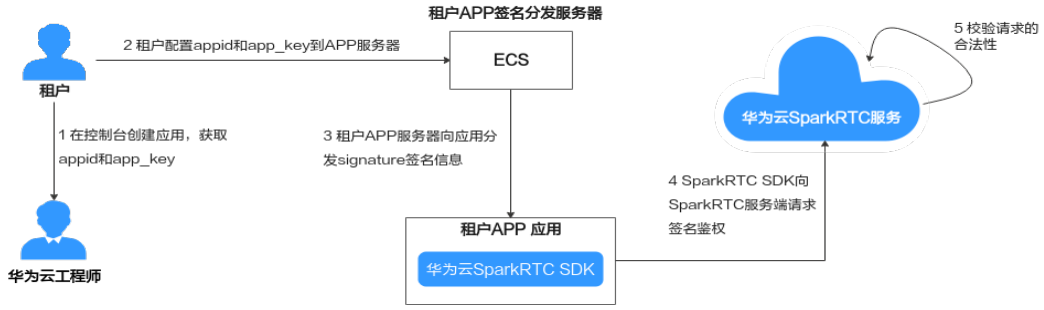
表 4-1 参数说明

参数	说明
app_key	华为云SparkRTC针对每个app生成的鉴权密钥，需要安全保存，谨防泄漏。 app_key的获取方法请参见**如何获取密钥？**。
app_id	华为云SparkRTC生成的应用ID。 app_id请在实时音视频控制台的“应用管理”中获取。
room_id	租户自行创建的房间ID。
user_id	租户接入华为云SparkRTC系统的用户ID。

参数	说明
ctime	签名鉴权的过期时间。是系统当前UTC时间（unix时间戳）加上鉴权过期时间（推荐2小时，最长需要小于12小时）。单位为秒。 **说明** ctime为创建时间+过期时间，例如，当前时间为9点，鉴权过期时间为30分钟，则ctime为9点30分。即超过9点30分后，signature签名将失效。

建议租户构建自己的应用签名分发服务器，以防止“app_key”下沉到终端APP的过程中造成不必要的泄漏，鉴权原理如图4-1所示。

图 4-1 鉴权原理



签名生成方法

您可以参考如下方法生成对应的签名。

1. 将“app_id”、“room_id”，“user_id ”和“ctime”拼接为一个字符串。

```
long ctime = System.currentTimeMillis() / 1000 + 60 * 60;  //有效时间为1小时，单位是秒
String content = app_id + "+" + room_id + "+" + user_id + "+" + ctime;
```

2. 使用“app_key”，通过HMAC-SHA256方式将字符串“content”进行加密，得到签名字符串。

```
String signatureStr = hmacSha(appKey, content, "HmacSHA256");
static String hmacSha(String KEY, String VALUE, String SHA_TYPE) {
    try {
        SecretKeySpec signingKey = new SecretKeySpec(KEY.getBytes("UTF-8"), SHA_TYPE);
        Mac mac = Mac.getInstance(SHA_TYPE);
        mac.init(signingKey);
        byte[] rawHmac = mac.doFinal(VALUE.getBytes("UTF-8"));

        byte[] hexArray = {
                (byte) '0', (byte) '1', (byte) '2', (byte) '3',
                (byte) '4', (byte) '5', (byte) '6', (byte) '7',
                (byte) '8', (byte) '9', (byte) 'a', (byte) 'b',
                (byte) 'c', (byte) 'd', (byte) 'e', (byte) 'f'
        };
        byte[] hexChars = new byte[rawHmac.length * 2];
        for (int j = 0; j < rawHmac.length; j++) {
            int v = rawHmac[j] & 0xFF;
            hexChars[j * 2] = hexArray[v >>> 4];
            hexChars[j * 2 + 1] = hexArray[v & 0x0F];
        }
        return new String(hexChars);
    } catch (Exception ex) {
        throw new RuntimeException(ex);
```

```
    }
  }
```

