实时音视频

无 UI 集成

产品文档



腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

无 UI 集成
SDK 下载
SDK 下载
发布日志（App）
发布日志（Electron）
发布日志（Web）
发布日志（Flutter）
API-Example
iOS
Mac
Android
Windows C++
Web
Electron
Flutter
Unity
React Native
API-使用指引
SDK 快速开始
Android
Mac
Windows C++
01. 导入 SDK 到项目中
iOS
Android
Mac
Windows C++
Web
Electron
Flutter
QT
Unity
02. 进入房间
Android&iOS&Windows&Mac
Web

版权所有：腾讯云计算（北京）有限责任公司

- Electron
- Flutter

03. 订阅音视频流
- Android&iOS&Windows&Mac
- Web
- Electron
- Flutter

04. 发布音视频流
- Android&iOS&Windows&Mac
- Web
- Electron
- Flutter

05. 退出房间
- Android&iOS&Windows&Mac
- Web
- Electron
- Flutter

06. 进阶指引
- 感知网络质量
  - Android&iOS&Windows&Mac
  - Web
  - Electron
  - Flutter
- 开启屏幕分享
  - iOS
  - Android
  - Mac
  - Web
  - Windows
  - Electron
  - Flutter
- 分享系统声音
  - Web
  - Mac
  - Electron
  - Flutter
- 设定画面质量
  - Android&iOS&Windows&Mac

版权所有：腾讯云计算（北京）有限责任公司

Web
Electron
Flutter
视频画面旋转和播放
Android&iOS&Windows&Mac
Electron
Web
Flutter
07. 常见问题
全平台
API-参考手册
iOS&Mac
API 概览
TRTCCloud
TRTCCloudDelegate
TRTCStatistics
TXAudioEffectManager
TXBeautyManager
TXDeviceManager
类型定义
废弃接口
错误码表
Android
API 概览
TRTCCloud
TRTCCloudListener
TRTCStatistics
TXAudioEffectManager
TXBeautyManager
TXDeviceManager
类型定义
废弃接口
错误码表
全平台（C++）
API 概览
ITRTCCloud
TRTCCloudCallback
ITRTCStatistics

ITXAudioEffectManager
ITXDeviceManager
类型定义
废弃接口
错误码表
Web
API 概览
错误码
Electron
API 概览
错误码
Flutter
API 概览
错误码
Unity
API 概览
错误码

无 UI 集成
SDK 下载
SDK 下载

最近更新时间：2023-06-14 16:42:39

实时音视频（TRTC） 是腾讯云提供的一套低延时、高质量的音视频通讯服务，致力于为腾讯云客户提供稳定、可靠和低成本的音视频传输能力。该服务由一套遍布全球的音视频传输网络和一组终端 SDK 组成，您可以在本页面下载到涵盖目前主流客户端平台和热门框架的实时音视频（TRTC） SDK。

实时音视频（TRTC）SDK

全功能（All-In-One）SDK

除实时音视频（TRTC）SDK，腾讯云还为多种音视频场景提供了直播、短视频、播放器等终端SDK快速集成方案。全功能（All-In-One）SDK 是为了解决多款SDK同时使用时引起的依赖库冲突问题，在一个 SDK 内包含了 实时音视频（TRTC）SDK、直播 SDK 、短视频 SDK 和播放器 SDK 的全部功能。如果您同时需要多个SDK，可以下载此版本。

版权所有：腾讯云计算（北京）有限责任公司

发布日志（App）

最近更新时间：2024-06-13 17:08:37

此页面包含 实时音视频（TRTC）SDK 的版本历史 和 全功能（All-In-One）SDK 的版本历史，请根据您的功能需要选择适合您的 SDK。

Version 11.9 @ 2024.6.12

新特性：

全平台：优化 AI 降噪效果，大幅提升在网吧、会议等场景声音体验。
全平台：支持播放音速达全量的版权音乐，提升K歌、合唱、语聊等热门场景的互动体验。
全平台：支持播放 ape 格式的音频文件。
全平台：优化现有操作事件准确性，提升无声、声音小等异常场景排障效率。
iOS&Android：支持高清截图能力。
iOS：TXDeviceManager 新增设置采集参数接口 setCameraCapturerParam。

缺陷修复：

Android：解决麦克风采集偶现电音问题。
全平台：修复特定 BGM 格式及非法数据引起的稳定性问题。
Android：解决外放模式偶现声音小的问题。
Windows：修复偶现获取当前摄像头不对的问题。

Version 11.8 @ 2024.5.10

新特性：

iOS：支持画中画能力。

功能优化：

Android：音频兼容性问题优化，支持更加完善的机型采播选路策略。
Window：优化AI降噪性能。
Windows：单主播场景上行画质优化。
全平台：优化 Bgm 错误提示信息。
Windows：缩放效果优化。

缺陷修复：

Android：优化外放声音小badcase。
Mac：修复开 Loopback 杀进程无声问题。
Mac：修复链接HDMI、DisplayPort无声问题。

全平台：修复实验性接口打开3A无效问题。
Windows：修复不同音质类型切换导致AGC失效问题。
Android：屏幕分享适配高TargetVersion。

接口行为调整：

全平台：C++接口支持 onMixedAllAudioFrame 回调。

Version 11.7 @ 2024.3.4

新特性：

全平台：增加摄像头采集的警告码。
全平台：新增重力感应接口 setGravitySensorAdaptiveMode。
全平台：支持回调本地人声音高信息，详见 enableAudioVolumeEvaluation 。

功能优化：

全平台：优化进房流程，大幅降低二次进房耗时。
全平台：仪表盘监控单个用户端到端通话质量最大路数从 16 路提升到 50 路。
Android：优化 Android 音频采集处理策略，降低杂音率，提升声音采集效果。
iOS：优化打断恢复重启策略，降低无声率。
iOS：优化unity 3D引擎兼容性问题。
iOS：优化后置三摄、双摄采集对焦效果，提升对焦速度。

缺陷修复：

Android 修复部分漏回声case。
Android 修复部分蓝牙被打断中途外放问题。

接口行为调整：

全平台：频繁调用 `switchRole` 接口时， `onSwitchRole` 只返回最后一次调用结果。
全平台：增加摄像头采集相关警告码

错误码	描述
WARNING_CAMERA_IS_OCCUPIED = 1114	摄像头被占用
WARNING_CAMERA_DEVICE_ERROR = 1115	摄像头设备异常
WARNING_CAMERA_DISCONNECTED = 1116	摄像头无法连接
WARNING_CAMERA_START_FAILED = 1117	摄像头启动失败

WARNING_CAMERA_SERVER_DIED = 1118	系统异常

全平台：setVideoMuteImage 接口默认垫片帧率由 0fps 调整为 5fps，限定垫片最大帧率为 10fps。
iOS：setSubStreamEncoderParam 接口行为由设置屏幕分享的编码参数调整为设置辅路编码参数。
Windows:selectScreenCaptureTarget 中默认高亮颜色（highLightColor）由绿色调整为黄色。
全平台：调整重力感应相关接口：

新增接口	(void)setGravitySensorAdaptiveMode: (TRTCGravitySensorAdaptiveMode)mode;	参见接口文档
废弃接口	setVideoEncoderRotation	废弃
	setVideoEncoderMirror	废弃
	setGSensorMode	废弃

Version 11.6 @ 2024.1.15

新特性：

iOS：新增 `TXLivePlayer` 支持画中画。
Windows：新增 startSystemAudioLoopback 支持采集指定的第三方进程音频，详情见 startSystemAudioLoopback。

功能优化：

Android&iOS：优化使用 URL 播放 BGM 的成功率。
Windows：优化并适配 Intel HEVC 软件解码器（Quick Sync Video）。
全平台：优化低带宽条件下的音频弱网表现。
全平台：优化低带宽条件下的视频弱网表现。
全平台：优化高丢包高延时下的音质。
全平台：优化 SDK 底层逻辑，提升整体稳定性。

缺陷修复：

Android：修复频繁 `switchRoom` 时偶现首帧解码慢的问题。
Android：修复在单次会话中尝试再次 `startScreenCapture` 时偶现的画面比例不正确的问题。
Windows：修复设置竖屏分辨率时部分虚拟摄像头采集失败的问题。

Version 11.5 @ 2023.11.27

功能优化：

全平台：优化视频模块的整体性能及稳定性。
全平台：优化音频模块的整体稳定性。

版权所有：腾讯云计算（北京）有限责任公司

全平台：优化部分 API 接口行为策略，详情见接口行为调整。
全平台：优化音频背景音乐模块整体策略及性能占用，减少背景音乐播放异常的情况。
Windows：优化 HEVC 硬件解码策略，适配 AMD、Nvidia 显卡。
Windows：优化屏幕共享整体性能，提升屏幕采集帧率及稳定性。
Android：优化 TRTC + VODPlayer 场景下的播放效果。
iOS&Mac：优化使用 Metal 进行预处理及渲染的性能。

接口行为调整：

全平台：视频设置分辨率为540P竖屏时（预期540x960），具体的编码处理分辨率由544x960调整为536x960。
全平台：背景音乐进度回调 `onPlayProgress` 回调间隔由200ms 调整为300ms。
全平台：背景音乐模块内部实现调整为单例，多实例情况下 musicID 需要全局唯一。开发者在使用子实例播放背景音乐时，请确保不同实例使用不同的 musicID。
全平台：本地录制事件状态码调整为异步返回，相关接口调用后默认返回 0，具体的状态码通过相应的事件回调获取。
全平台：启动录制事件回调 `onLocalRecordBegin` 调整以下状态码：

事件	11.5 之前版本状态码	11.5 版本状态码
录制已经启动，需要先停止录制	-1	-6
录制目录无写入权限，请检查目录权限问题	-2	-8
文件后缀名有误（比如不支持的录制格式）	-3	-2

iOS&Android：优化移动端屏幕共享连续性，在共享暂停时保留发送最后一帧，发送帧率为1 - 2fps。
iOS&Android：调整重力感应响应行为，只响应重力感应开或者关。

Version 11.4 @ 2023.08.30

新特性：

全平台：TRTCLocalRecordingParams 新增 `maxDurationPerFile` 用于控制分片录制时长，分片文件路径可通过 onLocalRecordFragment 回调获取。
Android&iOS：V2TXLivePusher 推流本地预览增加渲染模式设置接口 setRenderFillMode。
Mac：新增 `enableCrashMonitoring` ，支持捕获 crash 信息并进行本地存储，使用时需要在项目中添加 **TXCCrashMonitor.framework。**

功能优化：

全平台：优化提升 IPv6 网络环境下的整体表现。
全平台：优化合唱场景下歌词精准对齐。
全平台：优化 AI 降噪算法，进一步提升降噪效果。
全平台：优化提升纯音频场景下观众拉流播放流畅度。
全平台：优化 switchRoom 切换房间的平滑性，避免闪黑帧。
Android&iOS：优化提升直播播放秒开率。
Android&iOS：优化音频采集处理策略，降低因采集设备异常导致的无声问题概率。
Android：优化麦克风被系统静默后的回调通知。
Android：优化特定 Android 定制设备重力感应适配逻辑，避免设备返回的重力感应方向错误时画面旋转角度不对。
Android：优化渲染处理方式，支持 View 双指缩放时画面实时跟随，提升浮窗播放时的用户体验。
iOS：优化后台状态下音频采集策略，降低因系统打断引起的无声问题概率。
iOS：优化提升音频设备重启速度。

Version 11.3 @ 2023.07.07

新特性：

全平台：新增视频画面梯形校正功能（仅 Professinal 版本支持），用于手动修正透视画面角度畸变。详见 `setPerspectiveCorrectionPoints`。
全平台：新增声音频谱回调，可用于声浪动画或音量频谱展示。详见 `enableAudioVolumeEvaluation` 及 `TRTCVolumeInfo`。
全平台：新增混响效果“录音棚2”，详见 `TXVoiceReverbType`。
全平台：新增混流接口 SEI 参数设置，用于转推 CDN 分发时透传 SEI。详见 `TRTCTranscodingConfig`。
Windows：新增音速达版权曲库音乐打分功能，可用于合唱实时评分，详见 `createSongScore`。
iOS&Android：新增 `startPlayMusic` 对 .ogg 格式音乐文件的支持。
Flutter：新增 `setSystemAudioLoopbackVolume` 方法（iOS）。

功能优化：

全平台：优化自适应数字增益算法，提升声音听感，增强音量稳定性，避免音量过大或过小。
全平台：优化视频秒开耗时，提升进房后视频第一帧画面的加载速度。
全平台：优化单主播推流的弱网抗性，提升网络延时抖动情况下的流畅度。
Android：优化音频采集播放逻辑，避免在部分 Android 设备上出现声音异常的问题。
Android：优化视频辅流硬编表现，提升共享屏幕的画面质量。
iOS：优化音频设备重启逻辑，减少声音中断的发生次数。
iOS&Android：TXLivePlayer 删除点播相关接口，点播视频请使用 TXVodPlayer 进行播放。

缺陷修复：

Android：修复Android 12及以上版本系统部分本地录制的视频在苹果Safari 浏览器上无法播放的问题。

Version 11.2 @ 2023.06.05

新特性：

全平台：支持合唱场景下 BGM 伴奏、原唱无缝切换，详见 `setMusicTrack`。
Android：全功能版（Professional）、直播基础版（Smart）支持x86架构，并支持通过Maven获取。
Android：满足 Android 12 及以上版本的操作系统要求，在屏幕采集时启动前台服务，详见：`enableForegroundService`。
iOS：支持在 Apple 芯片设备上通过 Xcode 模拟器运行 SDK。
Mac：支持获取屏幕窗口信息时返回宽高等信息，对齐 Windows，详见：TXCScreenSourceInfo。

功能优化:

全平台：优化合唱场景的整体音质，提升合唱效果，减少合唱延迟。
全平台：优化上下麦时的音频效果，上下麦体验更平滑。
全平台：优化极限弱网下的音频体验。
全平台：优化直播单主播推流时的弱网体验。
全平台：优化视频通话场景大小流切换过程的流畅度。
Android&iOS：优化音乐场景下的音质表现，提升合唱体验。
Android&iOS：优化不同音量类型下使用蓝牙耳机的体验。
Android：优化硬件解码延时，提升首帧体验。
Android：优化耳返功能，提升开关耳返时的体验。
Android：优化 Android 设备的采集兼容性，减少音频异常问题。
iOS：优化画质表现，提升视频体验。

缺陷修复:

Windows：修复窗口分享时偶现闪烁问题。
Mac：修复 Intel 芯片设备上使用摄像头采集时偶现的硬编码器编码画面呼吸效应问题。

TRTC SDK 发布日志10.8 @ 2022.10.27

新特性

全平台：新增搓碟音效，提供更加全面的在线 K 歌体验，详见：TXAudioEffectManager.setMusicScratchSpeedRate。

功能优化

Android：优化视频解码启动速度，有效提升画面秒开速度，最快可以达到 50ms。
全平台：优化 NTP 时间的准确性，详见：TXLiveBase.updateNetworkTime。

缺陷修复

全平台：修复下特定场景下（无音视频上行）混流机器人 回推 TRTC 房间场景中，偶现的拉流异常以及回调错误的问题。
全平台：修复观众进房后切换角色时，因网络类型变化偶现的音视频上行失败问题。
全平台：修复在断网重连过程中出现的音质切换不生效问题。
全平台：修复在断网重连过程中偶现的上行无声问题。
Android & iOS：修复当调用 muteRemoteVideoStream 时会移除最后一帧视频画面的问题。

TRTC SDK 发布日志 10.7 @ 2022.09.20

新特性

全平台：云端混流支持调整每路输入流的音量，详见 TRTCMixUser.soundLevel。
全平台：新增了 onRemoteAudioStatusUpdated 回调接口，可用于更好地识别和监控远端音频流状态。

功能优化

全平台：升级编码内核，提升屏幕分享场景的画质。
全平台：优化弱网下编码码控效果。

缺陷修复

iOS：修复 iPad 部分设备采集音量较小的问题。
Android：修复偶现连接蓝牙耳机但是声音外放的问题。
全平台：修复频繁进退房场景下偶现的 crash 问题。

TRTC SDK 发布日志10.6 @ 2022.09.09

功能优化

全平台：提升在 IPv6 网络环境下的进房速度。
全平台：优化弱网络环境下音频的恢复效率以及音画同步效果，提升通话体验。
全平台：优化弱网络环境下的连接保持能力，减少断网重连概率。
全平台：优化 Music 档位（在 startLocalAudio 时指定）下音量较小的问题，提升用户体验。
Mac：优化使用蓝牙耳机时的沟通体验，杂音更少，声音更清晰。
Android：优化立体声采集的兼容性，支持更多机型。
Android：优化偶现的漏回声问题，提升沟通体验。

缺陷修复

Android & iOS：修复在 Speech 档位（在 startLocalAudio 时指定）下偶现的漏字问题。
Mac：修复切换麦克风时偶现的回声消除失效的问题。

版权所有：腾讯云计算（北京）有限责任公司

All-In-One SDK 10.5 @ 2022.08.24

直播功能模块

功能优化

Android：优化视频解码的内存管理，防止出现内存堆积。
Windows：优化内置麦克风的降噪效果，尤其是在音乐模式下，表现更佳。
Mac：优化开启麦克风采集时，大概率出现的杂音问题。

缺陷修复

全平台：修复 V2TXLivePlayer 在快直播拉流时，偶现 SEI 无法收到的问题。
全平台：修复 V2TXLivePlayer 在快直播拉流时，时间戳回退导致无声的问题。

短视频功能模块

缺陷修复

Android：修复 UGC 鸿蒙系统下图片转场生成的视频绿屏问题。
Android：修复编辑生成的视频长度不对的问题。
Android：修复多声道视频无法播放和重编码的问题。
Android：修复动感光波的特效在选择的时间段内只产生一次效果的问题。
Android&iOS：修复短视频录制片段回删后 BGM 播放进度不对的问题。

音视频通话功能模块

功能优化

全平台：优化 qos 策略，提升弱网体验。
iOS&Android：全链路降低延迟；优化耳返体验。
Android：优化视频解码的内存管理，防止出现内存堆积。
Windows：优化内置麦克风的降噪效果，尤其是在音乐模式下，表现更佳。
Mac：优化开启麦克风采集时，大概率出现的杂音问题。

缺陷修复

全平台：修复频繁进退不同房间时，回调事件：OnUserVideoAvailable 和 OnUserAudioAvailable 偶现异常的问题。

播放器功能模块

缺陷修复

Android&iOS：修复播放失败不带视频格式后缀短链异常。

All-In-One SDK @ 2022.07.25

直播功能模块

新特性

iOS&Android：V2TXLivePlayer 结束播放时，允许保留最后一帧。

功能优化

全平台：优化 TXLivePlayer\\V2TXLivePlayer 在 FLV 拉流时，内存占用的问题。
Android：修复 TXLivePlayer\\V2TXLivePlayer 拉流时，偶现卡顿的问题。
Android：优化低延迟耳返及双声道采集的兼容性。
Android：优化硬解码切软解码的策略，提升解码性能。
iOS：优化 iPad 采集音量小的问题。

缺陷修复

Android：修复 TXLivePlayer\\V2TXLivePlayer 拉流时，偶现切换到软解的问题。

短视频功能模块

功能优化

Android：增加短视频编辑 setBGMLoop 接口。

缺陷修复

Android：修复视频 setWaterMark 不生效的问题。
Android：修复 TXVideoEditor 预览渲染模式不对的问题。

音视频通话功能模块

新特性

iOS&Android：自定义视频采集支持 RGBA32 格式，详见：sendCustomVideoData。
Windows&Mac：水印设置支持本地预览，详见：setWaterMark。

功能优化

Android：优化低延迟耳返及双声道采集的兼容性。
Android：优化硬解码切软解码的策略，提升解码性能。
iOS：优化 iPad 采集音量小的问题。

缺陷修复

全平台：修复偶现的进退房回调异常的问题。
Windows：修复切换分享窗口，新窗口的内容被剪裁的问题。

版权所有：腾讯云计算（北京）有限责任公司

播放器功能模块

功能优化

Android&iOS：HLS 直播支持自适应播放。

缺陷修复

Android：修复 onNetStatus 和进度回调间隔异常 。
Android：修复播放器没有调用 setConfig 引起的空指针异常。
iOS：修复部分场景下重播卡顿问题。

All-In-One SDK 10.3 @ 2022.07.08

直播功能模块

新特性：
全平台：TXLivePlayer\\V2TXLivePlayer 支持 HLS 播放。
功能优化：
全平台：优化 Music 音质下的声音效果。
全平台：优化 TXLivePlayer\\V2TXLivePlayer 的 SEI 解析逻辑，兼容部分非标准 SEI。
全平台：优化 TXLivePlayer\\V2TXLivePlayer 在 FLV、RTMP 拉流时，时间戳回退导致的音画不同步问题。
缺陷修复：
全平台：修复 TXLivePlayer\\V2TXLivePlayer 播放部分快直播 AAC-HEv2 流时，声音异常的问题。
全平台：修复 TXLivePlayer 视频缓存计算不正确的问题。

短视频功能模块

缺陷修复:
Android：修复视频录制 setZoom 不生效的问题。
Android：修复三星s22录制失败的问题。
iOS：自定义视频预处理不回调的问题。

音视频通话功能模块

新特性：
Windows：新增录制本地录制功能，可用于在本地录制互动直播或音视频通话完整内容。详见 ITXLiteAVLocalRecord。
Windows&Mac：新增参数支持在 startMicDeviceTest 接口中开启/关闭 播放麦克风检测时麦克风采集到的声音。详见 startMicDeviceTest。
功能优化：
全平台：优化 Music 音质下的声音效果。

缺陷修复：

全平台：修复房间用户列表偶现的回调异常问题。

Windows：修复视频播放过程偶现的画面卡住问题。

Windows：修复视频播放过程偶现的播放失败问题。

Windwos：修复音频自定义采集场景中出现回声的问题。

播放器功能模块

新特性：

iOS：视频播放支持画中画模式。

缺陷修复：

Android：修复硬解后台连续播放视频列表会中断问题。

Android&iOS：修复 Seek 完成事件不回调问题。

All-In-One SDK 10.2 @ 2022.06.26

直播功能模块

新特性：

全平台：TXLivePlayer\\V2TXLivePlayer 拉流时，新增 License 校验。

全平台：V2TXLivePlayer 在 FLV 拉流时， 支持设定 HTTP Headers。

全平台：TXLivePusher\\V2TXLivePusher 在 RTMP 推流时，允许中途变更音频编码参数。

功能优化：

全平台：优化 V2TXLivePlayer 在快直播拉流时，自适应码率的接口。

全平台：优化 V2TXLivePlayer 在快直播拉流时，重连耗时过长的问题。

全平台：优化 TXLivePlayer\\V2TXLivePlayer 在 FLV、RTMP 拉流时，本地缓存偏低问题。

Android：优化 TXLivePlayer\\V2TXLivePlayer 在拉流时，首帧秒开速度。

iOS：优化 iOS SDK 体积。

iOS：LiteaVSDK Live 版打包 TXLiveBase.h。

缺陷修复：

全平台：修复 TXLivePlayer 设定卡顿阈值不生效的问题。

全平台：修复 V2TXLivePusher 在 RTC 推流时，音视频首帧回调时机异常问题。

Android：修复 TXLivePlayer\\V2TXLivePlayer 在拉流时，快速 stop、start 偶现黑屏的问题。

短视频功能模块

新特性：

Android：支持无音轨视频编辑。

功能优化：

Android：优化短视频编辑播放的首开速度。

版权所有：腾讯云计算（北京）有限责任公司

缺陷修复：

Android：修复录制裁剪区域不对的问题。

Android：修复 H265 视频硬解帧宽高不对的问题。

iOS：修复短视频裁剪时间不准的问题。

iOS：修复 iOS 14 以上机型可能出现录制的杂音问题。

iOS：修复拍摄完成视频后再返回录制偶现崩溃的问题。

音视频通话功能模块

新特性：

全平台：全新推出更加灵活，且功能强大的混流转推 API。详见：startPublishMediaStream。

全平台：新增 3D 音频特效功能，详见：enable3DSpatialAudioEffect。

全平台：新增人声检测功能，当 muteLoalAudio 和 setAudioCaptureVolume 为 0 时不会影响人声检测结果。详见 enableAudioVolumeEvaluation，Tips：方便提示用户开麦。

全平台：切换角色时，增加支持权限校验的功能。详见：`switchRole(TRTCRoleType role, const char* privateMapKey)` 。

iOS&Mac：自定义预处理的 C++ 接口，支持以纹理方式对接视频处理。

功能优化：

Android：优化耳返效果，降低延迟。

Android：优化音频的采集链路，解决部分机型存在的杂音问题。

iOS：优化上行视频处理链路，节省 CPU、GPU 占用。

Windows&Mac：优化窗口分享时的编码性能，编码宽高不再受采集窗口大小的影响。

Windows：优化性能，减少内存碎片及其分配时造成性能开销。

缺陷修复：

全平台：修复切换网络类型时，偶现的上行失败问题。

iOS：修复在部分 iOS 14系统上，本地录制文件存在的杂音问题。

播放器功能模块

功能优化：

Android&iOS：优化播放过程中回调 cachedBytes、IP 地址等参数。

缺陷修复：

Android&iOS：修复硬解播放H265格式视频失败问题。

Android&iOS：修复播放 HLS 直播异常。

iOS：修复某些场景下获取 supportedBitrates 异常。

版权所有：腾讯云计算（北京）有限责任公司

发布日志（Electron）

最近更新时间：2023-07-18 16:37:01

Version 11.0.501 @ 2023.6.30

功能改进

API 文档新增 onStartPublishing 和 onStopPublishing 两个事件。
API 文档完善 onScreenCapturePaused 接口字段说明。
Electron SDK 支持 Linux（beta 版）。

Version 10.9.405 @ 2023.4.17

功能新增

新增接口 setCameraCaptureParams，支持设置摄像头采集参数，目前仅支持 Windows。
新增接口 setVideoMuteImage，支持摄像头 mute 后，设置垫片图片。
新增接口 enableFollowingDefaultAudioDevice，支持扬声器、麦克风跟随当前系统设备。
修改接口 setMixTranscodingConfig，支持设置每路视频流的输入类型、渲染模式、垫片图片。
修改接口 getScreenCaptureSources，返回值新增 isMainWindow 字段，目前仅支持 Windows。