签名生成样例

为防止“app_key”密钥泄漏，建议您配置自己的应用签名分发服务器，向服务器传入“app_id”、“room_id”，“user_id”和“ctime”后，由服务器返回签名。详细代码示例如下所示：

```
package com.xxx.xxx.utils;

import androidx.annotation.NonNull;

import com.alibaba.fastjson.JSON;
import com.huawei.rtcdemo.Constants;

import java.io.IOException;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import okhttp3.Response;

public class SignatureUtil {
    private static final String TAG = "SignatureUtil";

    public interface onSignatureSuccess {
        void onSuccess(String signature);
    }

    public static void getSignature(String appid, String roomid, String userid, long ctime, String signatureKey,
onSignatureSuccess callback) {
        if (Constants.RTC_SIGNATURE_USE_LOCAL) {
            getSignatureLocal(appid, roomid, userid, ctime, signatureKey, callback);
        } else {
            getSignatureRemote(appid, roomid, userid, ctime, callback);
        }
    }

    private static void getSignatureLocal(String appid, String roomid, String userid, long ctime, String
signatureKey, @NonNull onSignatureSuccess callback) {
        String content = appid + "+" + roomid + "+" + userid + "+" + ctime; // here "+" is real char in content.
        String signature = SignatureUtil.hmacSha256(signatureKey, content);
        callback.onSuccess(signature);
    }

    private static void getSignatureRemote(String appid, String roomid, String userid, long ctime, @NonNull
onSignatureSuccess callback) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                HttpUtil httpUtil = new HttpUtil();
                // Constants.RTC_SIGNATURE_URL：带用户自己应用签名的分发服务器地址
                String url = Constants.RTC_SIGNATURE_URL + "?appid=" + appid + "&roomid=" + roomid +
"&userid=" + userid + "&ctime=" + ctime;
                Response response = httpUtil.sendGetMethodWithHead(url, "X-AUTH-TOKEN",
Constants.RTC_AUTH_TOKEN);
                if (response == null) {
                    return;
                }

                if (response.isSuccessful()) {
                    try {
                        String json = response.body().string();
                        String signature = JSON.parseObject(json).getString("signature");
                        callback.onSuccess(signature);
                    } catch (IOException e) {
                        LogUtil.e(TAG, "getSignature failed:" + e.getMessage());
```

```
                }
            } else {
                LogUtil.e(TAG, "getSignature failed!");
            }
        }
    }).start();
}

public static String hmacSha256(String KEY, String VALUE) {
    return hmacSha(KEY, VALUE, "HmacSHA256");
}

private static String hmacSha(String KEY, String VALUE, String SHA_TYPE) {
    try {
        SecretKeySpec signingKey = new SecretKeySpec(KEY.getBytes("UTF-8"), SHA_TYPE);
        Mac mac = Mac.getInstance(SHA_TYPE);
        mac.init(signingKey);
        byte[] rawHmac = mac.doFinal(VALUE.getBytes("UTF-8"));

        byte[] hexArray = {
                (byte) '0', (byte) '1', (byte) '2', (byte) '3',
                (byte) '4', (byte) '5', (byte) '6', (byte) '7',
                (byte) '8', (byte) '9', (byte) 'a', (byte) 'b',
                (byte) 'c', (byte) 'd', (byte) 'e', (byte) 'f'
        };
        byte[] hexChars = new byte[rawHmac.length * 2];
        for (int j = 0; j < rawHmac.length; j++) {
            int v = rawHmac[j] & 0xFF;
            hexChars[j * 2] = hexArray[v >>> 4];
            hexChars[j * 2 + 1] = hexArray[v & 0x0F];
        }
        return new String(hexChars);
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
}
}
```

5 附录

5.1 Grs 国家/地区码对照表

DR1：中国区

国家/地区名（英文全称）	国家/地区码（英文缩写）
China	CN

DR2：亚非拉（新加坡）

国家/地区名（英文全称）	国家/地区码（英文缩写）
The United Arab Emirates	AE
Afghanistan	AF
Antigua And Barbuda	AG
Anguilla	AI
Armenia	AM
Angola	AO
Antarctica	AQ
Argentina	AR
American Samoa	AS
Aruba	AW
Azerbaijan	AZ
Barbados	BB

国家/地区名（英文全称）	国家/地区码（英文缩写）
Bengal	BD
Burkina Faso	BF
Bahrain	BH
Burundi	BI
Benin	BJ
Saint Barthélemy	BL
Bermuda	BM
Brunei Darussalam	BN
Bolivia	BO
Brazil	BR
The Bahamas	BS
Bhutan	BT
Bouvet Island	BV
Botswana	BW
Belize	BZ
Cocos (Keeling) Islands	CC
Congo (Democratic Republic of the)	CD
Central African Republic	CF
Congo	CG
Côte d'Ivoire	CI
Cook Islands	CK
Chile	CL
Cameroon	CM
Colombia	CO
Costa Rica	CR
Cabo Verde	CV
Christmas Island	CX
Djibouti	DJ
Dominica	DM
Dominican Republic	DO

国家/地区名（英文全称）	国家/地区码（英文缩写）
Algeria	DZ
Ecuador	EC
Egypt	EG
Western Sahara	EH
Eritrea	ER
Ethiopia	ET
Fiji	FJ
Falkland Islands	FK
Micronesia (Federated States of)	FM
Gabon	GA
Grenada	GD
Georgia	GE
French Guiana	GF
Ghana	GH
Gambia	GM
Guinea	GN
Guadeloupe	GP
Equatorial Guinea	GQ
South Georgia and the South Sandwich Islands	GS
Guatemala	GT
Guam	GU
Guinea-Bissau	GW
Guyana	GY
Hong Kong	HK
Heard Island and McDonald Islands	HM
Honduras	HN
Haiti	HT
Indonesia	ID
India	IN