问题修复

修复应用层刷新页面，偶现应用奔溃问题。

Version 10.7.405 @ 2023.2.27

功能新增

新增接口 updateLocalView 和 updateRemoteView，支持修改页面上视频观看、预览的位置。
Windows 下，getScreenCaptureSources 接口返回的屏幕、窗口信息新增 isMainScreen 字段。

问题修复

修复房间中多人屏幕分享时，不能同时渲染观看问题。
视频渲染 DOM 元素缩放从 transform scale 改为 zoom，保持向后兼容。
解决频繁开关摄像头，本地预览偶现绿屏帧问题。
Mac 下音视频软解的动态库从实体文件改外链接文件，解决 Mac 下构建应用包时出现的签名异常。
解决 setRemoteRenderParams 接口设置的 fillMode 参数在 startRemoteView 之前调用不生效问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 10.3.406 @ 2023.2.4

功能新增

Windows 下，getScreenCaptureSources 接口返回的屏幕、窗口信息新增 isMainScreen 字段。

问题修复

Windows 下，启动麦克风时，若麦克风为 mute 状态，主动取消 mute。
修复 selectScreenCaptureTarget 接口 highlight 设置不生效问题。
Windows 下，视频渲染流程优化。

Version 10.3.405 @ 2022.12.12

问题修复

修复发现的一些易用性问题。

Version 10.7.404 @ 2022.10.31

功能新增

设置水印接口 setWaterMark 支持 Windows 系统；Mac 下新增支持通过图片路径设置水印。
推流到非腾讯云 CDN 接口 startPublishCDNStream 接口新增 streamId 入参，支持设置流 ID。

Version 10.6.404 @ 2022.10.31

功能新增

设置水印接口 setWaterMark 支持 Windows 系统；Mac 下新增支持通过图片路径设置水印。
推流到非腾讯云 CDN 接口 startPublishCDNStream 接口新增 streamId 入参，支持设置流 ID。

问题修复

修复调用 setRemoteVideoStreamType() 切小流视频渲染卡住问题。
修复视频渲染偶现绿屏帧问题。
修复 Mac 下摄像头检测不支持镜像问题，解决 Mac 下首次分享窗口不出现高亮绿框问题。
修复 Mac 下分享窗口、屏幕时上行帧率为零的问题，导致远端用户收不到 onUserSubStreamAvailable 事件问题。

Version 10.3.404 @ 2022.10.31

版权所有：腾讯云计算（北京）有限责任公司

问题修复

修复调用 setRemoteVideoStreamType() 切小流视频渲染卡住问题。

Version 10.6.403 @ 2022.09.09

功能新增

Windows & Mac: 新增本地媒体录制接口，支持直播时，将本地音视频数据录制保存到本地文件。具体接口包括：startLocalRecording、stopLocalRecording、onLocalRecordBegin、onLocalRecording、onLocalRecordComplete。

功能修改

Windows&Mac：废弃 setRenderMode 接口，不再支持调用此接口修改视频的默认渲染方式（WebGL 或 Canvas 2D），SDK 内部会自动选择合适的渲染方式，以提高视频渲染性能。

功能改进

视频渲染性能优化。
升级底层库。Mac 下支持构建 ARM64 指令集的应用程序，发挥 M1 芯片优势，提升性能。

Version 10.3.402 @ 2022.08.12

问题修复

Window & Mac：调用混流接口后，混流事件返回 -3324 user id invalid 报错问题。

Version 10.3.401 @ 2022.07.20

改进

性能优化。
升级底层库。

Version 9.3.201 @ 2022.01.05

功能新增

Windows & Mac：新增 onSpeedTestResult 网速测试的结果回调。

改进

Windows & Mac：改进 startSpeedTest 开始进行网络测速。

版权所有：腾讯云计算（北京）有限责任公司

Windows & Mac：改进 muteLocalVideo 暂停/恢复发布本地的视频流，新增 streamType 参数。
Windows & Mac：改进 muteRemoteVideoStream 暂停接收指定的远端视频流，新增 streamType 参数。
Windows & Mac：改进 startScreenCapture 启动屏幕分享，新增 params 参数。

问题修复

Mac：Mac OS 12 新系统下的摄像头采集问题。
Windows & Mac：优化弱网调控策略，同场景下更流畅。
Windows：优化 AGC 算法，降低出现声音过小和声音过大的问题的概率。
Winodws：修复屏幕分享时采集帧率异常的问题。

Version 8.9.102 @ 2021.08.11

功能新增

Windows & Mac：onStatistics 回调新增字段 gatewayRtt onStatistics。

问题修复

Mac：修复特殊机型写日志引起 crash。
Mac：修复禁麦的操作使用 API 接口 setAudioCaptureVolume(0) 后，发现麦克风检测音量为 0。
Windows：性能优化，修复打开摄像头后黑屏。
Windows：修复屏幕捕获自动减低分辨率后不恢复。
Windows & Mac：其他 bug 修复。

Version 8.6.101 @ 2021.05.28

功能新增

Windows & Mac：新增接口，支持屏幕分享时屏蔽应用窗口：addExcludedShareWindow、removeExcludedShareWindow、removeAllExcludedShareWindow。
Windows & Mac：获取可共享的窗口列表接口 getScreenCaptureSources，返回值列表元素新增 isMinimizeWindow 字段。
Windows & Mac：支持构造函数传入参数。

问题修复

Windows：插件加载不支持中文路径问题。
Windows & Mac：修复 webgl context lost 问题。
Windows & Mac：开启双路编码，进入房间后，切换小画面视频流，本地显示的远端成员画面卡住问题。
Windows & Mac：在客户端进房拉流的时候出现远端成员画面先模糊一下，然后逐渐清晰问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 8.4.1 @ 2021.03.26

功能新增

Mac：开始支持采集 Mac 操作系统的输出声音startSystemAudioLoopback，也就是跟 Windows 端一样的 SystemLoopback 能力，该功能可以让 SDK 采集当前系统的声音，开启这个功能后，主播就可以很方便地向其他用户直播音乐或者电影文件。
Mac：系统音频采集回调 onSystemAudioLoopbackError，您可以获取系统音频驱动的运行情况。
Mac：屏幕分享开始支持本地预览功能，您可以通过一个小窗口像用户展示屏幕分享的预览内容。
全平台：支持美颜插件机制。

质量优化

全平台：优化 Music 模式下的声音质量，更加适合类似 cloubhouse 的语音直播场景。
全平台：优化音视频链路的网络抗性，在 70% 的极端差网络环境下，音视频依然较为流畅。
Windows：优化部分场景下的直播音质，大幅减少了声音损伤问题。
Windows：性能优化，在部分使用场景下的性能较旧版本有 20%-30% 的提升。

问题修复

Mac：修复 Mac mini (m1) 换到全屏分享后，再切回某个窗口，远端还是展示的全屏分享窗口的问题。
Mac：解决 Mac 下屏幕分享无高亮的问题（Mac 系统 11.1，10.14.5 不出现绿框；Mac 系统 10.3.2 会出现绿框，但放大窗口会闪烁)。
Mac：修复 Mac mini m1 获取分享屏幕列表 crash，针对底层 sourceName 为 null 时上层返回""的问题。
Mac：修复 Mac mini m1，getCurrentMicDevice 导致 crash (sourceName) 可能为空问题
Windows：修复 Windows Server 2019 Datacenter x64 系统上启动桌面分享 crash 的问题。
Windows：修复分享窗口的同时改变目标窗口大小会偶发分享意外终止的 BUG。
Windows：修复部分型号的摄像头采集不出画面的问题。

Version 8.2.7 @ 2021.01.06

新增

Windows & Mac：新增 switchRoom 切换房间。
Windows & Mac：新增 setLocalRenderParams 设置本地图像（主流）的渲染参数。
Windows & Mac：新增 setRemoteRenderParams 设置远端图像的渲染参数。
Windows & Mac：新增 startPlayMusic 启动播放背景音乐。
Windows & Mac：新增 stopPlayMusic 停止播放背景音乐。
Windows & Mac：新增 pausePlayMusic 暂停播放背景音乐。
Windows & Mac：新增 resumePlayMusic 恢复播放背景音乐。
Windows & Mac：新增 getMusicDurationInMS 获取背景音乐文件总时长，单位毫秒。

版权所有：腾讯云计算（北京）有限责任公司

Windows & Mac：新增 seekMusicToPosInTime 设置背景音乐播放进度。
Windows & Mac：新增 setAllMusicVolume 设置背景音乐的音量大小，播放背景音乐混音时使用，用来控制背景音音量大小。
Windows & Mac：新增 setMusicPlayoutVolume 设置背景音乐本地播放音量的大小。
Windows & Mac：新增 setMusicPublishVolume 设置背景音乐远端播放音量的大小。
Windows & Mac：新增 onSwitchRoom 切换房间回调。
Windows & Mac：新增 setRemoteAudioVolume 设置远程用户播放音量。
Windows & Mac：新增 snapshotVideo 视频画面截图。
Windows & Mac：新增 onSnapshotComplete 截图完成时回调。

改进

Windows & Mac：enterRoom 和 switchRoom 支持 string 类型 strRoomId。
Windows & Mac：其他 bug 修复。

Version 7.9.348 @ 2020.11.12

改进

Windows：修复录音路径设置不支持中文路径文件夹。
Windows：窗口捕获指定区域捕获支持抗遮挡。

Version 7.8.342 @ 2020.10.10

新增

Windows & Mac：新增 onAudioDeviceCaptureVolumeChanged 当前音频采集设备音量变化回调。
Windows & Mac：新增 onAudioDevicePlayoutVolumeChanged 当前音频播放设备音量变化回调。

Version 7.7.330 @ 2020.09.11

新增

Windows & Mac：新增 setAudioQuality 用于设置音频质量。

改进

Windows：优化在某些低端摄像头下 CPU 使用率过高的问题。
Windows：优化对多款 USB 摄像头和麦克风的兼容性，提升设备的打开成功率。
Windows：优化摄像头和麦克风设备的选择策略，避免由于摄像头或麦克风在使用中插拔导致的采集异常问题。

版权所有：腾讯云计算（北京）有限责任公司

Windows & Mac：其他 bug 修复。

Version 7.6.300 @ 2020.08.26

新增

Windows & Mac：新增 setCurrentMicDeviceMute 、getCurrentMicDeviceMute、setCurrentSpeakerDeviceMute、getCurrentSpeakerDeviceMute 用于控制 PC 的麦克风和扬声器。

Version 7.5.210 @ 2020.08.11

改进

Windows & Mac：修复 SDK 回调乱序问题。
Windows & Mac：解决切换渲染模式导致崩溃的问题。
Windows & Mac：修复某些分辨率渲染失败的问题。
Windows & Mac：其他 bug 修复。

Version 7.4.204 @ 2020.07.01

改进

Windows：优化 Windows 平台的回声抵消（AEC）效果。
Windows：增强 Windows 平台的摄像头采集的设备兼容性。
Windows：增强 Windows 平台的音频设备（麦克风和扬声器）的设备兼容性。
Windows：修复 Windows 版本 onPlayAudioFrame 回调的 UserID 不正确的问题。
Windows：64 位支持系统混音

Version 7.2.174 @ 2020.04.20

改进

Mac：修复 Mac 偶现本地自定义渲染分辨率不一致问题。
Windows：优化 Windows 端 getCurrentCameraDevice 逻辑，在未使用摄像头时，返回第一个设备作为默认设备。
Windows：修复高亮窗口在屏幕分享时显示为灰屏的问题。
Windows：修复 Win10 系统获取屏幕分享缩略图偶现卡死问题。
Windows & Mac：修复切换角色时，自定义流 ID 偶现未及时生效的问题。
Windows & Mac：修复屏幕分享设置编码参数不生效的问题。

版权所有：腾讯云计算（北京）有限责任公司

Windows：修复 Windows 端屏幕分享后，webrtc 要很久才能看到画面的问题。

Version 7.1.157 @ 2020.04.02

新增

支持使用 主路 进行 屏幕分享。

改进

优化 混流预设模版 易用性。
优化 混流，提升成功率。
优化 Windows 屏幕分享。

Version 7.0.149 @ 2020.03.019

新增

trtc.d.ts 文件，方便使用 typescript 的开发者。

发布日志（Web）

最近更新时间：2024-07-22 16:45:43

版本号 `major.minor.patch` 具体规则如下：

major：主版本号，如有重大版本重构则该字段递增，通常各主版本间接口不兼容。

minor：次版本号，各次版本号间接口保持兼容，如有接口新增或优化则该字段递增。

patch：补丁号，如有功能改善或缺陷修复则该字段递增。

注意：

建议您及时更新至最新版本，以便获得更好的产品稳定性及在线支持。

版本升级注意事项请参见：升级指引。

Version 5.7.0 @2024.07.19

Feature

支持 基础美颜插件。

支持辅流 sei 消息收发。

Improvement

使用 captureElement 参数采集屏幕分享时，预览框只显示当前页面。

规避 Android Webview 插拔耳机可能导致无声的问题，chrome issue。

提升 Android 设备 h264 支持度检测的准确性。

Bug Fixed

修复在支持 h264 解码，但不支持 h264 编码的环境中，远端视频黑屏的问题。

Version 5.6.3 @2024.06.28

Feature

混音插件支持监听播放进度回调。

虚拟背景插件支持设置虚化程度。

Improvement

提升 iOS 音频自动播放的成功率。

提升 iOS 通话打断后，恢复正常通话的成功率。

Bug Fixed

修复 Chrome 91 以下版本偶现重连的问题。

修复 mute/unmute audio 后，音频播放出现延迟的问题。

修复特定场景下，误抛远端推屏幕分享事件的问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 5.6.2 @2024.06.07

Improvement

优化观众拉流通话体验，减少卡顿。

提升断网重连成功率。

Bug Fixed

修复移动端偶现自动播放失败后，无法恢复播放的问题。

修复 Mac Safari 采集 1920 * 1280 分辨率失败的问题。

修复偶现拉流无声的问题。

修复特定调用顺序下 muteRemoteAudio 误报 abort error 的问题。

Version 5.6.1 @2024.05.23

Bug Fixed

修复在手动拉音频的场景下，偶现无声的问题。

Version 5.6.0 @2024.05.17

Breaking Changed

关闭默认进房自动拉视频流，由业务侧按需调用 trtc.startRemoteVideo 来拉流播放视频。参见：升级指引。

Feature

支持收发自定义消息，参见：trtc.sendCustomMessage。

Bug Fixed

修复 Chrome 123 偶现拉流异常的问题。

修复在 iOS 12.0 版本进房失败的问题。

修复编码镜像偶现失效的问题。

修复混流转推 CDN 收不到流的问题。

修复混音插件 loop 不生效的问题。

修复在混用 roomId 和 strRoomId 多次进房后，进房异常的问题。

Version 5.5.2 @2024.04.29

Improvement

优化背景虚化插件资源加载逻辑，可通过自行部署模型文件加快加载速度，参见：开启虚拟背景。

规避 iOS 17 播放视频闪烁的问题，webkit bug。

版权所有：腾讯云计算（北京）有限责任公司

Bug Fixed

修复 Android qq 浏览器偶现 startLocalVideo 失败的问题。

修复 Chrome 123+ 版本订阅小流时，偶现报错的问题。

修复偶现播放黑屏问题。

Version 5.5.1 @2024.04.12

Improvement

提升弱网重连成功率。

提升设备恢复采集成功率。

Bug Fixed

修复 iOS 15.1 版本播放远端视频没有显示的问题。

修复使用 updateScreenShare 重新推屏幕分享后，从辅流变成主流的问题。

修复 high audio profile 码率不符合预期的问题。

修复水印插件传入 base64 无法渲染的问题。

Version 5.5.0 @2024.03.29

Improvement

优化移动端 AI 降噪插件的兼容性。

提升设备异常自动恢复采集的成功率。

Bug Fixed

修复 iOS 16 中偶现本地视频播放黑屏的问题。

修复 iOS 14 偶现无声的问题。

修复 startLocalAudio 偶现报错问题。

Version 5.4.3 @2024.03.15

Feature

支持 混音插件 传入 MediaStreamTrack。

支持 trtc.getAudioTrack 获取屏幕分享音频 MediaStreamTrack。

Bug Fixed

修复偶现 setRemoteAudioVolume 0 不生效的问题。

修复 updateScreenShare({ publish: true }) 后，屏幕分享音频没有推流的问题。

修复 Safari 中无法开启虚拟背景的问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 5.4.2 @2024.03.01

Bug Fixed

修复偶现 startRemoteVideo 失败的问题。

修复偶现取消推流失败的问题。

修复部分低端机型出现音画不同步的问题。

修复 Nginx 代理无法进房的问题。

Version 5.4.1 @2024.02.05

Improvement

优化重连逻辑，提升弱网重连成功率。

Bug Fixed

修复 updateLocalVideo 导致 mirror 失效的问题。

修复 CONNECTION_STATE_CHANGED 事件没有 CONNECTING 状态的问题。

Version 5.4.0 @2024.01.16

Feature

支持获取视频帧，参考：getVideoSnapshot()。

支持 mute video 时设置图片垫片，参考：startLocalVideo() 的 mute 参数。

支持只拉可视区域视频流，非可视区域的视频不拉流。参考：多人视频通话最佳实践。

支持屏幕分享采集页面某个 DOM 元素。参考：startScreenShare()。

Improvement

优化进房逻辑，降低进房耗时。

优化笔记本盒盖重新打开时的重连逻辑。

Bug Fixed

修复 Chrome 69 以下版本偶现推流失败的问题。

修复 iOS 13 & 14 1080P推流黑屏问题。

Version 5.3.2 @2023.12.26

Feature

支持水印插件，参考：开启水印。

支持编码翻转，参考：startLocalVideo() 的 mirror 参数。

版权所有：腾讯云计算（北京）有限责任公司

Improvement

优化音视频编码稳定性，提升编码质量。

Bug Fixed

修复 CDNStreaming 插件的已知问题。

修复 setRemoteAudioVolume 0 后，音量事件返回的音量值为 0 的问题。

修复部分外接麦克风开启降噪偶现丢音的问题。

Version 5.3.1 @2023.12.08

Bug Fixed

修复混音插件异常问题。

修复 Chrome 74 以下版本无法进房问题。

修复开启 AI 降噪时，部分音频接口表现不符合预期的问题。

修复多 trtc 实例场景下，销毁其中一个实例，其他实例收不到 DEVICE_CHANGED 事件的问题。

Version 5.3.0 @2023.12.01

Feature

支持 SEI 消息收发，可用于实现歌词同步、直播答题等功能，参考 sendSEIMessage。

支持动态开关大小流，参考 updateLocalVideo 的 option.small 参数。

支持静音推流，参考 startLocalAudio() 的 mute 参数。

支持切换角色时更新 privateMapKey，参考 switchRole 的 privateMapKey 参数。

新增 TRTC.EVENT.TRACK 事件。

Improvement

优化进房流程，缩短进房耗时。

优化高分辨率通话场景、低版本 Android Chrome 设备的编码质量。

优化获取设备逻辑，在无媒体访问权限的情况下，SDK 可能会暂时请求获取媒体权限，以保证能正常获取到媒体设备，随后会释放媒体设备。

优化混音插件的 url 参数的解析逻辑。

提升 AI 降噪插件的降噪效果。

Bug Fixed

修复 Android Chrome 无法编码 120p 的问题。

修复屏幕分享不推流场景下，停止屏幕分享会导致摄像头推流停止的问题。

修复 CDN 混流插件参数失效问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 5.2.1 @2023.11.08

Feature

startLocalAudio & updateLocalAudio 接口新增 captureVolume 参数，支持调节麦克风采集音量。

移动端设备支持 TRTC.EVENT.DEVICE_CHANGED 事件，可实现在连接耳机时，自动切换到耳机麦克风。参考：处理设备插拔。

Bug Fixed

修复切换麦克风后无声的问题。

修复停止屏幕分享时，TRTC.EVENT.PUBLISH_STATE_CHANGED 事件的 mediaType 类型错误的问题。

Version 5.2.0 @2023.10.31

Feature

支持背景虚化插件。

新增 TRTC.EVENT.STATISTICS 事件，可用于获取通话相关统计数据。

Improvement

提升设备采集成功率。

优化“画中画模式”的镜像处理逻辑。

当用户系统拒绝浏览器授权时，可调用 RtcError.handler() 跳转至系统授权设置，引导用户快速打开授权。参考错误码：5302。

Bug Fixed

修复低版本 Chrome 偶现拉流无声问题。

Version 5.1.3 @2023.09.11

Feature

trtc.setRemoteAudioVolume 支持设置音量大于 100，可以增益远端播放音量。

Improvement

规避 iOS 15.1 切换摄像头导致页面崩溃的 iOS bug。

Bug Fixed

修复 Firefox 取消推流后再重新推流失效的问题。

修复 Firefox 采集特定分辨率视频失败的问题，例如：640 * 360。

修复偶现远端视频未播放的问题。

Version 5.1.2 @2023.08.25

Improvement

降低进房耗时。

Bug Fixed

修复 webpack 打包构建 trtc.esm.js 偶现报错的问题。

修复 startLocalAudio 传入自定义采集 audioTrack 不生效的问题。

Version 5.1.1 @2023.08.18

Improvement

默认 video profile 变更为 480p_2，在保障画质的情况下，降低上行带宽消耗。

规避 Android Chrome 115 低于 360p 分辨率偶现无法编码的 Chrome Bug。

Bug Fixed

修复 Chrome 57、iOS 12 无法进房推拉流的问题。

修复仪表盘视频码率异常问题。

Version 5.1.0 @2023.08.11

Breaking Change

限制 trtc.enterRoom 接口的 roomId 参数为 number 类型，不再支持传入 string 类型。若要使用字符串房间号，请使用 strRoomId 参数。升级时需注意，详情参考：升级指引。

Feature

支持背景音乐插件，参考教程：实现背景音乐。

支持 AI 降噪插件，参考教程：实现 AI 降噪。

支持 CDN 混流插件，参考教程：实现云端混流及转推 CDN。

Bug Fixed

修复设置屏幕分享采集分辨率不生效的问题。

修复偶现播放远端屏幕分享失败的问题。

Version 5.0.3 @2023.07.31

Improvement

优化重连机制，提升网络连接稳定性。

Bug Fixed

修复 trtc.stopRemoteVideo 停止播放主流时，辅流也被停止播放的问题。

Version 5.0.2 @2023.07.21

Improvement

优化多人音视频场景下的性能表现及弱网抗性。

优化设备采集逻辑，规避部分联想设备无法开启摄像头的问题。

优化屏幕分享采集参数，规避长时间屏幕分享偶现采集掉帧的问题。

Bug Fixed

修复小流码率设置不生效的问题。

修复 systemAudio 参数不生效的问题。

修复偶现远端用户关闭屏幕分享后，video 标签未销毁的问题。

Version 5.0.1 @2023.06.25

Feature

支持同时在多个位置播放视频

Bug Fixed

修复点击浏览器悬浮窗口关闭屏幕分享后，无法重新开启屏幕分享的问题。

Version 5.0.0 @2023.05.26

TRTC Web SDK 新架构版本，提供扁平化的接口，大幅简化 API，降低接入成本。新 API 的特点：

更易于接入的扁平化 API。

更好的稳定性。

更好的性能表现。

发布日志（Flutter）

最近更新时间：2024-06-24 15:22:32

Version 2.8.2 @ 2024.6.20

新特性

Android&iOS：新增 onAudioRouteChanged 回调。
添加新的音频路由如 TRTC_AUDIO_ROUTE_WIREDHEADSET，详情请见： TRTCCloudDef 。

缺陷修复

Windows：修复 onDeviceChange 回调无法收到的问题。

Version 2.8.1 @ 2024.6.14

新特性

Android&iOS：新增 setVoicePitch 接口
添加新的变声特效 Studio 2，详情请见 TXVoiceReverbType 。

Version 2.8.0 @ 2024.5.21

新特性

Windows：新增 getScreenCaptureSources 和 selectScreenCaptureTarget 接口，支持 Windows 屏幕分享。

Version 2.7.9 @ 2024.5.20

依赖更新

Android SDK 更新至 11.8.0.14188
iOS SDK 更新至 11.8.15687

Version 2.7.8 @ 2024.4.18

依赖更新

版权所有：腾讯云计算（北京）有限责任公司

Windows SDK 更新至 11.7.0.14863。
MacOS SDK 更新至 11.7.15304。
Android SDK 更新至 11.7.0.13910。
iOS SDK 更新至 11.7.15343。

新特性

Android&iOS：新增 createSubCloud 和 destroySubCloud API。

缺陷修复

Windows：修复了 onRecvCustomCmdMsg 回调数据解析错误问题。

Version 2.7.7 @ 2024.4.3

缺陷修复

Web：修复了 switchRole 调用无效的问题。

Version 2.7.6 @ 2024.2.29

缺陷修复

Windows：修复了 Window 库中的数据上传问题。

Version 2.7.5 @ 2024.2.29

新特性

Windows：更新 startSystemAudioLoopback 接口以支持收集特定应用程序的声音。

Version 2.7.4 @ 2024.2.29

新特性

Windows：新增 startSystemAudioLoopback 接口，支持系统音频捕获，如第三方音乐播放器。

Version 2.7.3 @ 2024.1.15

缺陷修复

Web：修复了由于引入 dart:ffi 导致的 Web 平台编译失败问题。

Version 2.7.2 @ 2024.1.11

依赖更新

Window：升级客户端 SDK 版本至 11.4.0。

Version 2.7.1 @ 2023.12.28

缺陷修复

macOS：修复了在 TexTure 渲染过程中偶尔出现的崩溃问题。

新特性

Android&iOS：新增 setAudioFrameListener API。

Version 2.7.0 @ 2023.12.13

新特性

Web：将 Web TRTC SDK 升级到最新版本（v5），以实现更稳定的功能。

缺陷修复

Web：修复了从 Android 和 iOS 设备进行屏幕共享时，无法在 Web 版本中查看的问题。

Version 2.6.0 @ 2023.11.21

缺陷修复

Web：修复了在调用 getCurrentDevice 和 getDevicesList API 时出现的异常问题。

Version 2.5.9 @ 2023.9.28

新特性

Android&iOS：新增 startPublishMediaStream API。

Android&iOS：新增 updatePublishMediaStream API。
Android&iOS：新增 stopPublishMediaStream API。

Version 2.5.8 @ 2023.9.13

新特性

将文档跳转链接替换为国际站点。

Version 2.5.7 @ 2023.9.11

依赖更新

Android SDK 更新至 11.4.0.13189。
iOS SDK 更新至 11.4.14445。
macOS SDK 更新至 11.4.14445。

Version 2.5.6 @ 2023.8.9

缺陷修复

Windows：优化 Dart 代码风格。

Version 2.5.5 @ 2023.8.2

依赖更新

Android SDK 更新至 11.3.0.13200。
iOS SDK 更新至 11.3.14354。
macOS SDK 更新至 11.3.14333。

Version 2.5.4 @ 2023.7.10

依赖更新

Android SDK 更新至 11.3.0.13176。
iOS SDK 更新至 11.3.14333。

版权所有：腾讯云计算（北京）有限责任公司

Version 2.5.3 @ 2023.6.27

依赖更新

Android SDK 更新至 11.2.13145。
iOS SDK 更新至 11.2.14217。

Version 2.5.2 @ 2023.6.16

缺陷修复

Windows：修复了 startSpeedTest 函数返回过长数据事件且无响应的问题。
Web：将 setAudioPlayoutVolume 和 getAudioPlayoutVolume 标记为不可用。

Version 2.5.1 @ 2023.6.2

新特性

Windows&Mac&Web：恢复对 Windows&Mac&Web 平台的支持。
iOS：新增 setSystemAudioLoopbackVolume API，支持在屏幕共享时调整系统音量。

缺陷修复

iOS：修复了在特定场景下 TRTCCloudVideoView 偶尔出现的内存泄漏问题。

依赖更新

Windows&Mac：将客户端 SDK 版本升级至 11.1.0。

Version 2.5.0 @ 2023.5.4

功能优化

暂时移除了 Web、MacOS 和 Windows 平台的支持。

Version 2.4.6 @ 2023.5.4

依赖更新

Android SDK 更新至 11.1.0.13111。

iOS SDK 更新至 11.1.14143。

Version 2.4.5 @ 2023.3.14

新特性

Android：新增 startSystemAudioLoopback 功能。

Version 2.4.4 @ 2023.3.6

功能优化

优化部分代码。

Version 2.4.2 @ 2023.1.9

依赖更新

Android SDK 更新至 10.9.0.24004。

Version 2.4.2 @ 2023.1.9

缺陷修复

修复 iOS 平台 snapshotVideo 为空的问题。

Version 2.4.1 @ 2022.12.1

依赖更新

Android SDK 更新至 10.8.0.13065。
iOS SDK 更新至 10.8.12025。

Version 2.4.0 @ 2022.10.31

功能优化

优化部分代码。

Version 2.3.9 @ 2022.10.18

依赖更新

Android SDK 更新至 10.7.0.13053。
iOS SDK 更新至 10.7.11936。

Version 2.3.8 @ 2022.9.20

功能优化

优化 Windows 平台，自动添加相关 DLL 文件。

Version 2.3.7 @ 2022.9.16

缺陷修复

修复 Web 平台 "Can't use 'Function' as a name" 的问题。

Version 2.3.6 @ 2022.9.5

功能优化

优化部分代码。

Version 2.3.5 @ 2022.8.23

依赖更新

Android SDK 更新至 10.3.0.11196。
iOS SDK 更新至 10.3.12231。

Version 2.3.4 @ 2022.7.21

新特性

更新 Windows、MacOS 和 Web 平台，支持纯视频模式。
setMixTranscodingConfig: 仅支持混合视频。

Version 2.3.2 @ 2022.7.14

依赖更新

Android&iOS：SDK 更新至 10.3。

Version 2.3.1 @ 2022.6.23

功能优化

优化部分代码。

Version 2.3.0 @ 2022.6.20

依赖更新

Android&iOS：SDK 更新至 10.1 版本的 LiteAVSDK_Professional。

新特性

支持第三方美颜。

Version 2.2.4 @ 2022.5.11

功能优化

优化 setMixTranscodingConfig 接口。

Version 2.2.3 @ 2022.5.7

依赖更新

Android SDK 设置为 9.9.0.11820。
iOS SDK 设置为 9.5.11346。

Version 2.2.2 @ 2022.5.5

功能优化

更新日志模块。

Version 2.2.1 @ 2022.4.21

功能优化

PlatformView 支持 'onTap' 事件。

Version 2.2.0 @ 2022.3.30

依赖更新

将 Android 和 iOS SDK 更新为 TXLiteAVSDK_Live。

Version 2.1.7 @ 2022.3.22

缺陷修复

修复 2.1.6 版本 iOS 视频渲染的问题。

Version 2.1.6 @ 2022.3.17

功能优化

优化 iOS 纹理。

Version 2.1.5 @ 2022.3.11

功能优化

更新 remoteView 调整参数顺序。

Version 2.1.4 @ 2022.3.10

功能优化

更新文档。

版权所有：腾讯云计算（北京）有限责任公司

Version 2.1.3 @ 2022.3.10

功能优化

更新文档。

Version 2.1.2 @ 2022.3.7

依赖更新

Android SDK 更新至 9.5.11207。
iOS SDK 更新至 9.5.11207。

Version 2.1.1 @ 2022.2.15

缺陷修复

修复 onSpeedTest 回调数据错误的问题。

Version 2.1.0 @ 2022.1.25

功能优化

优化初始化时机。

Version 2.0.9 @ 2022.1.13

缺陷修复

修复未找到 web 文件夹的问题。

依赖更新

Android & iOS SDK 更新至 9.5。

Version 2.0.7 @ 2022.1.10

功能优化

解决警告。

Version 2.0.6 @ 2022.1.10

功能优化

删除 web 文件夹。

Version 2.0.5 @ 2022.1.10

功能优化

优化文档显示。

Version 2.0.1 @ 2022.1.7

功能优化

将视频纹理渲染封装成 PlatformView。

Version 2.0.0 @ 2022.1.4

功能优化

支持流畅的 Web。

Version 1.3.1 @ 2022.1.4

功能优化

优化部分文档。

Version 1.3.0 @ 2021.11.22

功能优化

Android 视频 SurfaceView 渲染更改为 GLSurfaceView。

Version 1.2.9 @ 2021.11.15

依赖更新

将底层 Android SDK 版本更新为9.3.10765。

Version 1.2.8 @ 2021.11.15

功能优化

Android 的底层 GLSurfaceView 被 TextureView 替换，updateView 仅支持 TextureView。

Version 1.2.7 @ 2021.11.5

功能优化

屏幕共享支持指定大小的流。

Version 1.2.6 @ 2021.11.1

缺陷修复

修复前一版本导致的 iOS 视频渲染问题。

Version 1.2.5 @ 2021.10.27

新特性

Android 视频渲染支持混合集成模式。默认模式为虚拟显示模式。TRTCCloudVideoView 的视图模式传输给 TRTCCloudDef.TRTC_VideoView_Model_Hybrid。

Version 1.2.4 @ 2021.9.29

功能优化

Fluent 的 Windows 端支持纹理渲染。
在线提供 Fluent 英文文档。

Version 1.2.3 @ 2021.9.28

新特性

Android 支持 updateLocalView 和 updateRemoteView 接口。

Version 1.2.2 @ 2021.9.10

缺陷修复

修复 Android 纹理渲染多人切换视频内存增长问题。

Version 1.2.1 @ 2021.9.10

功能优化

优化部分功能。

Version 1.2.0 @ 2021.9.10

新特性

指定美颜、设备和音效管理模块的返回值。

Version 1.1.9 @ 2021.8.19

缺陷修复

修复 iOS 和 MacOS 截图失败等问题。

Version 1.1.8 @ 2021.8.12

功能优化

Android 纹理渲染兼容 meglhelper。

Version 1.1.7 @ 2021.8.10

版权所有：腾讯云计算（北京）有限责任公司

缺陷修复

修复 Android 缺少 businessInfo 字段的问题。

Version 1.1.6 @ 2021.8.3

功能优化

优化部分功能。

Version 1.1.5 @ 2021.8.3

缺陷修复

修复在 iOS 和 MacOS 的发布模式下播放音乐时出现特殊字符串参数导致的崩溃问题。

Version 1.1.4 @ 2021.8.3

缺陷修复

修复 iOS 和 MacOS 下特殊字符串参数导致的崩溃问题。

Version 1.1.3 @ 2021.7.27

缺陷修复

修复 iOS 和 MacOS 下 onspeedtest 回调没有网络质量数据的问题。
修复 Android 纹理渲染 meglcore 为空的问题。

Version 1.1.2 @ 2021.7.23

缺陷修复

修复 iOS 和 MacOS 不支持辅助流渲染的问题。

Version 1.1.1 @ 2021.7.21

新特性

新的纹理渲染形式。

Version 1.1.0 @ 2021.7.14

缺陷修复

修复 Android 停止远程视图后再次启动远程视图视频无法渲染的问题。

Version 1.0.9 @ 2021.6.30

新特性

Android 和 iOS 支持本地录制 startLocalRecording 。

Version 1.0.8 @ 2021.6.28

新特性

支持 Windows 和 macOS。目前仅支持音频相关接口，视频渲染暂不支持。

Version 1.0.5 @ 2021.6.9

缺陷修复

修复 Android 视频视图销毁后出现的平台异常错误。

Version 1.0.4 @ 2021.6.2

新特性

iOS 添加 updateLocalView 和 updateRemoteView 接口。

Version 1.0.3 @ 2021.6.1

缺陷修复

修复 Android 关闭麦克风后设置音频路由无效的问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.0.2 @ 2021.5.27

功能优化

修改文档注释。

Version 1.0.1 @ 2021.4.28

缺陷修复

修复 Android 房间 ID 超过 2147483647 时无法进入房间的问题。值范围支持：1 - 4294967294。

Version 1.0.0 @ 2021.4.23

功能优化

升级 Flutter 2.0，支持零安全。

API-Example
iOS

最近更新时间：2024-07-05 19:30:25

本文介绍了如何快速运行腾讯云 TRTC iOS SDK Demo。

环境准备

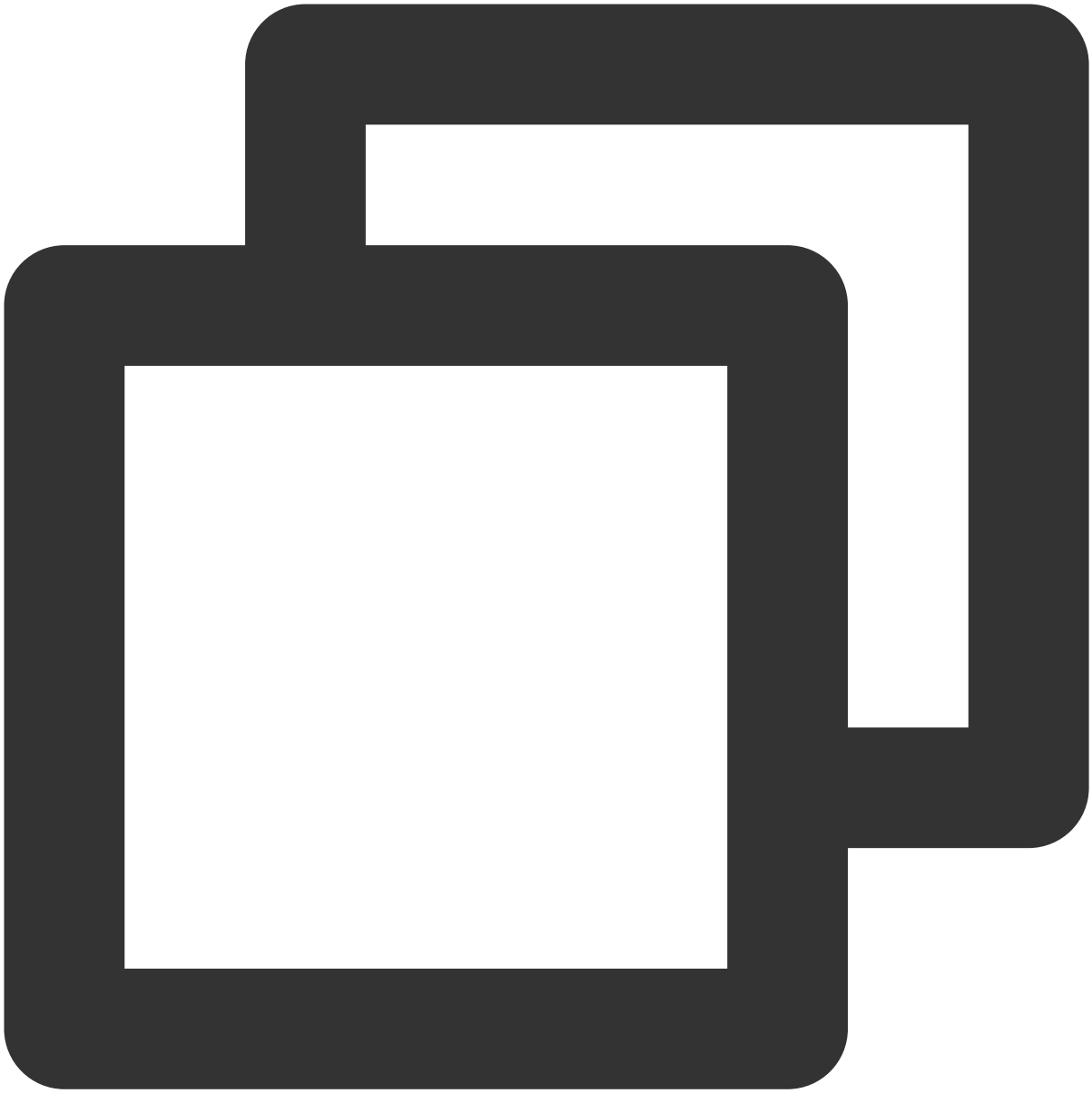
Xcode 11.0及以上版本
确保您的项目已设置有效的开发者签名
Qt Creator 4.13.3 (Mac) 及以上版本

操作步骤

步骤1. 下载 Demo

您可以在 github 下载 iOS 平台的示例代码，或者在终端运行以下命令：

版权所有：腾讯云计算（北京）有限责任公司



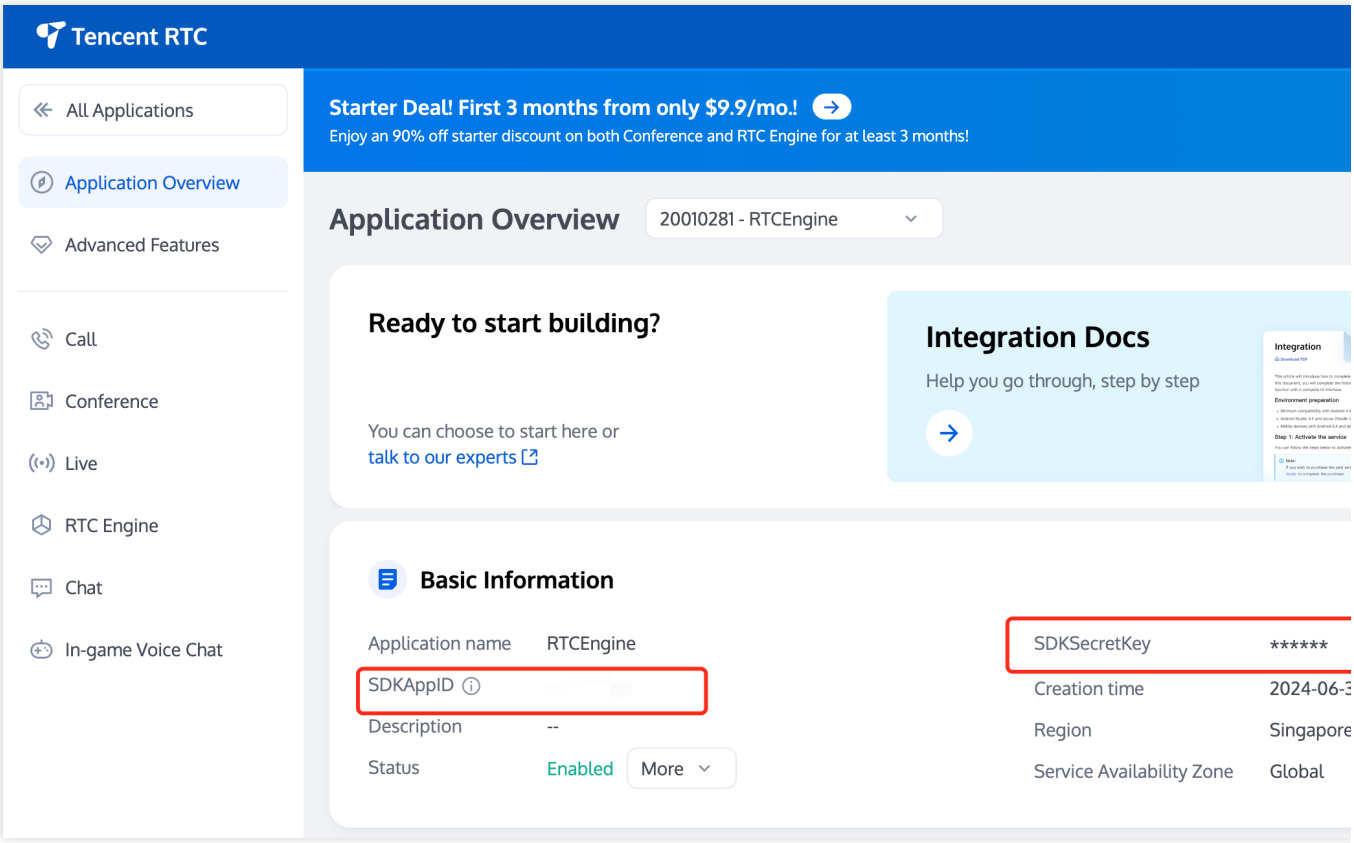
```
git clone https://github.com/Tencent-RTC/TRTC_iOS.git
```

在终端窗口中进入目标工程目录后执行 `pod init` 命令即可，导入iOS平台SDK 中其他步骤已经在示例代码中操作过。

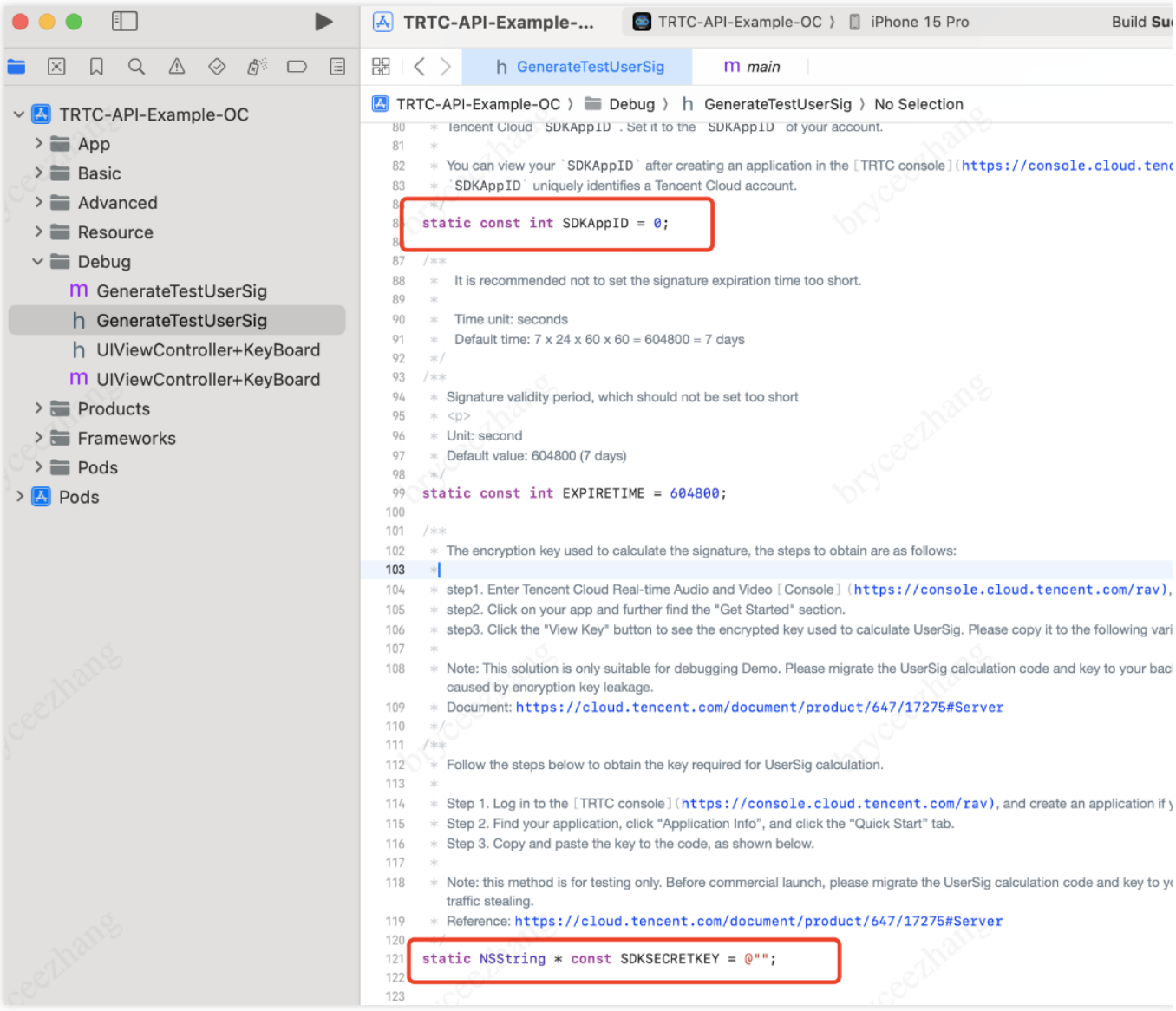
步骤2. 配置 Demo

1. 登录 TRTC Console，单击 **Create Application** 。如果您已经完成创建，可以跳过该操作。

2. 在创建应用成功后，您可以在基本信息中获取到您的 **SDKAppID** 和 **SDKSecretKey** 。



3. 您需要将 `TRTC-API-Example-XX/Debug` 目录下 `GenerateTestUserSig.h` 或 `GenerateTestUserSig.swift` 中的 `SDKAPPID` 和 `SDKSECRETKEY` 替换为您在**步骤2**中获取到的值。



注意：

本文使用的生成 UserSig 的方案是在本地配置 SDKSecretKey，该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。正确的 UserSig 签发方式是在您的服务端集成 服务端生成 UserSig，当用户进房时：

发起 http 请求到您的服务端。

服务端生成一个 UserSig。

将其返回给用户供进房使用。

若要将 Demo 部署到公网体验，需要通过 HTTPS 协议，即 `https://域名/xxx` 访问，原因可参见文档 页面访问协议限制说明。

步骤3. 跑通 Demo

使用 XCode（11.0及以上的版本）打开源码目录下的 `TRTC-API-Example-OC.xcworkspace` 工程，编译并运行 TRTC-API-Example 工程即可。