国家/地区名（英文全称）	国家/地区码（英文缩写）
British Indian Ocean Territory	IO
Iraq	IQ
Jamaica	JM
Jordan	JO
Kenya	KE
Kyrgyzstan	KG
Cambodia	KH
Kiribati	KI
Comoros	KM
Saint Kitts and Nevis	KN
Kuwait	KW
Cayman Islands	KY
Kazakhstan	KZ
Laos	LA
Lebanon	LB
Saint Lucia	LC
Sri Lanka	LK
Liberia	LR
Lesotho	LS
Libya	LY
Morocco	MA
Madagascar	MG
Marshall islands	MH
Mali	ML
Myanmar	MM
Mongolia	MN
Macao	MO
Northern Mariana Islands	MP
Martinique	MQ
Mauritania	MR

国家/地区名（英文全称）	国家/地区码（英文缩写）
Montserrat	MS
Mauritius	MU
Maldives	MV
Malawi	MW
Mexico	MX
Malaysia	MY
Mozambique	MZ
Namibia	NA
New Caledonia	NC
Niger	NE
Norfolk Island	NF
Nigeria	NG
Nicaragua	NI
Nepal	NP
Nauru	NR
Niue	NU
Oman	OM
Panama	PA
Peru	PE
French Polynesia	PF
Papua New Guinea	PG
Philippines	PH
Pakistan	PK
Pitcairn	PN
Puerto Rico	PR
Palestine, State of	PS
Palau	PW
Paraguay	PY
Qatar	QA
Reunion !Réunion	RE

国家/地区名（英文全称）	国家/地区码（英文缩写）
Rwanda	RW
Saudi Arabia	SA
Solomon Islands	SB
Seychelles	SC
Singapore	SG
Saint Helena, Ascension and Tristan da Cunha	SH
Sierra Leone	SL
Senegal	SN
Somalia	SO
Suriname	SR
South Sudan	SS
Sao Tome and Principe	ST
El Salvador	SV
Swaziland	SZ
Turks and Caicos Islands	TC
Chad	TD
French Southern Territories	TF
Togo	TG
Thailand	TH
Tajikistan	TJ
Tokelau	TK
Timor-Leste	TL
Turkmenistan	TM
Tunisia	TN
Tonga	TO
Trinidad and Tobago	TT
Tuvalu	TV
Taiwan, Province of China	TW
Tanzania, United Republic of	TZ

国家/地区名（英文全称）	国家/地区码（英文缩写）
Uganda	UG
Uruguay	UY
Uzbekistan	UZ
Venezuela (Bolivarian Republic of)	VE
Virgin Islands, British	VG
Virgin Islands, U.S.	VI
Vietnam	VN
Vanuatu	VU
Wallis and Futuna	WF
Samoa	WS
Yemen	YE
Mayotte	YT
South Africa	ZA
Zambia	ZM
Zimbabwe	ZW
Cuba	CU
North Korea	KP
Sudan	SD
Syria	SY
Japan	JP
Korea (Republic of)	KR

DR3：欧洲区（德国）

国家/地区名（英文全称）	国家/地区码（英文缩写）
Andorra	AD
Albania	AL
Austria	AT
Aland Islands	AX
Bosnia and Herzegovina	BA

国家/地区名（英文全称）	国家/地区码（英文缩写）
Belgium	BE
Bulgaria	BG
Canada	CA
SWITZERLAND	CH
Cyprus	CY
Czech Republic	CZ
Germany	DE
Denmark	DK
Estonia	EE
Spain	ES
Finland	FI
Faroe Islands	FO
France	FR
United Kingdom of Great Britain and Northern Ireland	GB
Guernsey	GG
Gibraltar	GI
Greenland	GL
Greece	GR
Croatia	HR
Hungary	HU
Ireland	IE
Israel	IL
Isle of Man	IM
Iceland	IS
Italy	IT
Jersey	JE
Liechtenstein	LI
Lithuania	LT
Luxembourg	LU

国家/地区名（英文全称）	国家/地区码（英文缩写）
Latvia	LV
Monaco	MC
Moldova	MD
Montenegro	ME
Saint Martin (French part)	MF
Macedonia (the former Yugoslav Republic of)	MK
Malta	MT
Netherlands	NL
Norway	NO
Poland	PL
Saint Pierre and Miquelon	PM
Portugal	PT
Romania	RO
Serbia	RS
Sweden	SE
Slovenia	SI
SJMSvalbard	SJ
Slovakia	SK
San Marino	SM
Sint Maarten (Dutch part)	SX
Türkiye	TR
United States Minor Outlying Islands	UM
America	US
Holy See	VA
Saint Vincent and the Grenadines	VC
Kosovo	XK->YK
Australia	AU
New Zealand	NZ
Netherlands Antilles	BQ->AN

国家/地区名（英文全称）	国家/地区码（英文缩写）
Ukraine	UA
Curaçao	CW

6 修订记录

表 6-1 修订记录

发布日期	修改说明
最新时间	Web SDK的修订记录，详见**修订记录**。
2020-10-30	第一次正式商用发布。