步骤4. 体验 Demo

您可以选择自己感兴趣的功能进行体验。



常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com 。

版权所有：腾讯云计算（北京）有限责任公司

Mac

最近更新时间：2024-07-09 15:09:33

本文介绍了如何快速运行腾讯云 TRTC Mac SDK Demo。

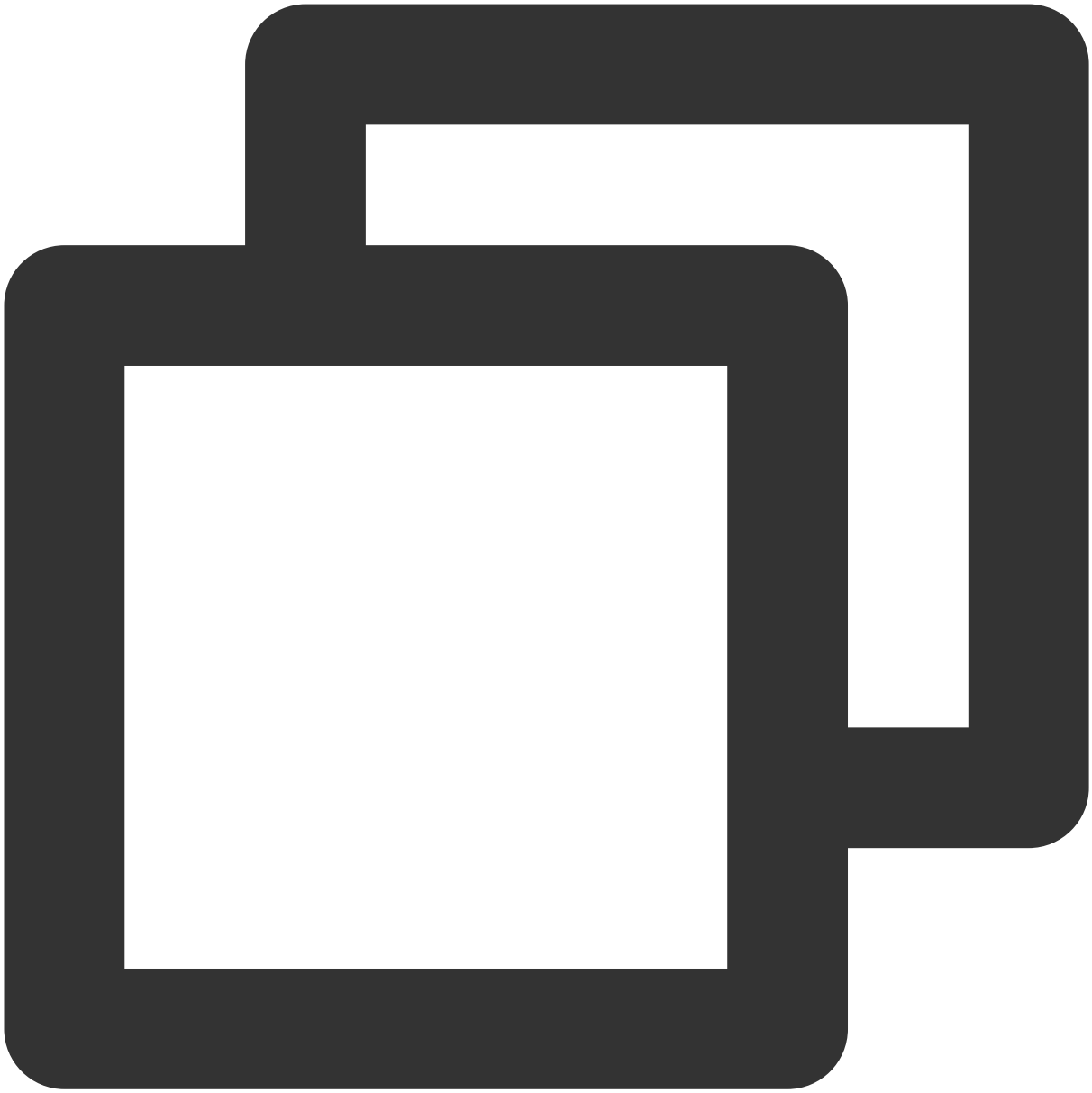
环境准备

Xcode 11.0及以上版本。
确保您的项目已设置有效的开发者签名。
Qt Creator 4.13.3 (Mac) 及以上版本。

操作步骤

步骤1. 下载 Demo

您可以在 github 下载 macOS 平台的示例代码，或者在终端运行以下命令：



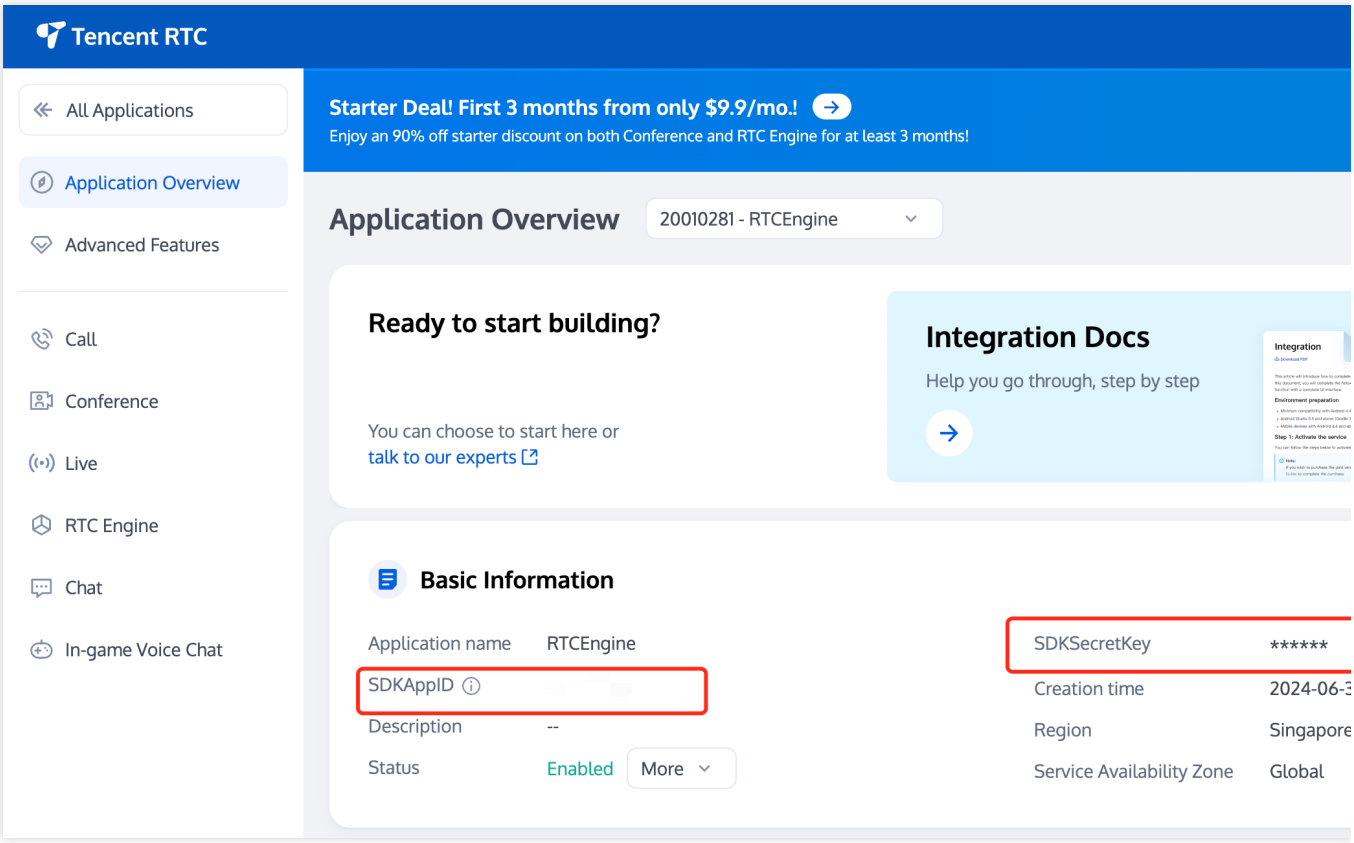
```
git clone https://github.com/Tencent-RTC/TRTC_Mac.git
```

在终端窗口中进入目标工程目录后执行 `pod init` 命令即可，导入macOS的SDK 中其他步骤已经在示例代码中操作过。

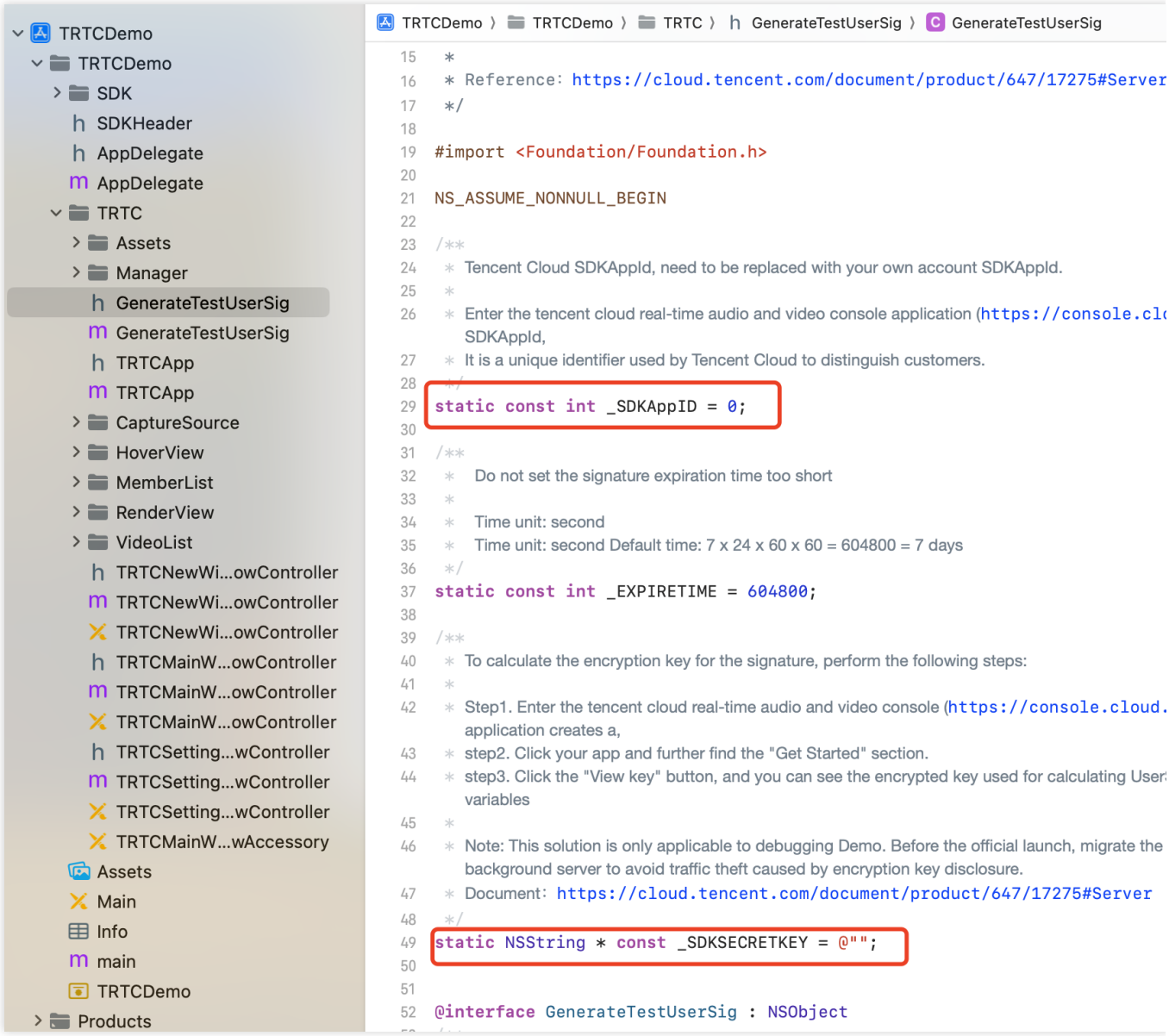
步骤2. 配置 Demo

1. 登录 TRTC Console，单击 **Create Application** 。如果您已经完成创建，可以跳过该操作。

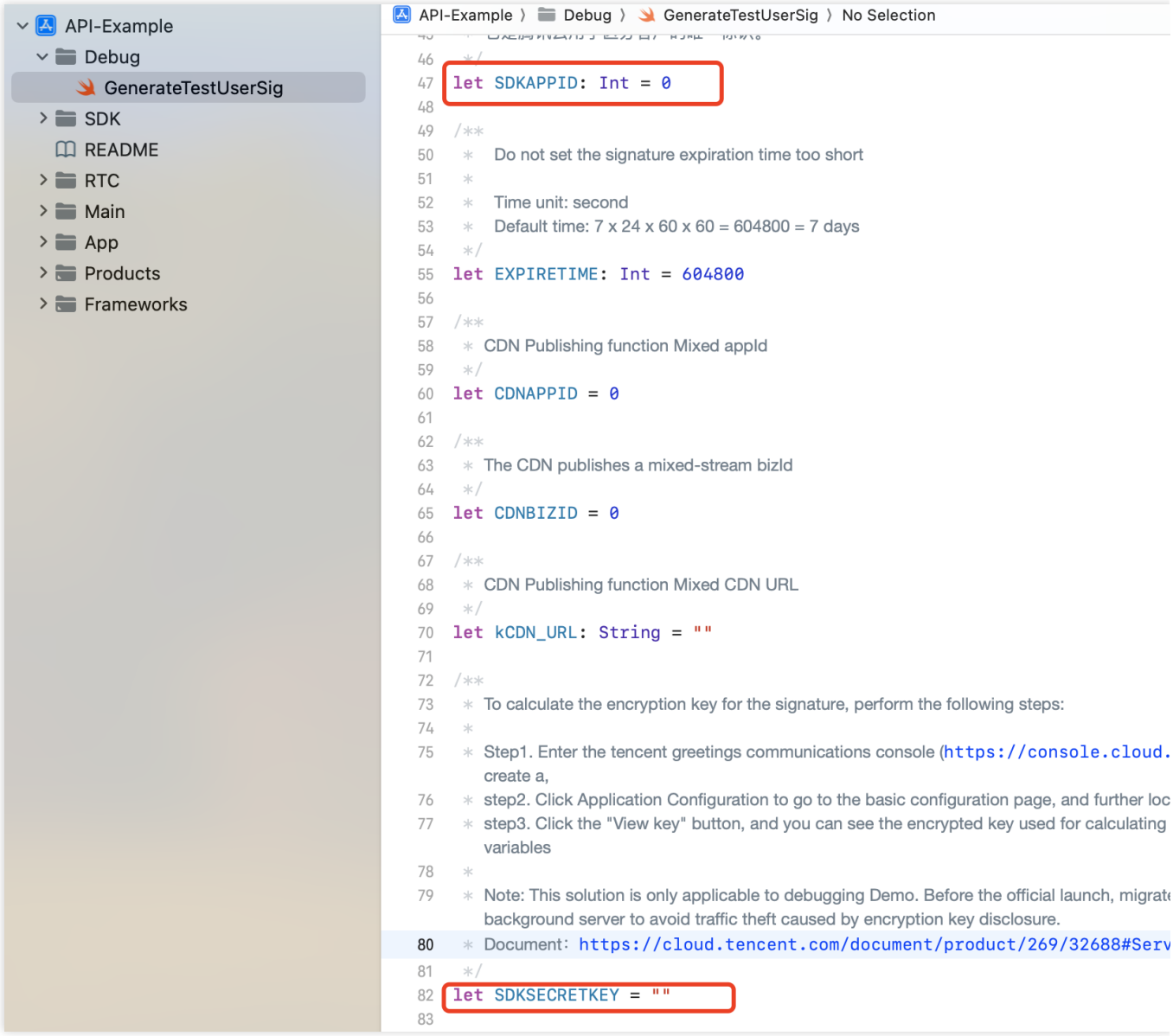
2. 在创建应用成功后，您可以在基本信息中获取到您的 **SDKAppID** 和 **SDKSecretKey** 。



3. 如果您选择的是 **OCDemo** ，将 `TRTCDemo/TRTC` 目录下 `GenerateTestUserSig.h` 中的 **SDKAppID** 和 **SDKSecretKey** 修改为在**步骤2**中获得的值。



如果您选择的是 **SwiftDemo** 将 `API-Example/Debug` 目录下 `GenerateTestUserSig.swift` 中的 **SDKAppID** 和 **SDKSecretKey** 修改为在**步骤2**中获得的值。



注意：

本文使用的生成 UserSig 的方案是在本地配置 SDKSecretKey，该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。正确的 UserSig 签发方式是在您的服务端集成 服务端生成 UserSig，当用户进房时：

发起 http 请求到您的服务端。

服务端生成一个 UserSig。

将其返回给用户供进房使用。

若要将 Demo 部署到公网体验，需要通过 HTTPS 协议，即 `https://域名/xxx` 访问，原因可参见文档 页面访问协议限制说明。

步骤3. 跑通 Demo

使用 XCode（11.0及以上的版本）打开源码目录下的 `TRTCDemo.xcworkspace/API-Example.xcworkspace` 工程，编译并运行 TRTC-API-Example 工程即可。

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com 。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2024-07-05 19:30:25

本文介绍了如何快速运行腾讯云 TRTC Android SDK Demo。

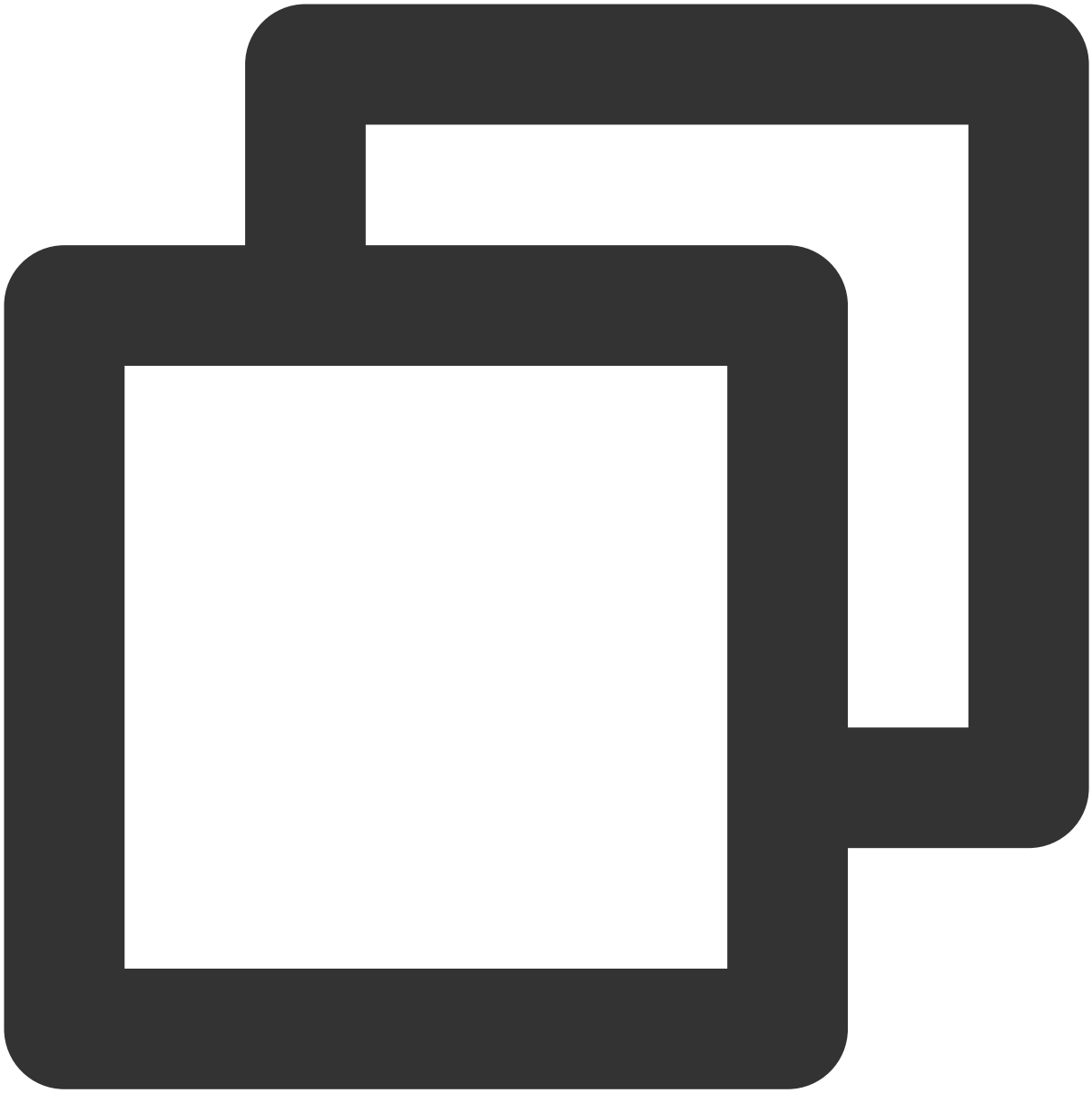
环境准备

最低兼容 Android 4.1（SDK API Level 16），建议使用 Android 5.0 （SDK API Level 21）及以上版本。
Android Studio 3.5 及以上版本。
App 要求 Android 4.1 及以上设备。

操作步骤

步骤1. 下载 Demo

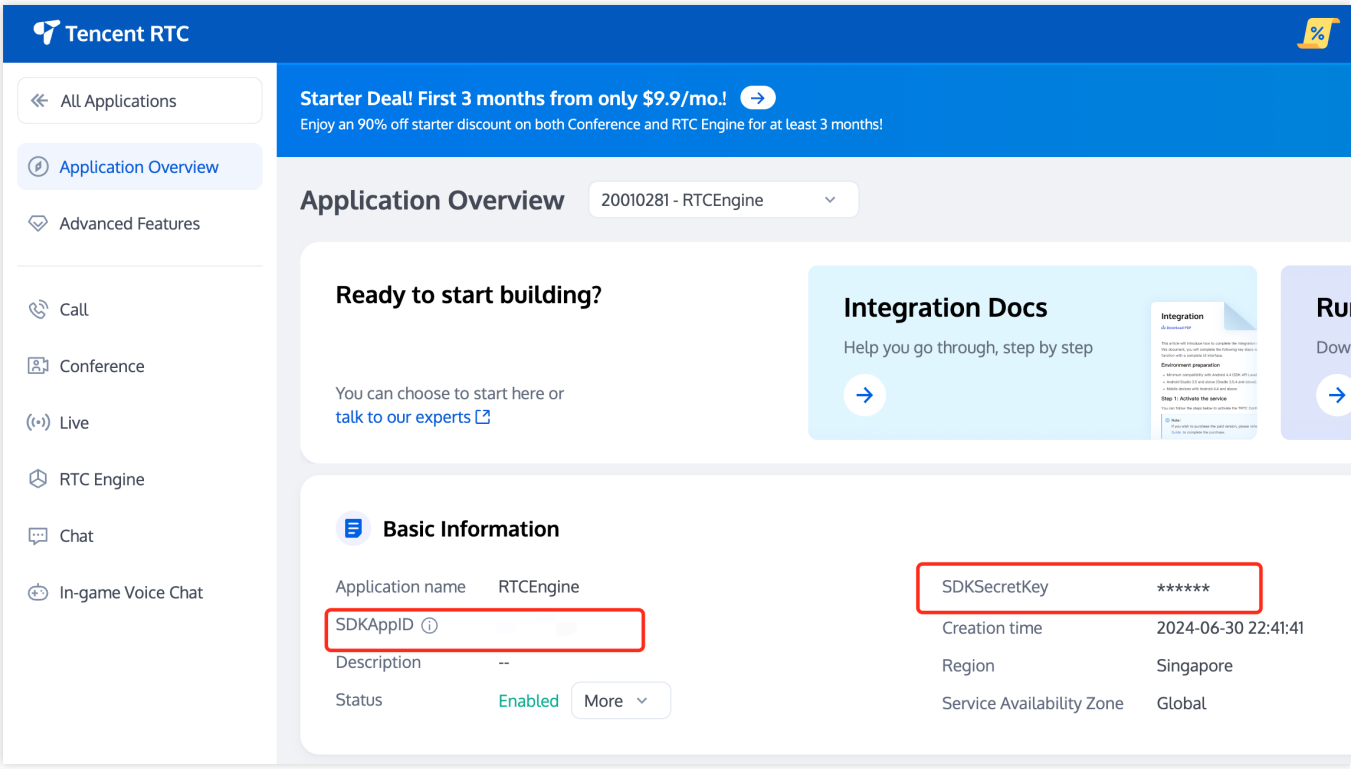
您可以在 github下载 Andorid 平台的示例代码，或者在终端运行以下命令：



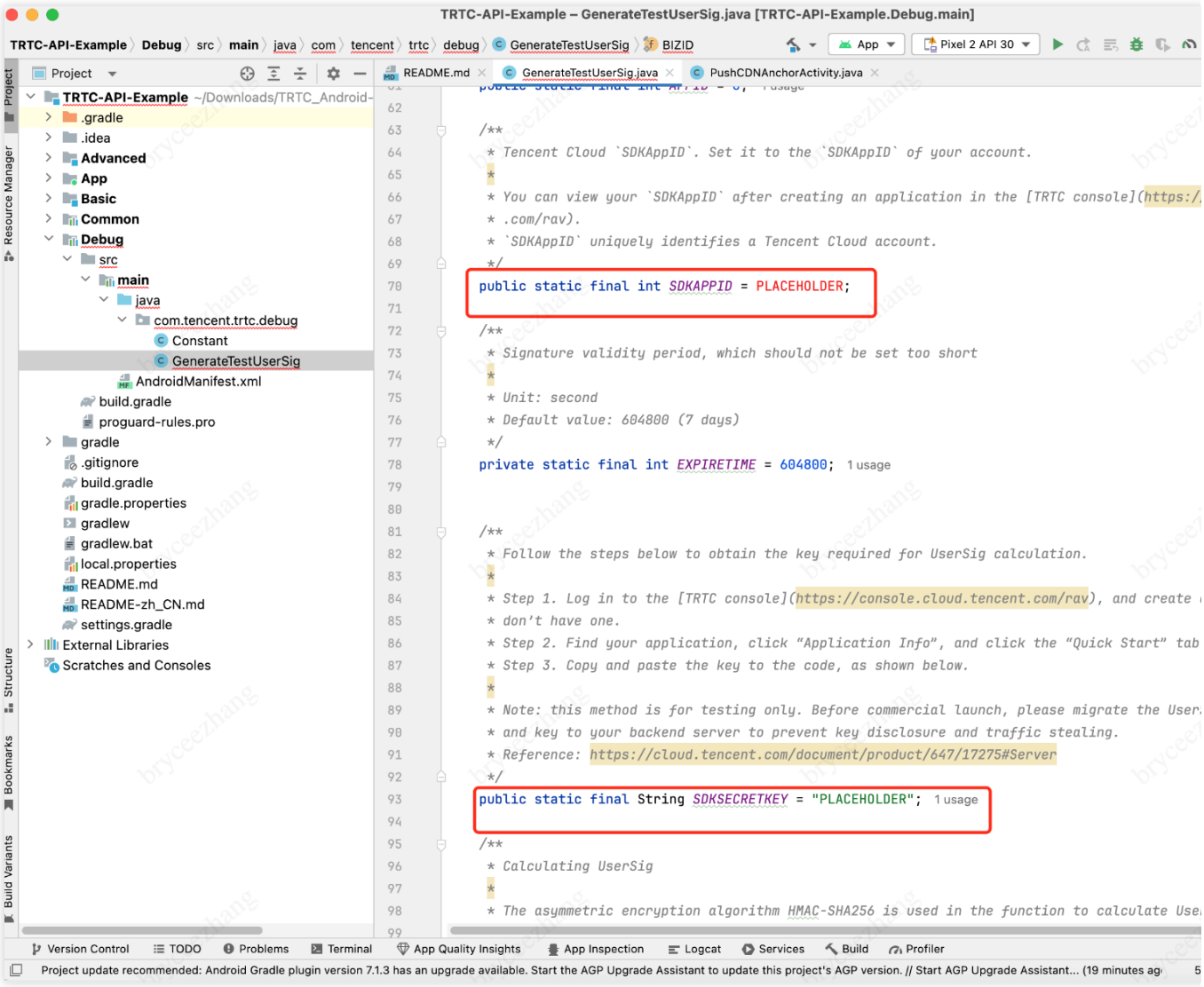
```
git clone https://github.com/Tencent-RTC/TRTC_Android.git
```

步骤2. 配置 Demo

1. 登录 TRTC Console，单击 **Create Application** 。如果您已经完成创建，可以跳过该操作。
2. 在创建应用成功后，您可以在基本信息中获取到您的 **SDKAppID** 和 **SDKSecretKey** 。



3. 您需要将 `TRTC-API-Example/Debug/src/main/java/com.tencent.trtc.debug` 目录下 `GenerateTestUserSig` 中的 **SDKAPPID** 和**SDKSECRETKEY** 替换为您在**步骤2**中获取到的值。



注意：

本文使用的生成 UserSig 的方案是在本地配置 SDKSecretKey，该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。正确的 UserSig 签发方式是在您的服务端集成 服务端生成 UserSig，当用户进房时：

发起 http 请求到您的服务端

服务端生成一个 UserSig

将其返回给用户供进房使用

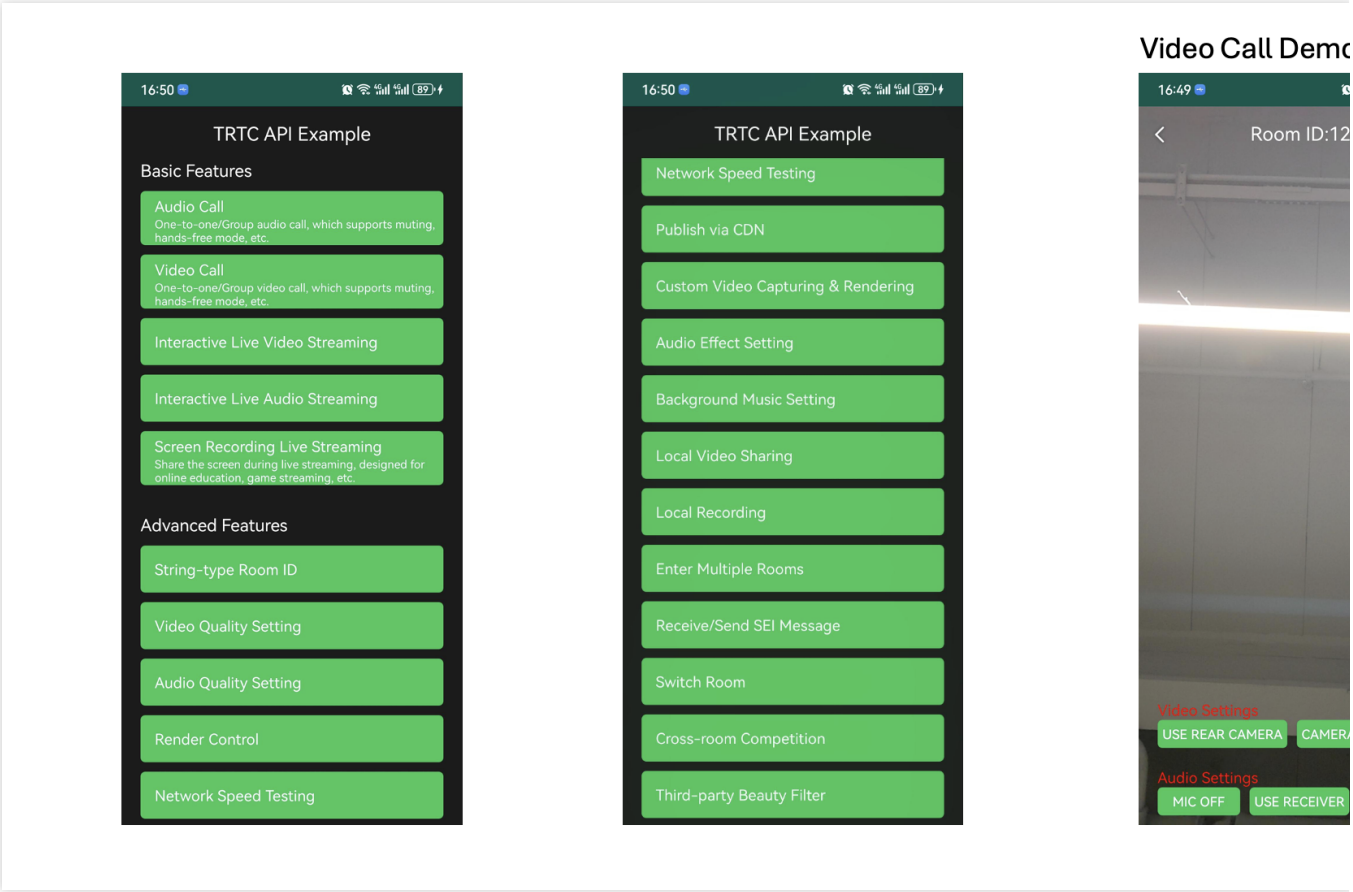
若要将 Demo 部署到公网体验，需要通过 HTTPS 协议，即 `https://域名/xxx` 访问，原因可参见文档 页面访问协议限制说明。

步骤3. 跑通 Demo

配置完成后，使用 Android Studio（3.5及以上的版本）打开源码工程 `TRTC-API-Example` ，单击**运行**即可体验。

步骤4. 体验 Demo

您可以选择自己感兴趣的功能进行体验。



常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com 。

Windows C++

最近更新时间：2024-07-05 19:30:25

本文介绍了如何快速运行腾讯云 TRTC Windows SDK Demo。

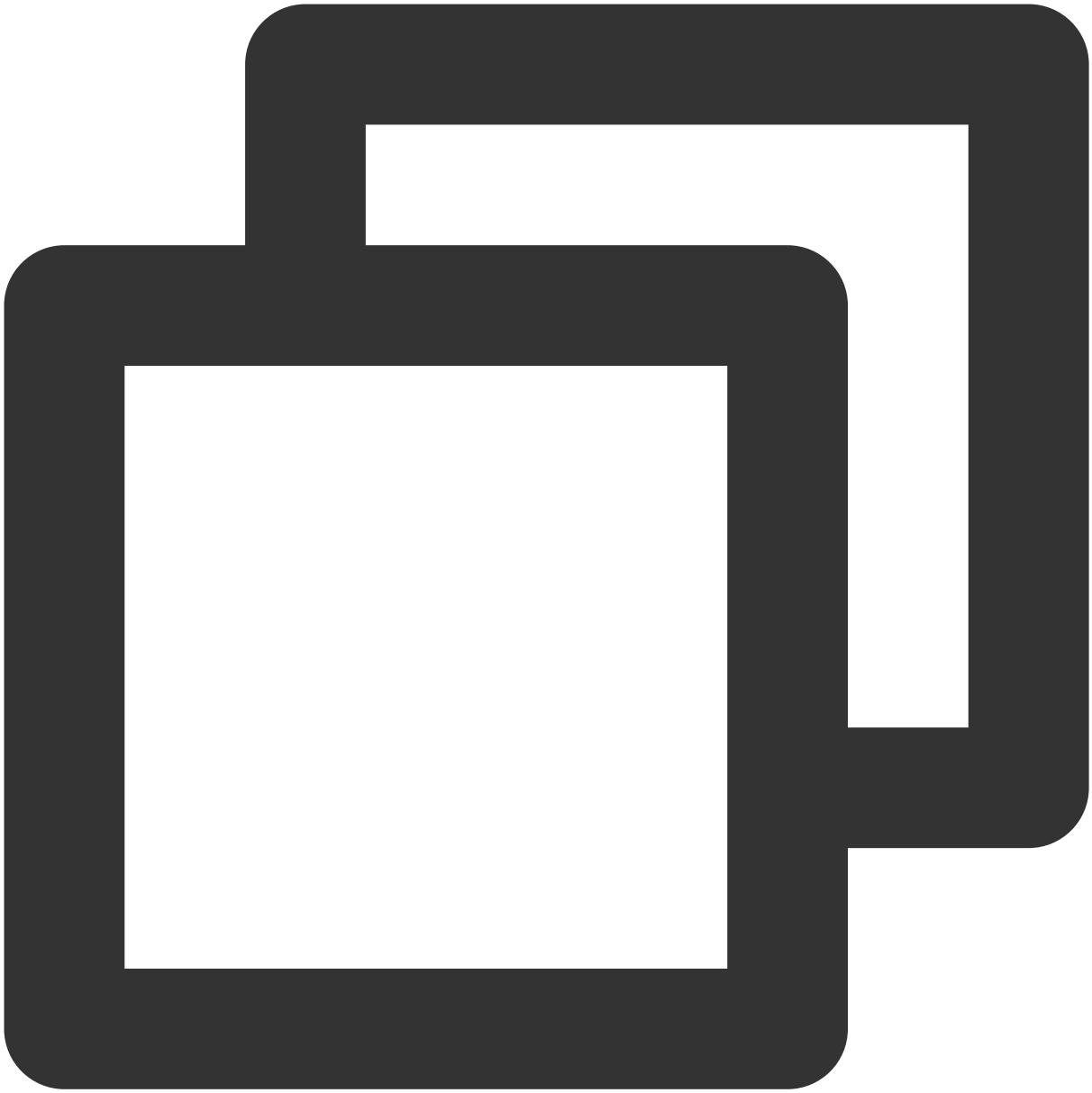
环境准备

Microsoft Visual Studio 2017 及以上版本，推荐使用 Microsoft Visual Studio 2019。
下载并安装 QT 开发环境（QT 5.14.x版本）。
下载并安装 .vsix 插件文件，根据官网文件命名找对应插件版本安装即可。
打开 VS 并在工具栏中找到 **Extension > QT VS Tools > Qt Options > Qt Versions**, 添加下载好的 Qt 编译器 msvc .
将 `SDK/CPlusPlus/Win64/lib` 下的所有的 `.dll` 文件拷贝到工程目录下的 `debug/release` 文件夹下。
注意：
`debug/release` 文件夹均是在 VS 上的环境配置完后自动生成。如果是32位程序，则需要拷贝 `SDK/CPlusPlus/Win64/lib` 下的所有 `.dll` 到 `debug` / `release` 文件夹下。

操作步骤

步骤1. 下载 Demo

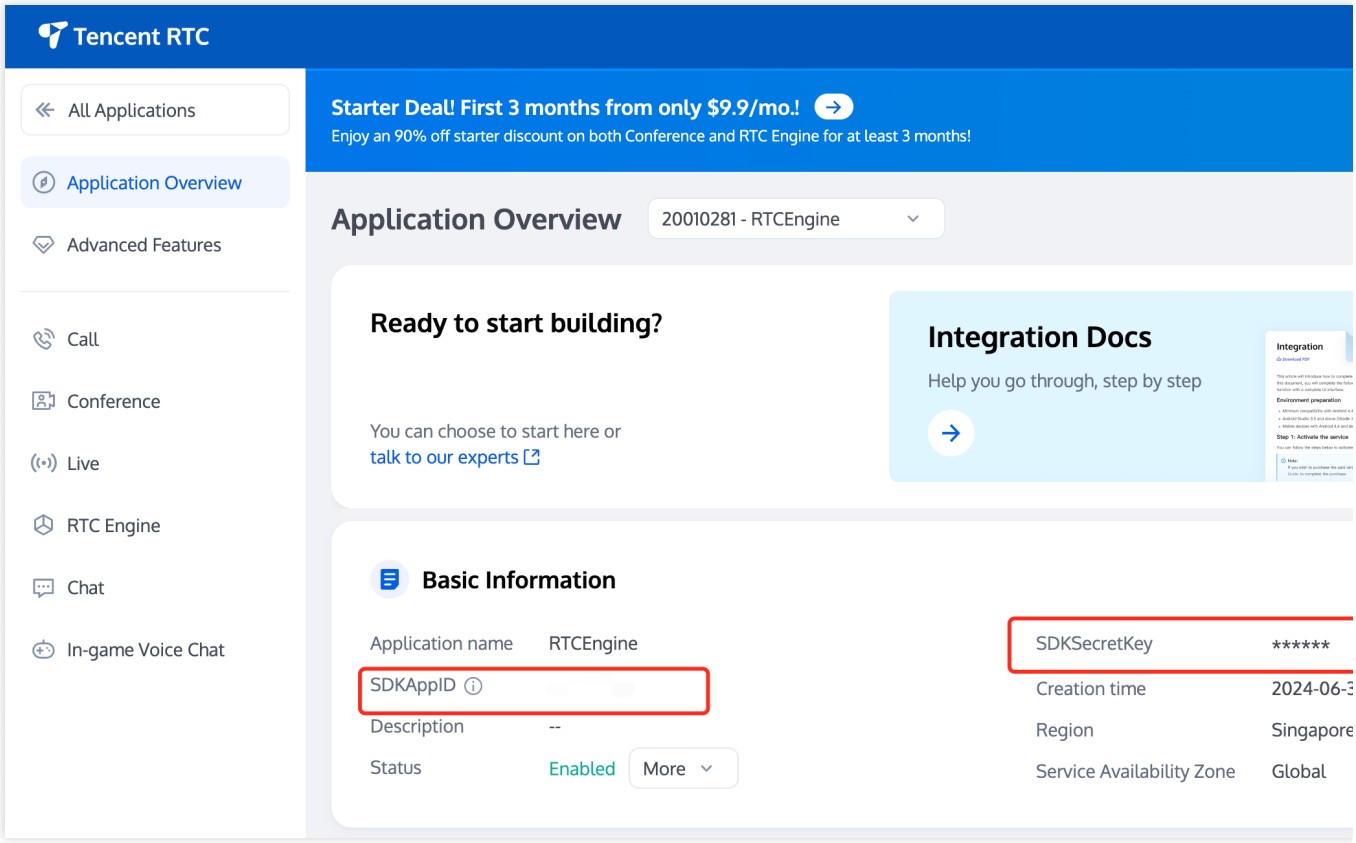
您可以在 github 下载 TRTC_Windows-C++ 平台的示例代码，或者在终端运行以下命令：



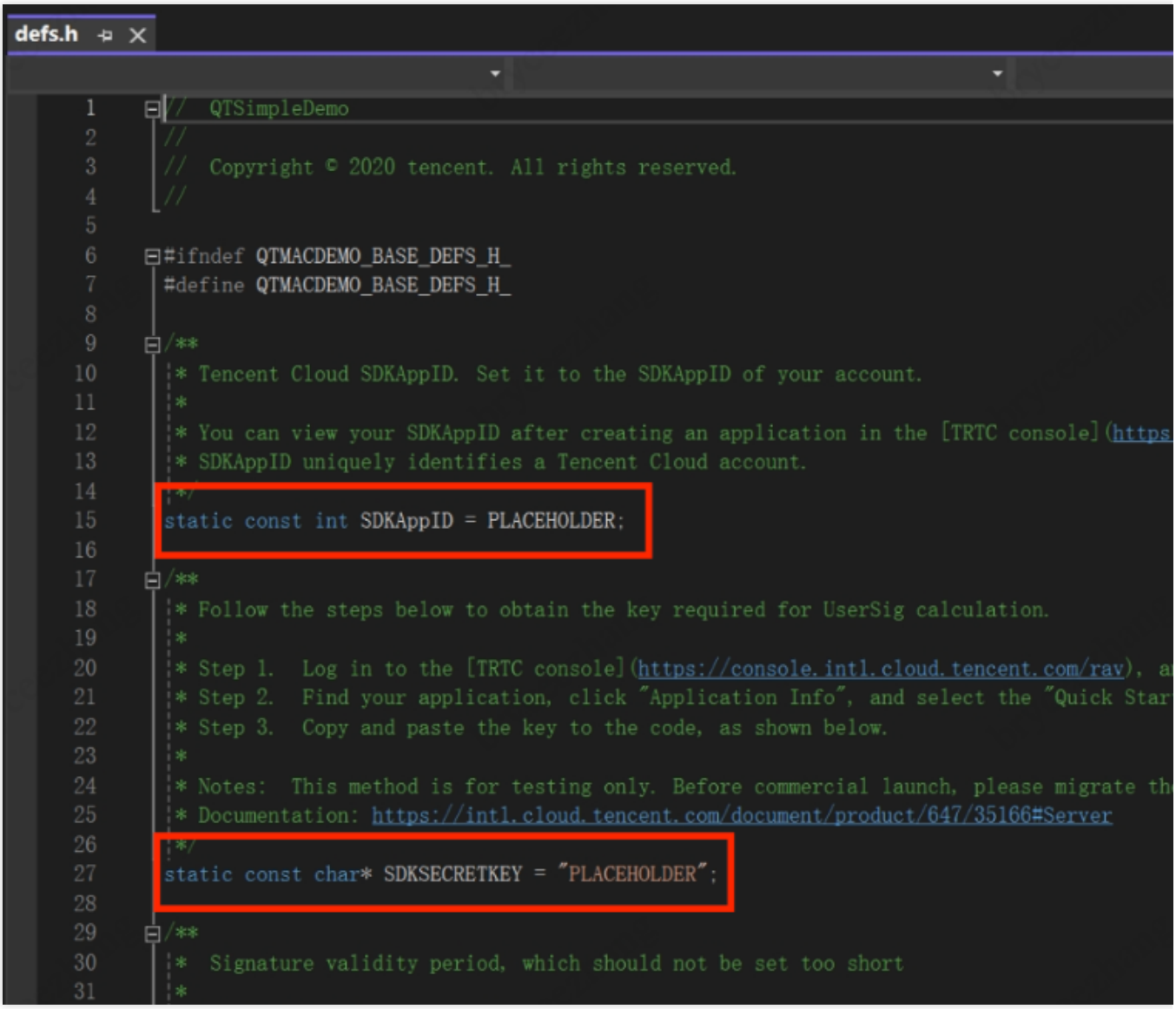
```
git clone https://github.com/Tencent-RTC/TRTC_Windows.git
```

步骤2. 配置 Demo

1. 登录 TRTC Console，单击 **Create Application** 。如果您已经完成创建，可以跳过该操作。
2. 在创建应用成功后，您可以在基本信息中获取到您的 **SDKAppID** 和 **SDKSecretKey** 。



3. 将 `TRTC-API-Example-C++/TRTC-API-Example-Qt/src/Util/` 目录下 `defs.h` 中的 **SDKAPPID** 和 **SDKSECRETKEY** 替换为**步骤2**中获取的值。



注意：

本文使用的生成 UserSig 的方案是在本地配置 SDKSecretKey，该方法中 SDKSecretKey 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。正确的 UserSig 签发方式是在您的服务端集成 服务端生成 UserSig，当用户进房时：

发起 http 请求到您的服务端

服务端生成一个 UserSig

将其返回给用户供进房使用

若要将 Demo 部署到公网体验，需要通过 HTTPS 协议，即 `https://域名/xxx` 访问，原因可参见文档 页面访问协议限制说明。

步骤3. 跑通 Demo

使用 Microsoft Visual Studio（推荐使Microsoft Visual Studio 2019）打开TRTC-API-Example-Qt目录下的源码工程 `QTDemo.sln` ，并配置 Qt 环境，（推荐使用 QT 5.14版本）。单击**运行**即可体验。

常见问题

如果您的接入和使用中遇到问题，请参见 常见问题。

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com 。

Web

最近更新时间：2024-05-29 15:21:54

本文主要介绍如何快速运行腾讯云 TRTC Web SDK Demo。

前提条件

您已 注册腾讯云 账号。

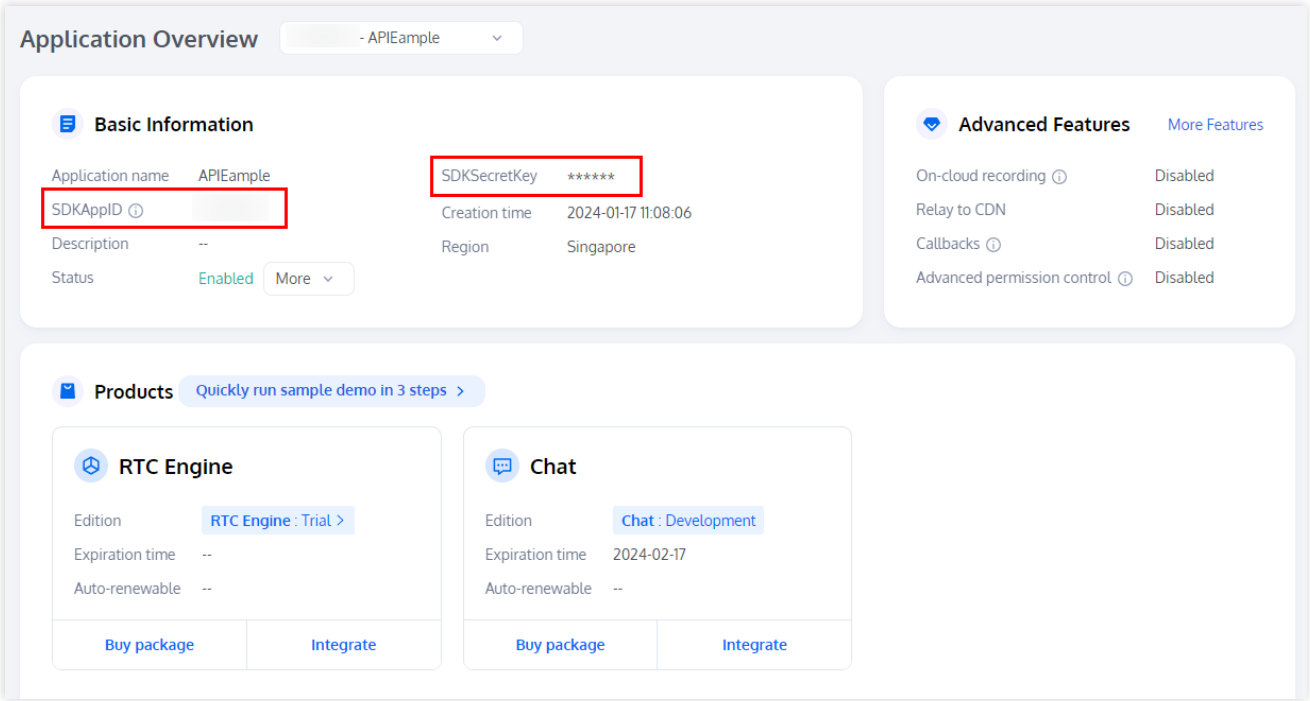
操作步骤

1. 创建应用

登录到 实时音视频 TRTC 控制台，单击 **Create Application**，创建一个 **RTC Engine** 应用，如果已经创建可以忽略。

2. 获取 SDKAppID 和 SDKSecretKey

在应用创建成功后，您可以在基础信息中获取到您的 `SDKAppID` 和 `SDKSecretKey`，体验 Demo 需要用到这两个信息。



3. 运行 Demo

1. 运行在线 Demo

TRTC Web 目前提供以下几种基础 Demo，您可以选择您熟悉的项目框架进行运行体验：

1.1 quick-demo-js 是使用 Javascript 开发的 Demo。 源码：github。

1.2 quick-demo-vue2-js 是使用 Vue2 + Javascript 开发的 Demo。源码：github。

1.3 quick-demo-vue3-ts 是使用 Vue3 + Typescript 开发的 Demo。源码：github。

2. 运行本地 Demo

2.1 从 Github or Zip 下载源码。

2.2 按下述指引运行本地 Demo。

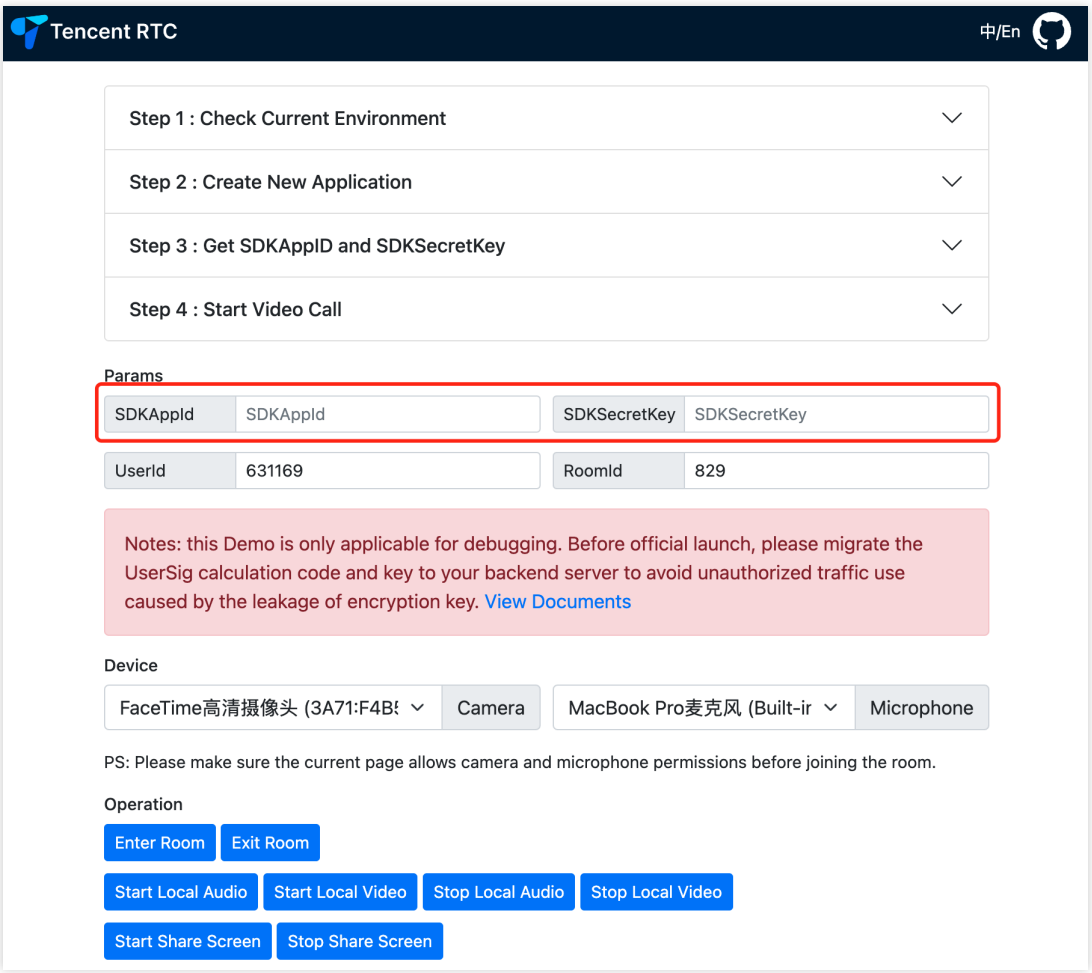
quick-demo-js

quick-demo-vue2-js

quick-demo-vue3-ts

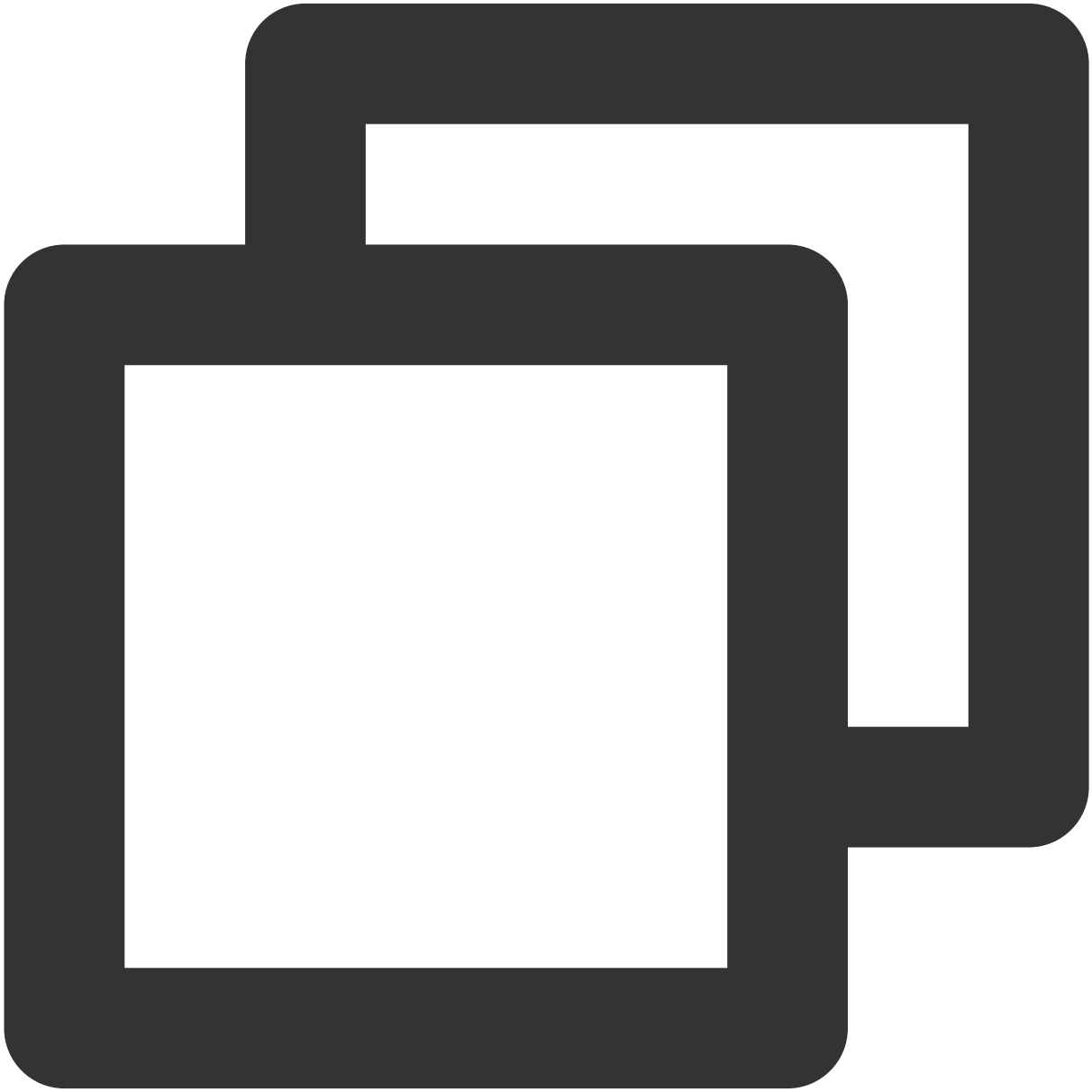
1. 在下载的源码中找到并使用浏览器打开 `TRTC_Web/quick-demo-js/index.html` 文件。

2. 在浏览器打开的页面中填写**步骤2**获取的 **SDKAppId** 和 **SDKSecretKey**。



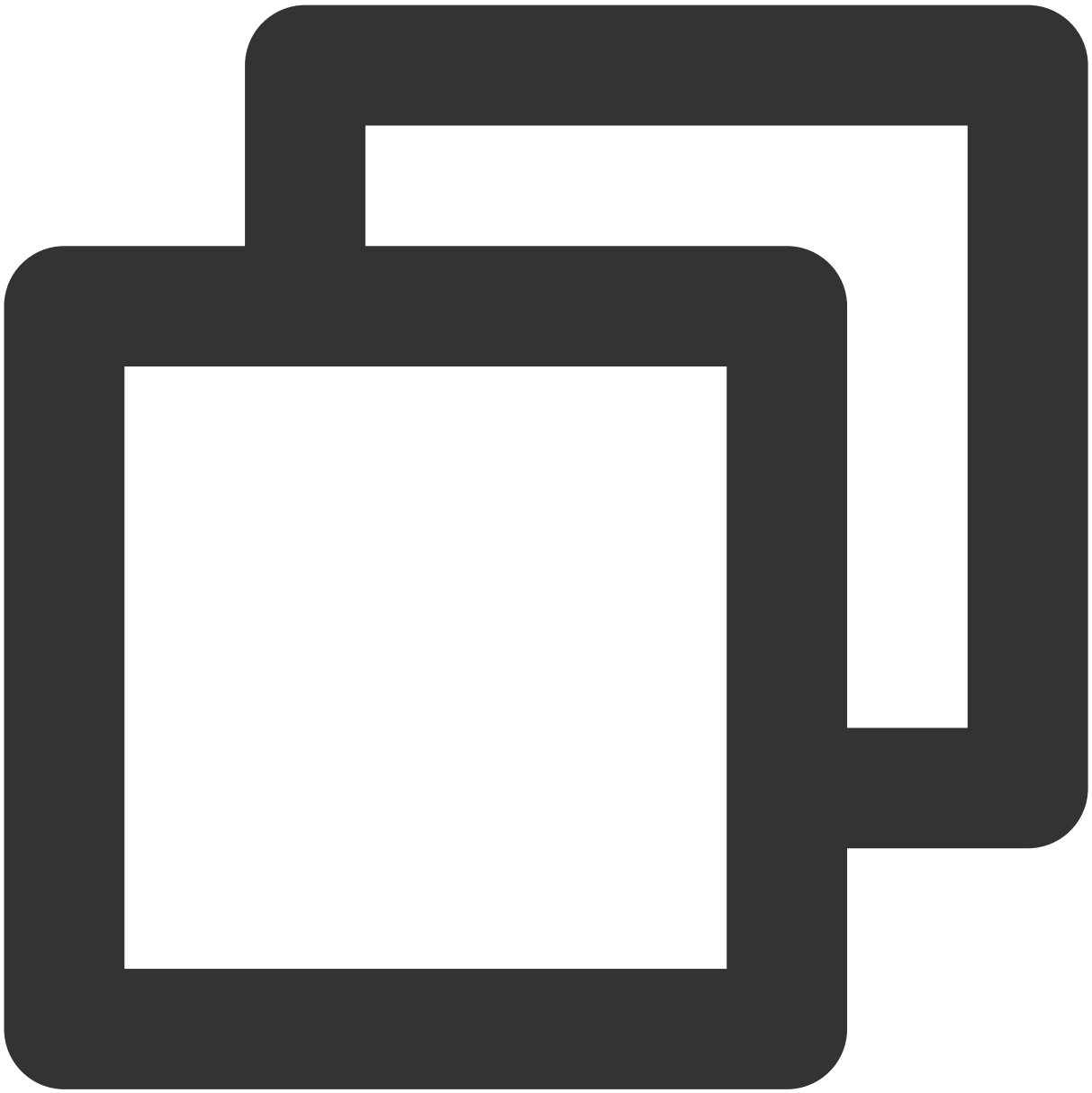
3. 功能体验

单击 **Enter Room** 进入房间。
单击 **Start Local Audio/Video**，可采集麦克风或摄像头。
单击 **Stop Local Audio/Video**，可终止采集麦克风或摄像头。
单击 **Start Share Screen** 开始屏幕分享。
单击 **Stop Share Screen** 取消屏幕分享。
4. 加入房间后您可以通过分享邀请链接与被邀请人一起体验 TRTC Web 语音及视频互通功能。
1. 在下载的源码中找到并进入到 `TRTC_Web/quick-demo-vue2-js/` 目录下。



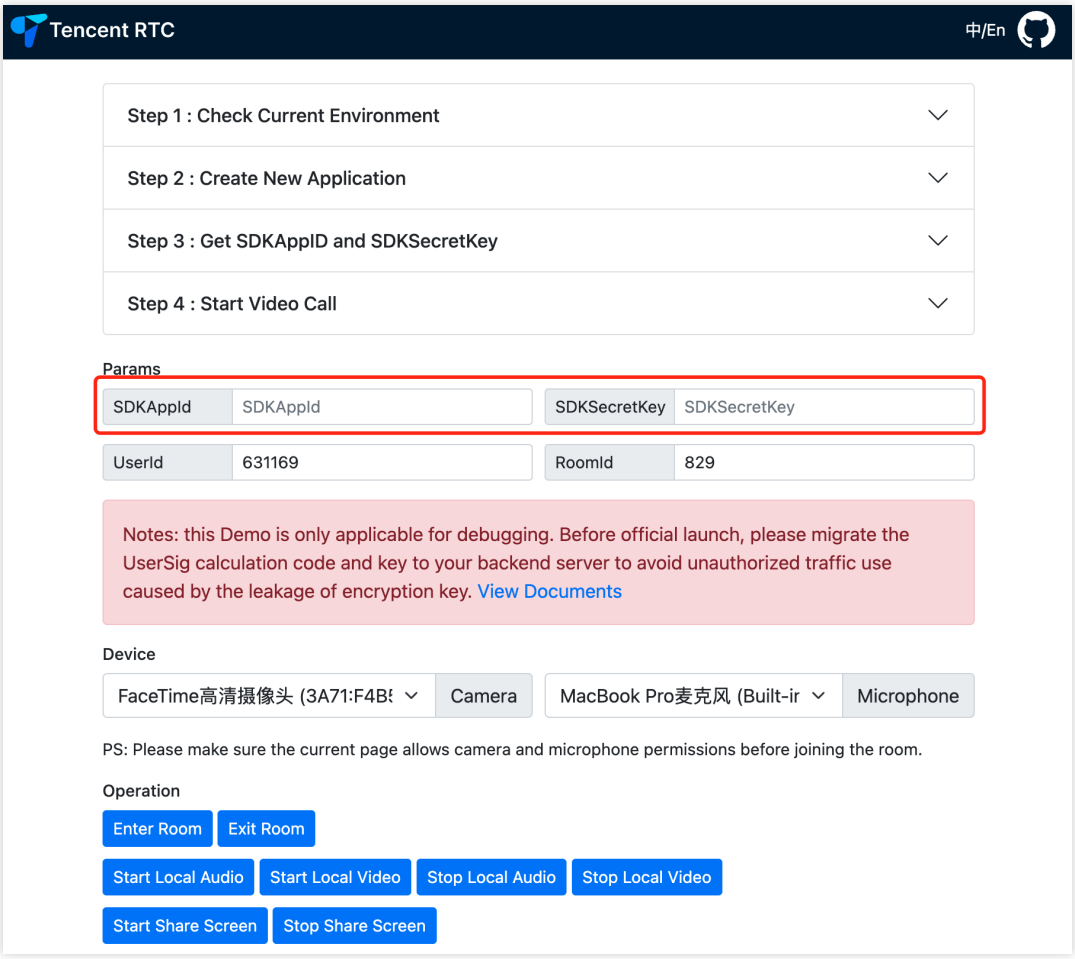
```
cd quick-demo-vue2-js
```

2. 本地运行 Demo，在终端执行以下命令，将会自动安装依赖并运行 demo。
运行该命令需要先安装 Nodejs。



```
npm start
```

3. 默认浏览器会自动打开 `http://localhost:8080/` 地址。
4. 在浏览器打开的页面中填写**步骤2**获取的 **SDKAppId** 和 **SDKSecretKey**。



5. 功能体验

点击**进入房间**按钮进入房间。

点击**采集麦克风/摄像头**按钮，可采集麦克风或摄像头。

点击**终止采集麦克风/摄像头**按钮，可终止采集麦克风或摄像头。

点击**开始共享屏幕**按钮开始屏幕分享

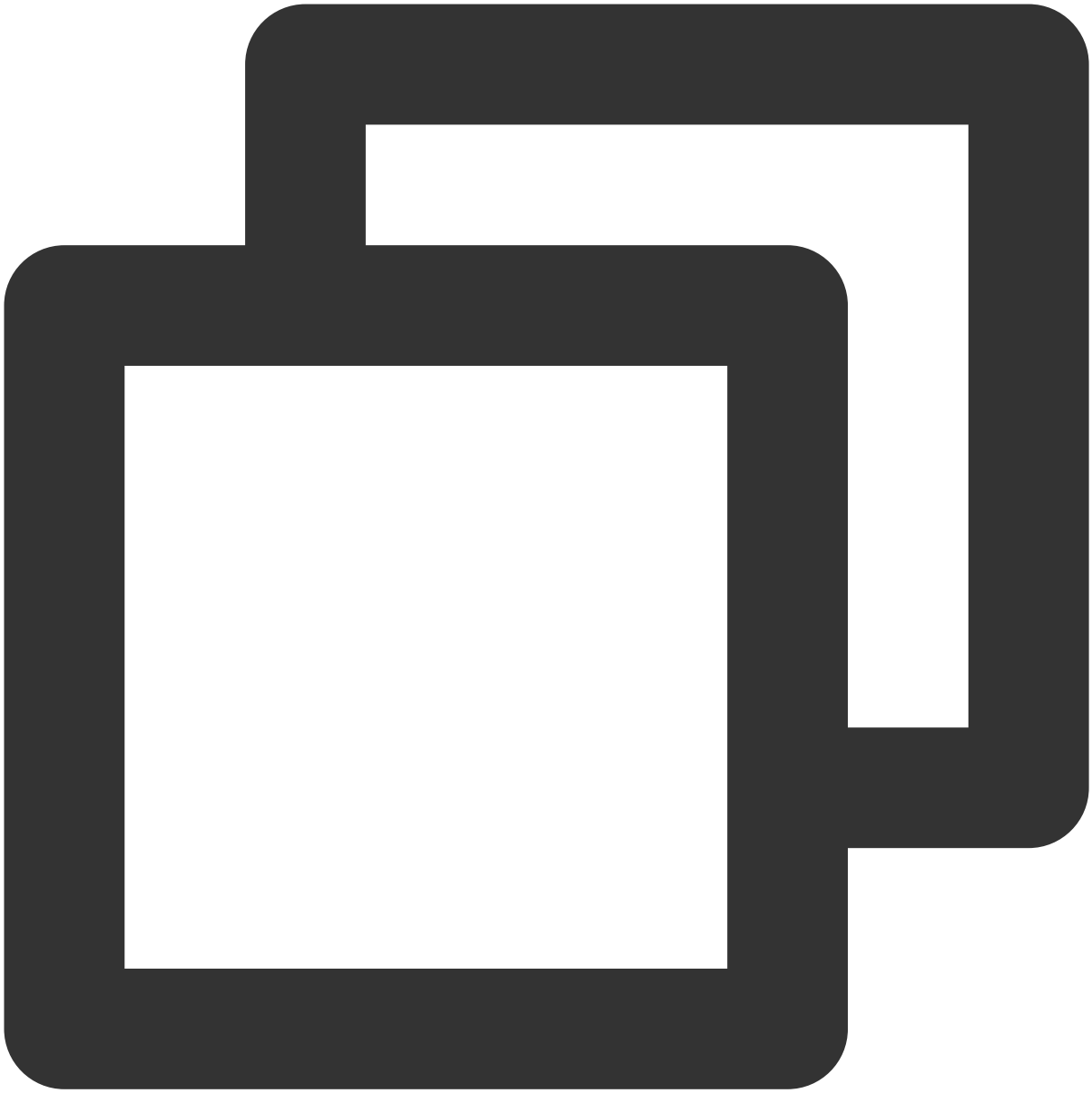
点击**停止共享屏幕**按钮取消屏幕分享

6. 加入房间后您可以通过分享邀请链接与被邀请人一起体验 TRTC Web 语音及视频互通功能。

1. 在下载的源码中找到并进入到 `TRTC_Web/quick-demo-vue3-ts/` 目录下。

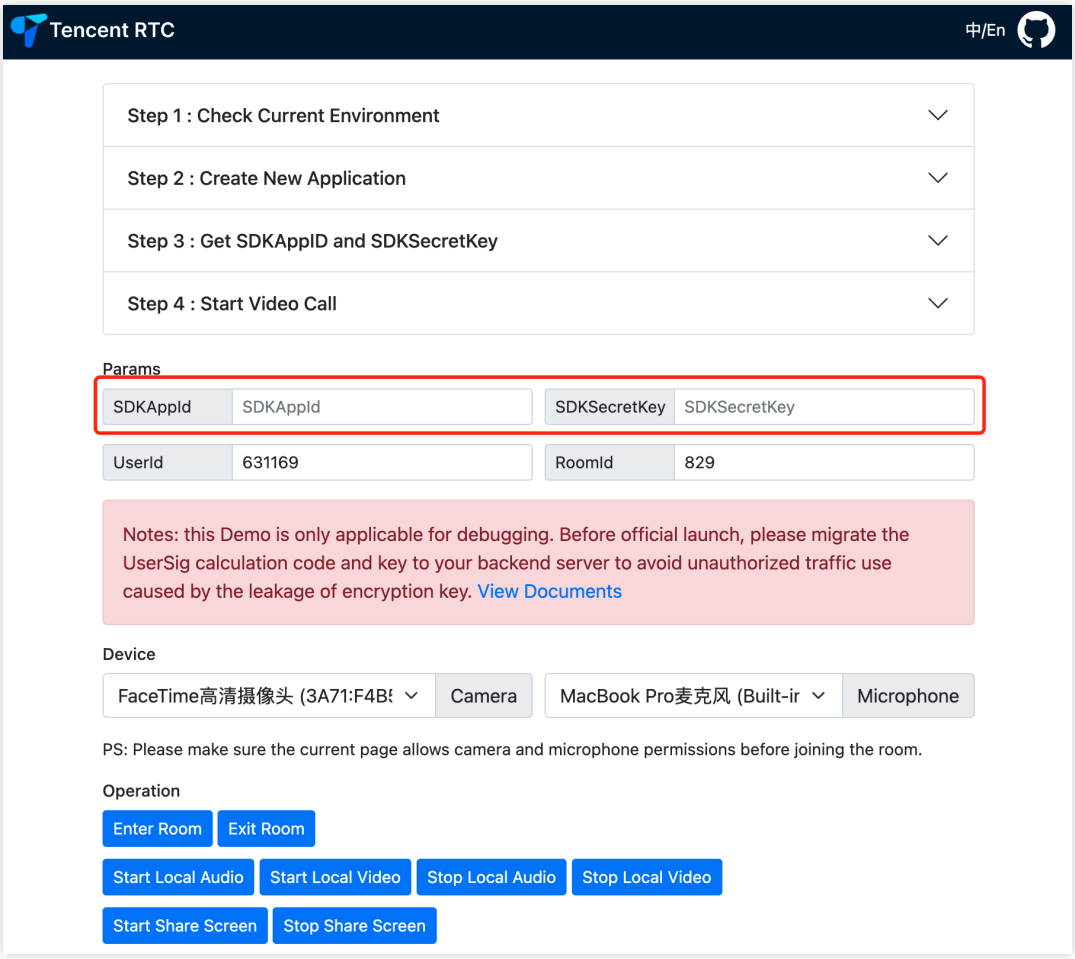
2. 本地运行 Demo，在终端执行以下命令，将会自动安装依赖并运行 demo。

运行该命令需要先安装 Nodejs。



```
npm start
```

3. 默认浏览器会自动打开 `http://localhost:3000/` 地址。
4. 在浏览器打开的页面中填写**步骤2**获取的 **SDKAppId** 和 **SDKSecretKey**。



5. 功能体验

输入 userId 和 roomId。

点击**Enter Room**按钮进入房间。

点击**Start Local Audio/Video**按钮，可采集麦克风或摄像头。

点击**Stop Local Audio/Video**按钮，可终止采集麦克风或摄像头。

点击**Start Share Screen**按钮开始屏幕分享。

点击**Stop Share Screen**按钮取消屏幕分享。

6. 加入房间后您可以通过分享邀请链接与被邀请人一起体验 TRTC Web 语音及视频互通功能。

注意：

本文使用的生成 UserSig 的方案是在本地配置 SECRETKEY，该方法中 SECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。正确的 UserSig 签发方式是在您的服务端集成 服务端生成 UserSig，当用户进房时：

发起 http 请求到您的服务端

服务端生成一个 UserSig

将其返回给用户供进房使用

若要将 Demo 部署到公网体验，需要通过 HTTPS 协议，即 `https://域名/xxx` 访问，原因可参见文档 页面访问协议限制说明。

常见问题

支持的平台

TRTC Web SDK 基于 WebRTC 实现，目前支持桌面端和移动端的主流浏览器（**Chrome, Edge, Safari, Firefox, Opera**），详细支持度表格请参见 支持的平台。

URL 域名协议限制

由于浏览器安全策略的限制，使用 WebRTC 能力对页面的访问协议有严格的要求，请参照以下表格进行开发和部署应用。

应用场景	协议	接收（播放）	发送（上麦）	屏幕分享	备注
生产环境	HTTPS 协议	支持	支持	支持	**推荐**
生产环境	HTTP 协议	支持	不支持	不支持	-
本地开发环境	http://localhost	支持	支持	支持	**推荐**
本地开发环境	http://127.0.0.1	支持	支持	支持	-
本地开发环境	http://[本机IP]	支持	不支持	不支持	-
本地开发环境	file:///	支持	支持	支持	-

防火墙限制

在使用 TRTC Web SDK 时，用户可能因防火墙限制导致无法正常进行音视频通话，请参见 应对防火墙限制相关 将相应端口及域名添加至防火墙白名单中。

其他

其他常见问题，请参见：Web 端常见问题。

版权所有：腾讯云计算（北京）有限责任公司

Electron

最近更新时间：2024-05-13 10:43:56

本文主要介绍如何快速运行腾讯云 TRTC-API-Example（Electron）。

前提条件

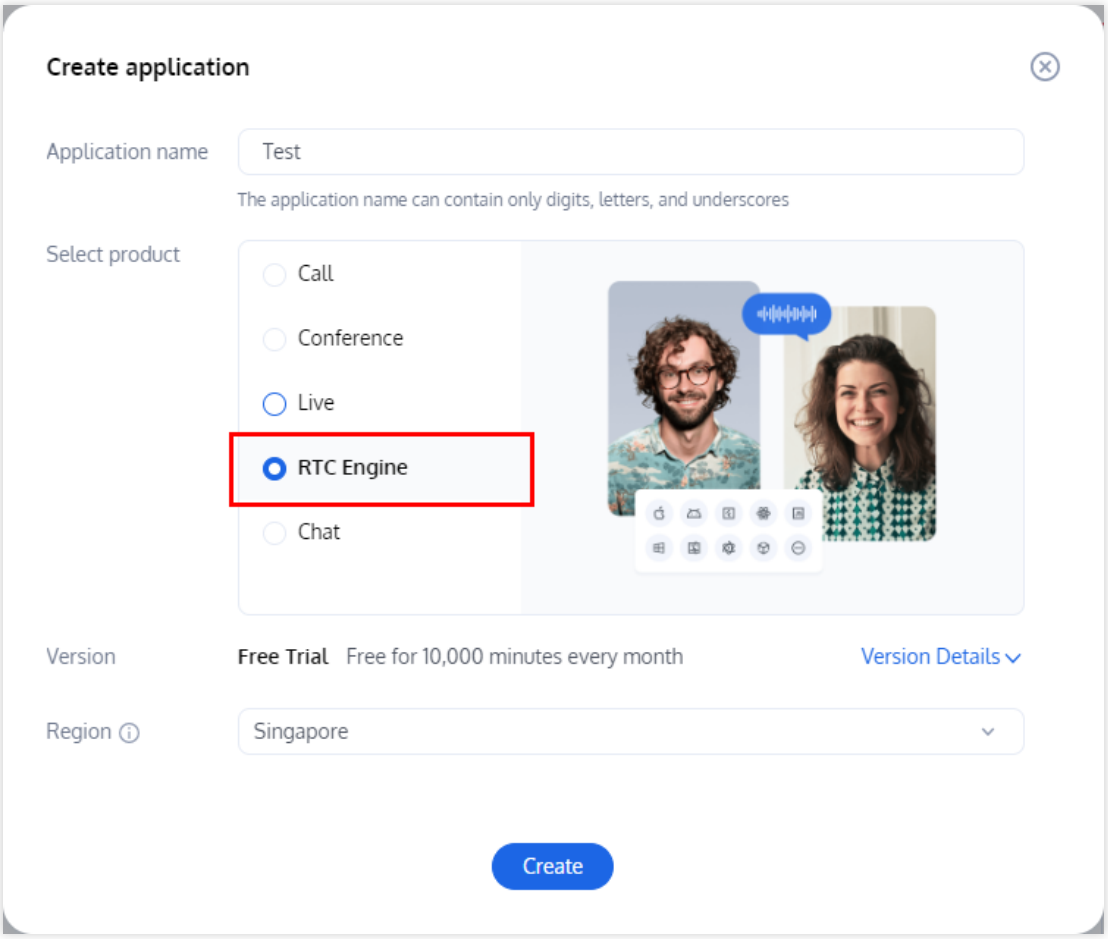
您已 注册腾讯云 账号。

操作步骤

步骤1：创建新的应用

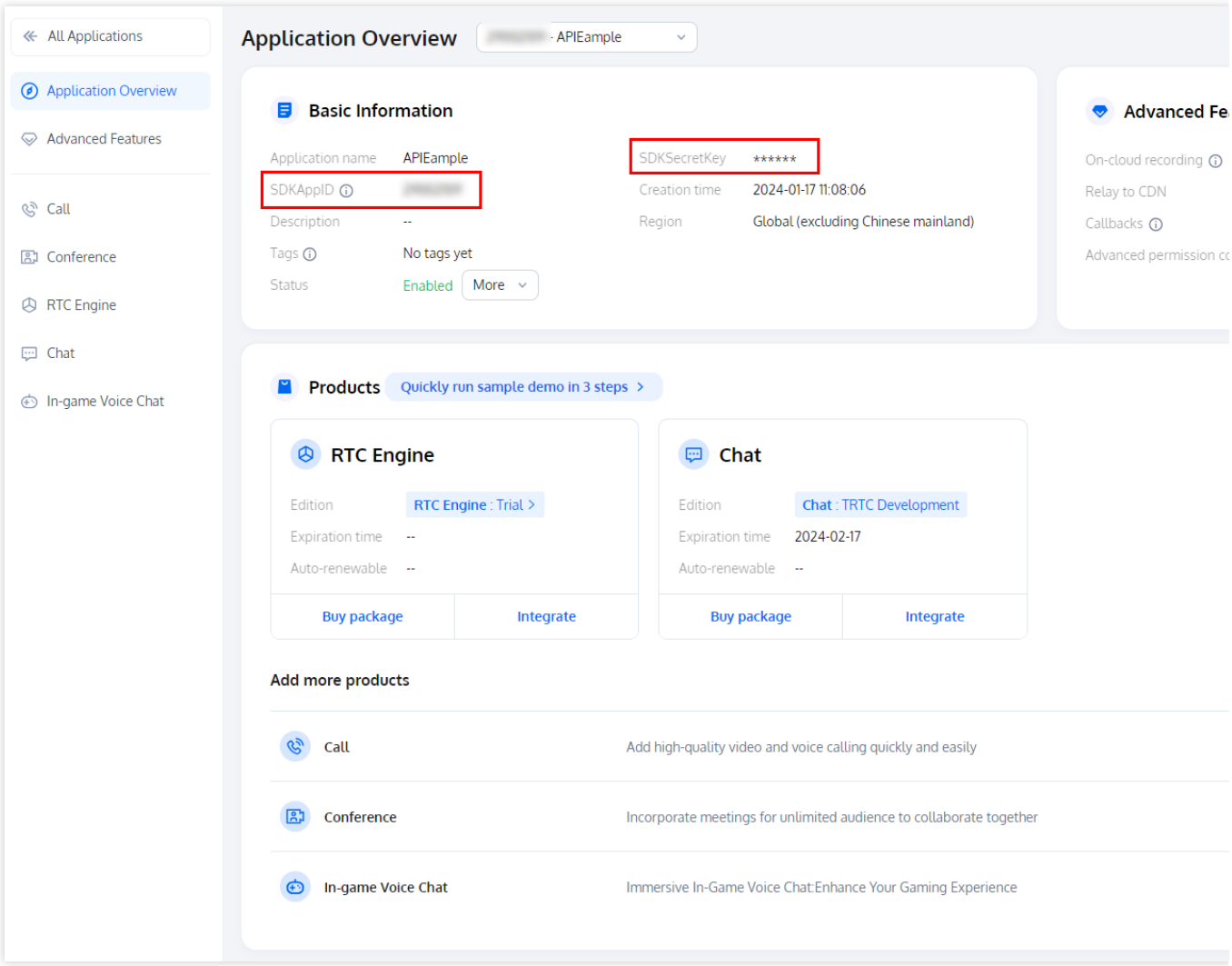
1. 登录 实时音视频控制台总览页，单击**创建应用**。
2. 在弹窗页面中选择 RTC Engine，输入应用名称，然后单击**创建**。

版权所有：腾讯云计算（北京）有限责任公司



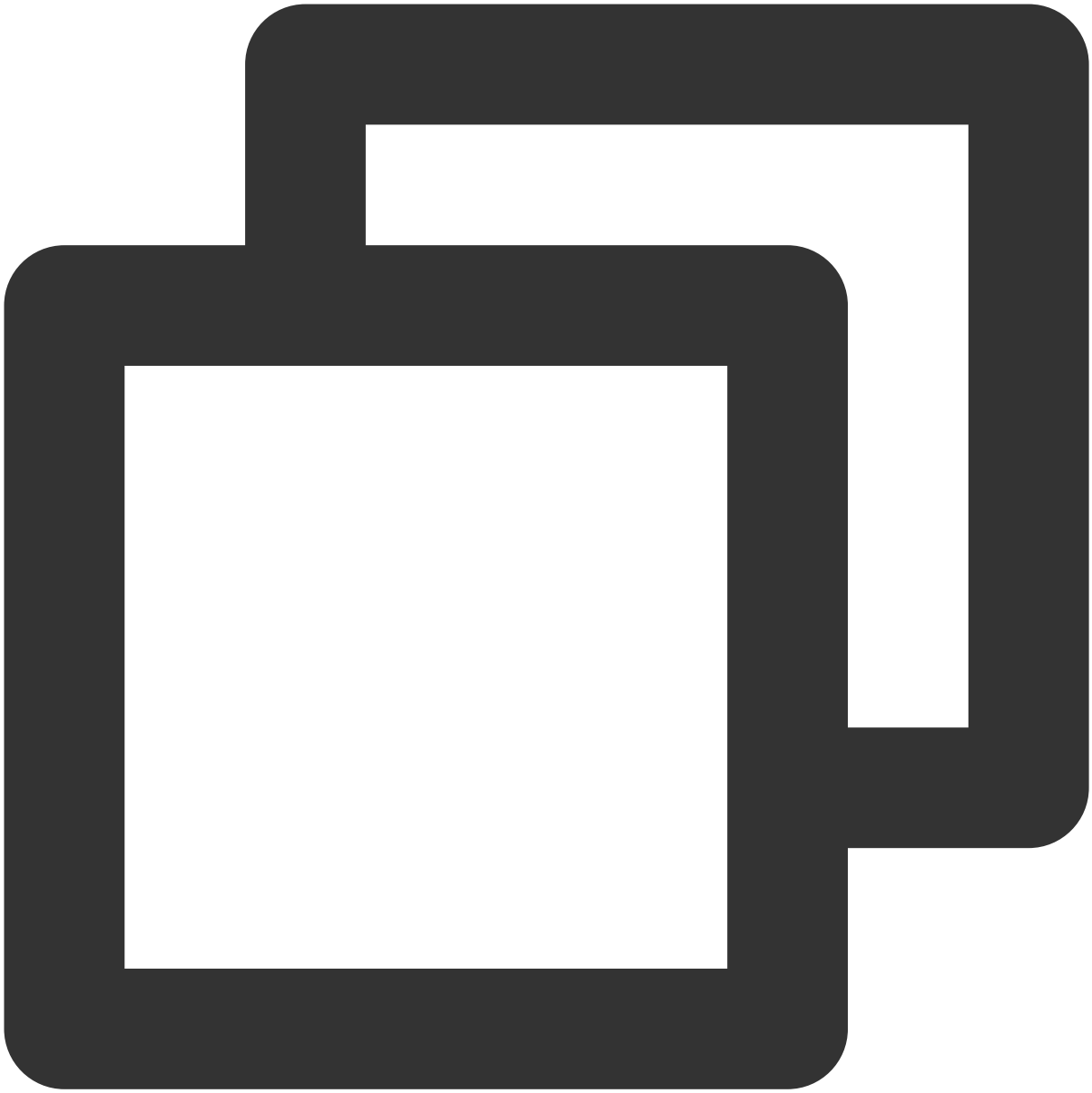
步骤2：获取 SDKAppID 和 SecretKey

在应用创建成功后，您可以在基本信息中获取到您的 `SDKAppId` 和 `SDKSecretKey`，体验 Demo 需要用到这两个信息。



步骤3：下载SDK和示例代码

1. 您可以根据自己的业务平台，前往 Github 下载对应平台的示例代码。



```
git clone https://github.com/LiteAVSDK/TRTC_Electron.git
```

2. 导入SDK的步骤可以参考：Electron SDK导入

步骤4：配置工程

1. 找到并打开 TRTC-API-Example/assets/debug/gen-test-user-sig.js 文件。
2. 设置其中的相关参数：

SDKAPPID：默认为 0，请设置为实际的 SDKAppID。

SDKSECRETKEY：默认为 ''，请设置为实际的密钥信息。

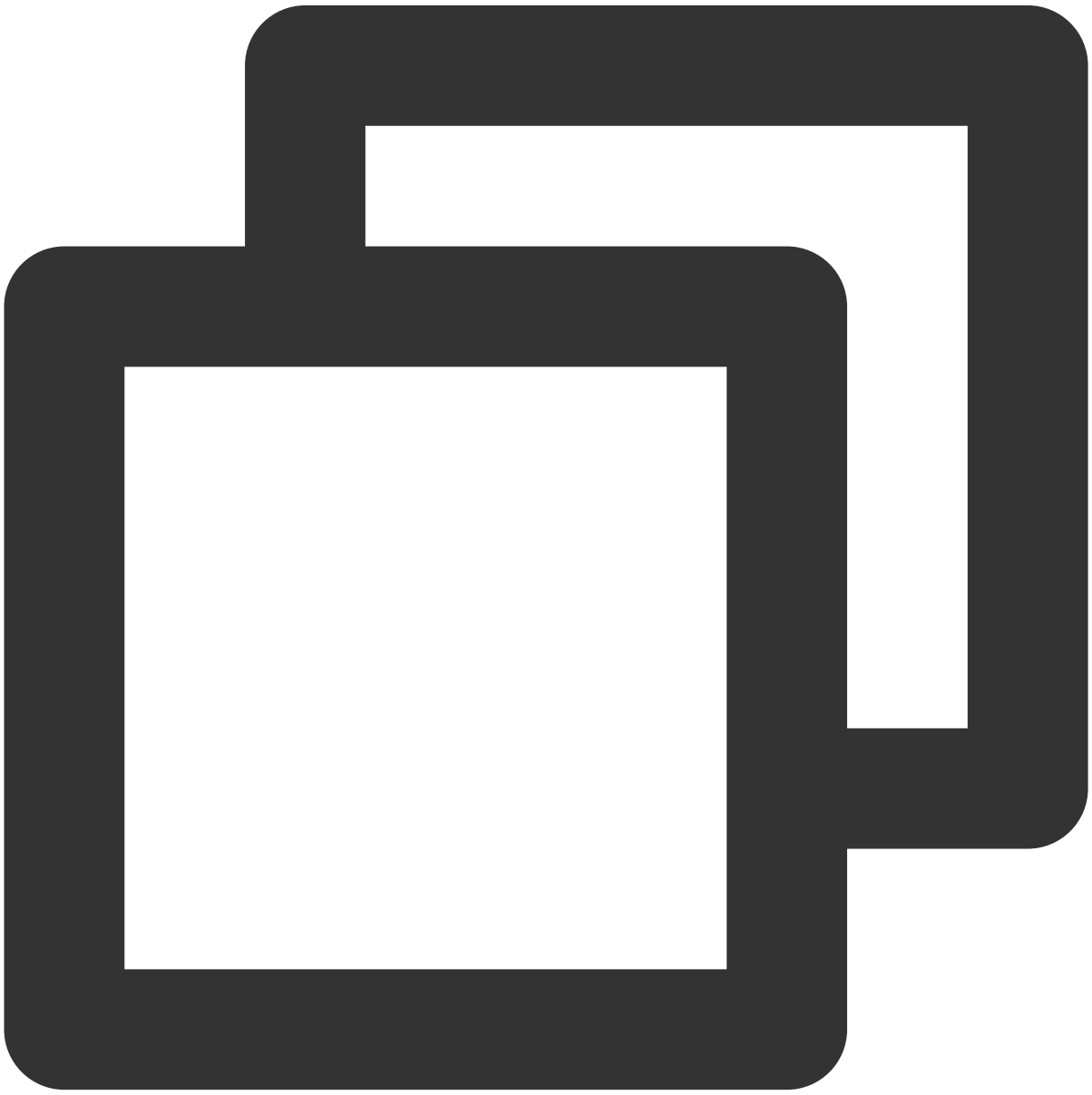
此时工程配置已经完成。

说明：

本文提到的生成 UserSig 的方案是在客户端代码中配置 SDKSECRETKEY，该方法中 SDKSECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TRTC-API-Example 和功能调试**。

正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤5：编译运行



```
cd TRTC-API-Example
npm install
cd src/app/render/main-page
npm install

cd ../../..
npm run start
```

常见问题

1. 防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以在对 UDP 有拦截的办公网络下无法使用。如遇到类似问题，请参见 应对公司防火墙限制 排查并解决。

Flutter

最近更新时间：2024-05-13 11:31:34

本文主要介绍如何快速运行腾讯云 TRTC Demo（Flutter）。

注意：

目前 Windows/MacOs 端暂不支持屏幕分享及设备选择功能。

环境要求

Flutter 2.0 及以上版本。

Android 端开发：

Android Studio 3.5及以上版本。

App 要求 Android 4.1及以上版本设备。

iOS & macOS 端开发：

Xcode 11.0及以上版本。

osx 系统版本要求 10.11 及以上版本

请确保您的项目已设置有效的开发者签名。

Windows 开发：

操作系统：Windows 7 SP1 或更高的版本（基于 x86-64 的 64 位操作系统）。

磁盘空间：除安装 IDE 和一些工具之外还应有至少 1.64 GB 的空间。

安装 Visual Studio 2019。

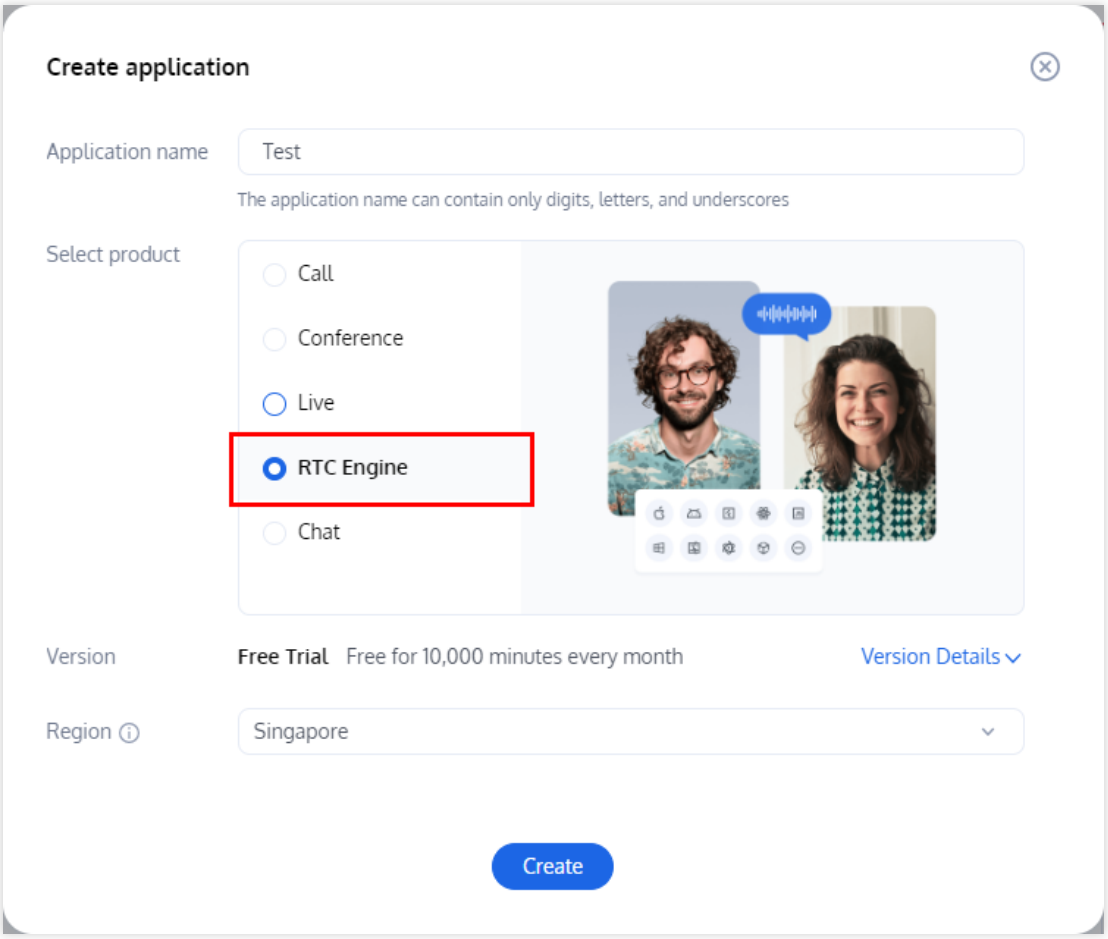
前提条件

您已 注册腾讯云 账号。

操作步骤

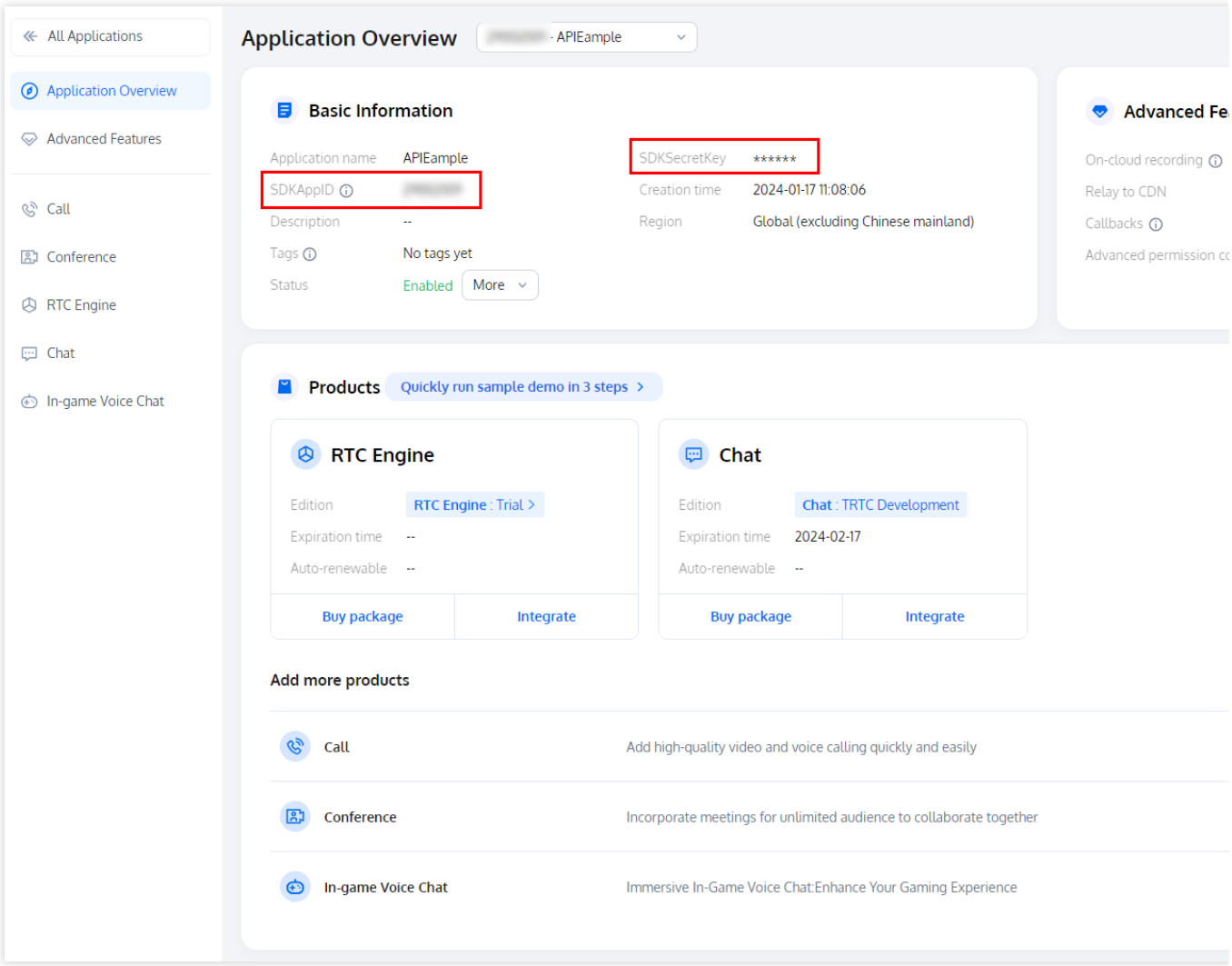
步骤1：创建新的应用

1. 登录 实时音视频控制台，点击 **Create Application**。
2. 在弹框页面选择 RTC Engine，然后单击 **Create**。



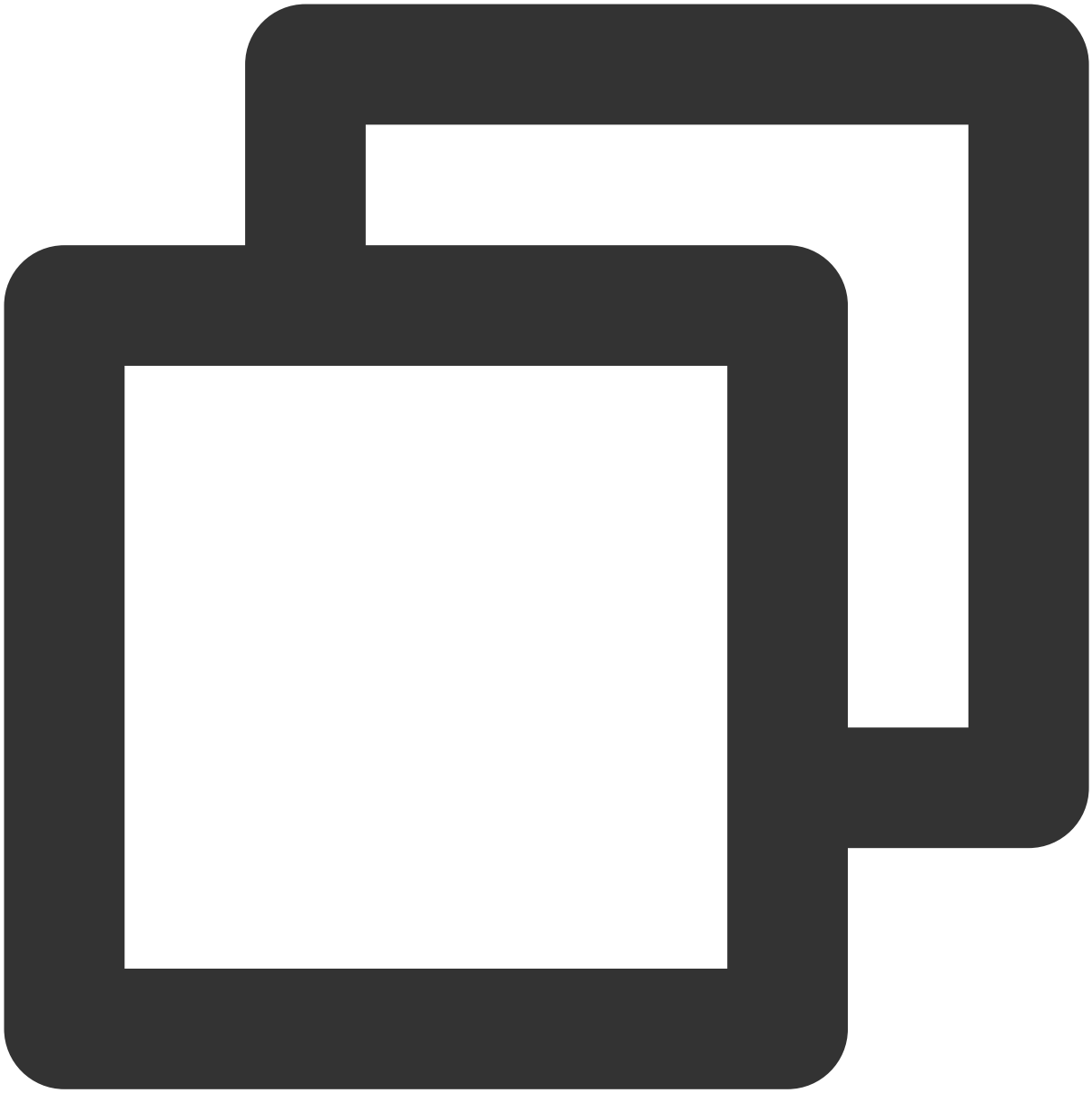
步骤2：获取 SDKAppID 和 SDKSecretKey

在应用创建成功后，您可以在基本信息中获取到您的 SDKAppId 和 SDKSecretKey，体验 Demo 需要用到这两个信息。



步骤3：下载 SDK 和示例代码

1. 您可以根据自己的业务平台，前往 Github 下载对应平台的示例代码。



```
git clone https://github.com/LiteAVSDK/TRTC_Flutter.git
```

2. 导入 SDK 的步骤可以参考：Flutter SDK 导入。

步骤 4：配置工程

1. 找到并打开 `TRTC-Simple-Demo/lib/debug/GenerateTestUserSig.dart` 文件，设置 `GenerateTestUserSig.dart` 文件中的相关参数：

SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。

SDKSECRETKEY：默认为 PLACEHOLDER，请设置为实际的 SDKSecret key。

2. 此时工程配置已经完成。

说明：

本文提到的生成 UserSig 的方案是在客户端代码中配置 SDKSECRETKEY，该方法中 SDKSECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TRTC-Simple-Demo 和功能调试**。

正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤 5：编译运行

1. 执行 `flutter pub get`

2. 编译、运行或调试项目工程。

Android端

iOS端

Windows端

macOS端

1. 执行 flutter run。

2. 使用 Android Studio（3.5及以上的版本）打开源码工程，单击运行即可。

1. 到 iOS 目录，执行 pod install。

2. 使用 XCode（11.0及以上的版本）打开源码目录下的 /ios 工程，编译并运行 Demo 工程即可。

1. 启用 Windows 支持：flutter config --enable-windows-desktop。

2. flutter run -d windows。

1. 启用 macOS 支持：flutter config --enable-macos-desktop。

2. 到 macos目录 pod install。

3. 执行 flutter run -d macos。

常见问题

如何查看 TRTC 日志？

TRTC 的日志默认压缩加密，后缀为 `.xlog` 。地址如下：

iOS 端：sandbox 的 `Documents/log` 。

Android 端：

6.7及之前的版本：`/sdcard/log/tencent/liteav` 。

6.8之后的版本：`/sdcard/Android/data/包名/files/log/tencent/liteav/` 。

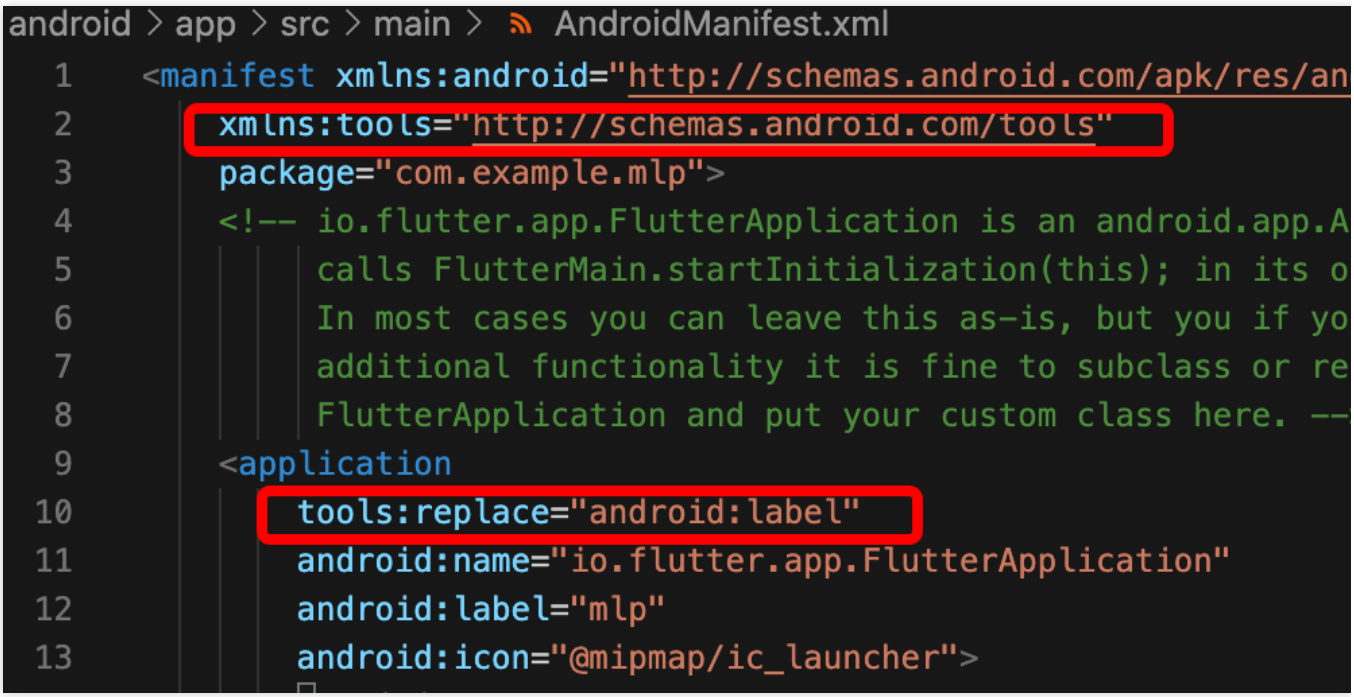
iOS 无法显示视频（Android 没问题）？

请确认在您的 info.plist 中，`io.flutter.embedded_views_preview` 是否为 YES。

Android Manifest merge failed 编译失败？

请打开 `/example/android/app/src/main/AndroidManifest.xml` 文件。

1. 将 `xmlns:tools="http://schemas.android.com/tools"` 加入到 manifest 中。
2. 将 `tools:replace="android:label"` 加入到 application 中。



说明：

更多常见问题，请参见 Flutter 相关问题。

Unity

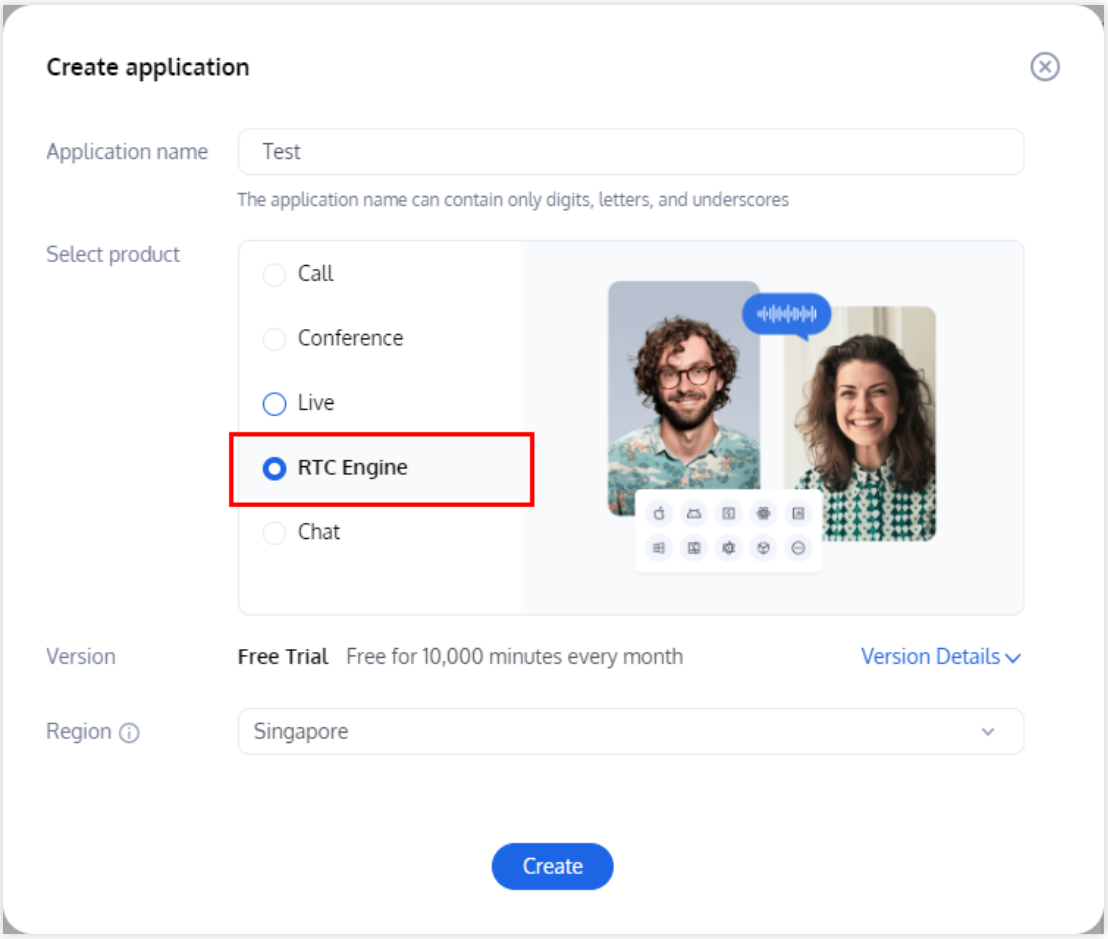
最近更新时间：2024-05-13 10:43:55

这个示例项目演示了如何在 Unity 中快速集成 TRTC SDK，实现在游戏中的音视频通话。
在这个示例项目中包含了以下功能：
加入通话和离开通话。
自定义视频渲染。
设备管理、音乐特效和人声特效。
说明：
具体 API 功能参数说明，请参见 Unity API 概览。
Unity 建议版本： 2020.2.1f1c1。
目前支持 Android、iOS、Windows、Mac(Mac 还在内测中)平台。
需要包含 `Android Build Support` 、 `iOS Build Support` 、 `Windows Build Support` 和 `MacOS Build Support` 模块。
其中 iOS 端开发还需要：
Xcode 11.0及以上版本。
请确保您的项目已设置有效的开发者签名。

操作步骤

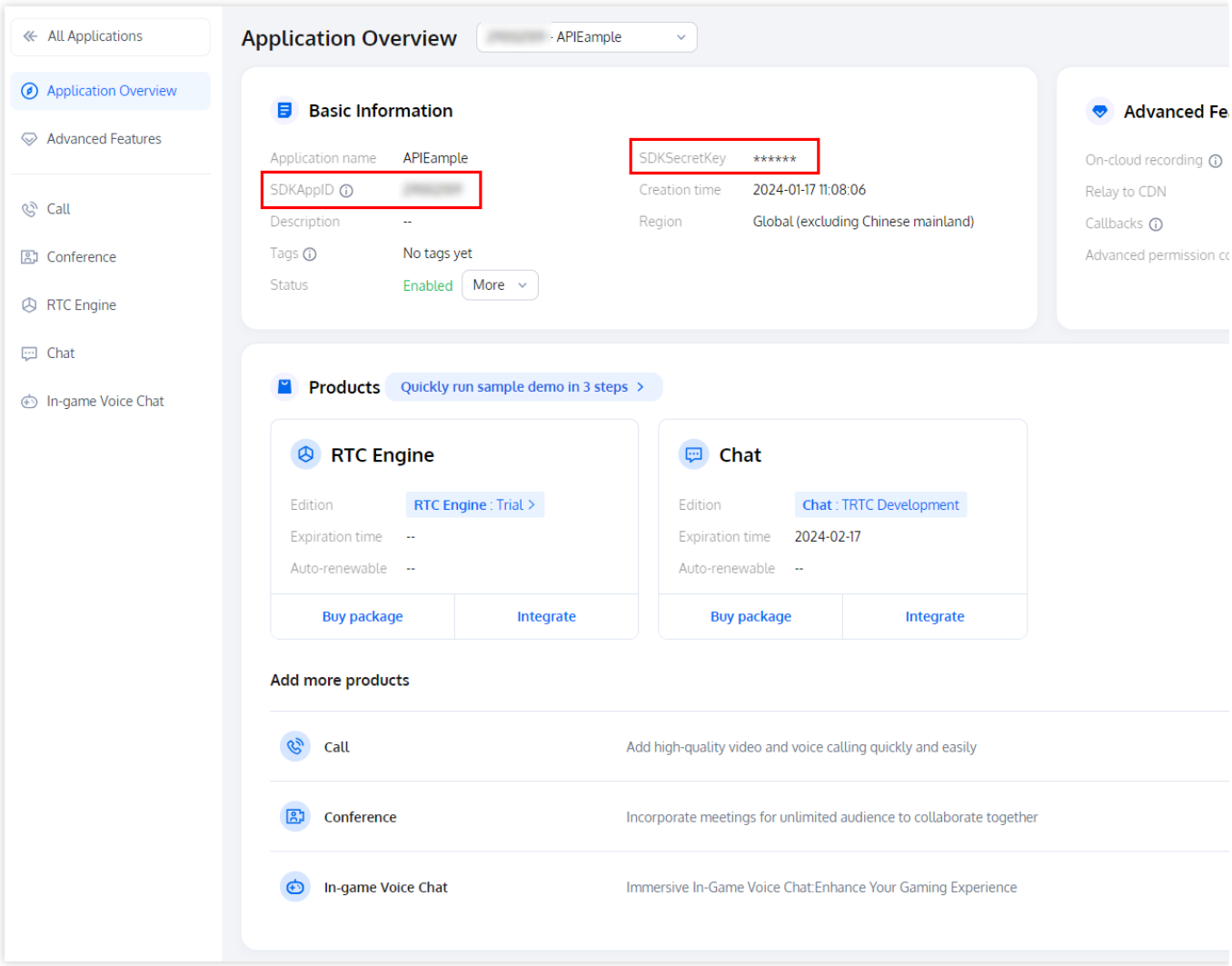
步骤 1：创建新的应用

1. 登录 实时音视频控制台总览页，单击**创建应用**。
2. 在弹窗页面中选择 RTC Engine，输入应用名称，然后单击**创建**。



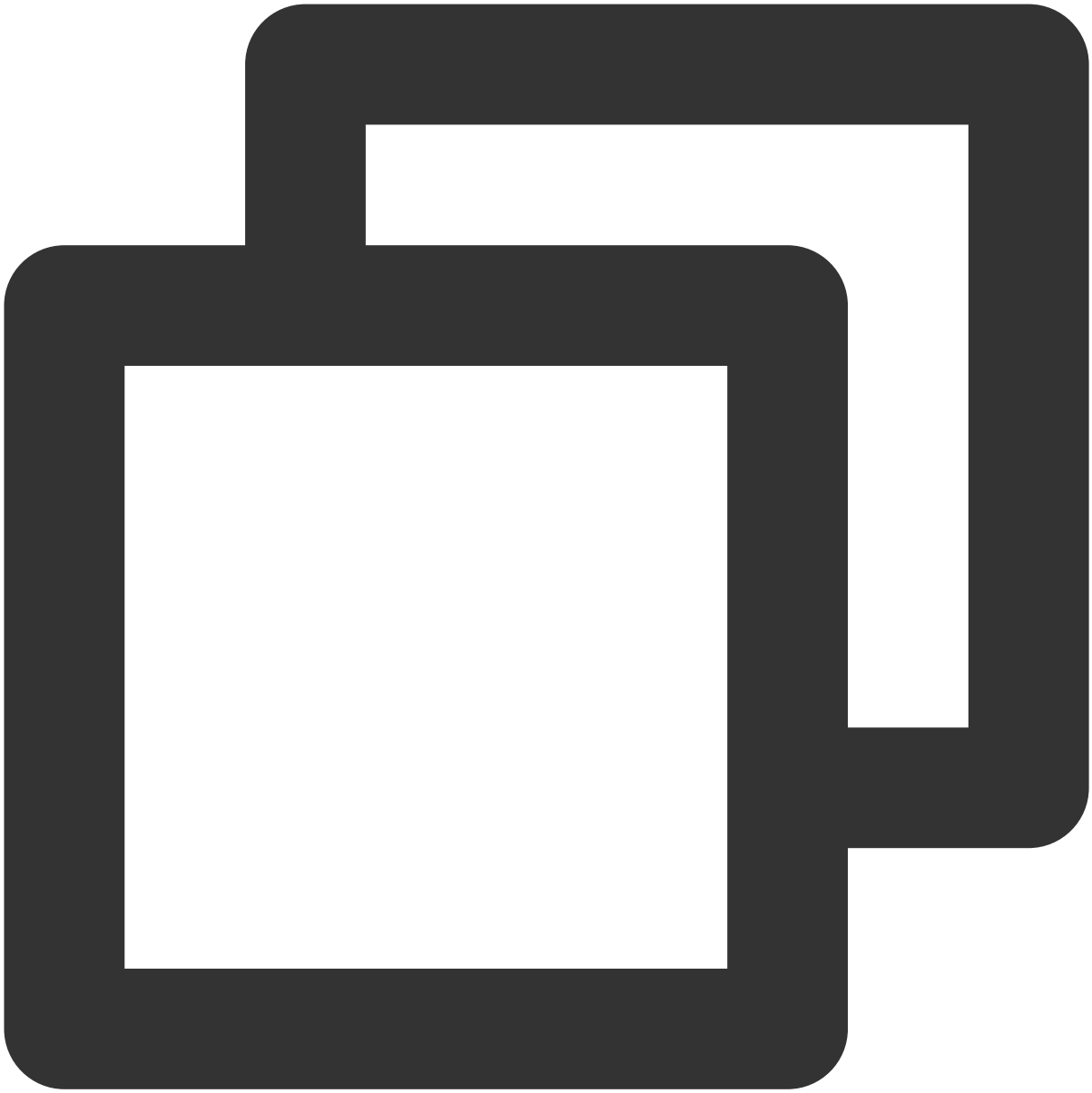
步骤2：获取 SDKAppID 和 SecretKey

在应用创建成功后，您可以在基本信息中获取到您的 `SDKAppID` 和 `SDKSecretKey`，体验 Demo 需要用到这两个信息。



步骤 3：下载示例代码

1. 您可以根据自己的业务平台，前往Github下载相关 SDK 及配套的 Demo 源码。



```
git clone https://github.com/LiteAVSDK/TRTC_Unity.git
```

2. 导入SDK的步骤可以参考：Unity 导入SDK。

步骤 4：配置工程

1. 打开下载完成的示例代码，找到并打开 `TRTC-Simple-Demo/Assets/TRTCSDK/Demo/Tools/GenerateTestUserSig.cs` 文件，设置 `GenerateTestUserSig.cs` 文件中的相关参数：

SDKAPPID：默认为 PLACEHOLDER ，请设置为实际的 SDKAppID。

SDKSECRETKEY：默认为 PLACEHOLDER，请设置为实际的 Secret key。

2. 此时工程配置已经完成，您可以单击**下一步**。

说明：

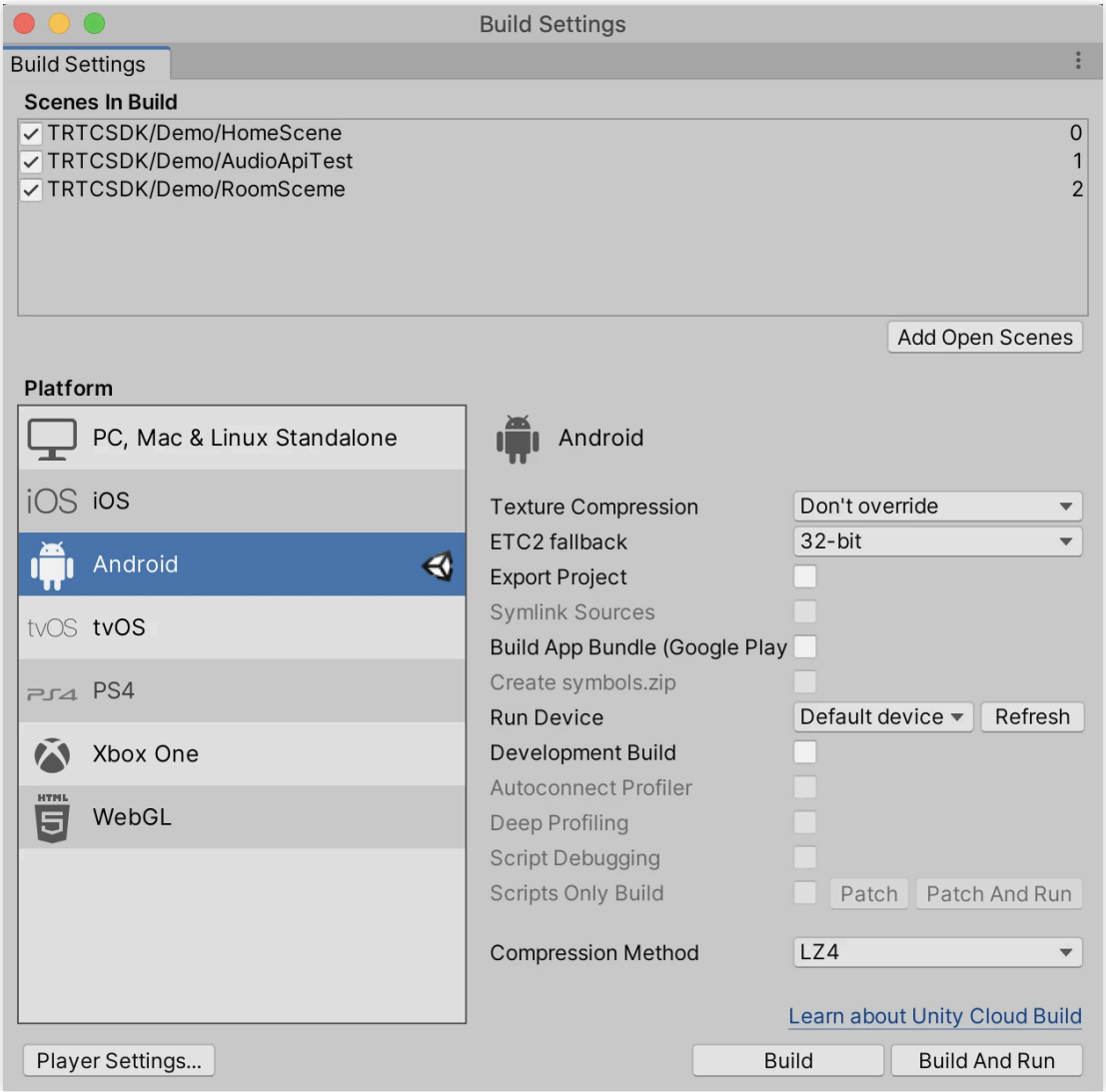
本文提到的生成 UserSig 的方案是在客户端代码中配置 SDKSECRETKEY，该方法中 SDKSECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 TRTC-Simple-Demo 和功能调试**。

正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

步骤 5：编译运行

Android 平台

1. 配置 Unity Editor，单击**File** > **Build Setting**，切换至 Android。

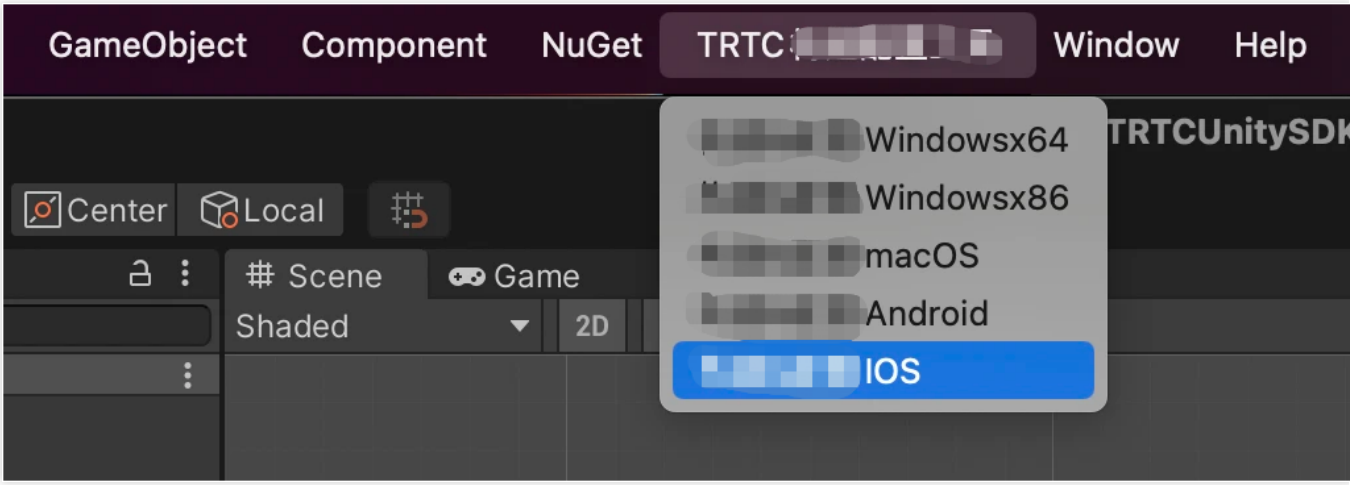


2. 连接 Android 真机，单击**Build And Run**，Demo 就能跑起来。

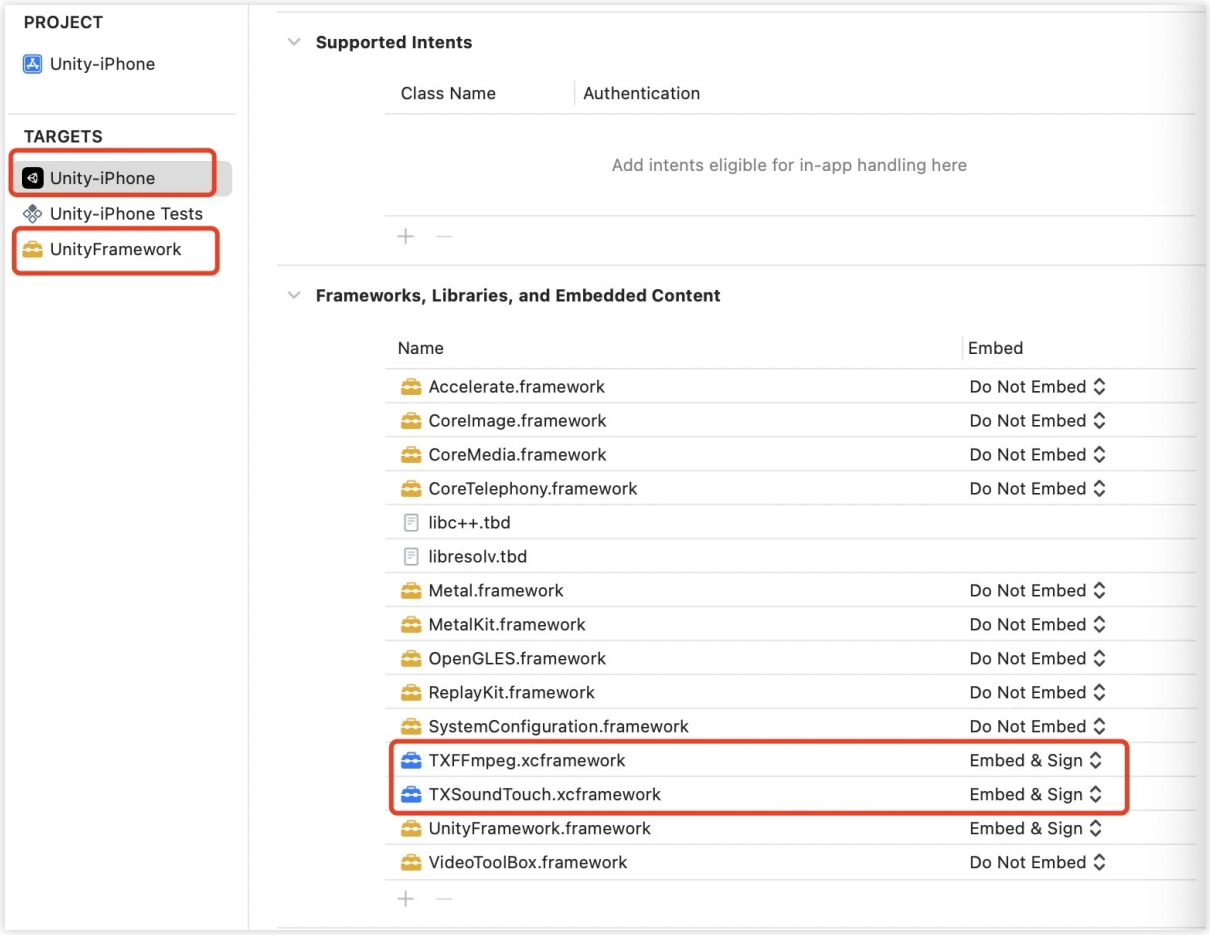
3. 接口测试，需要先点击调用 enterRoom ，然后自行测试其他相关，数据展示窗口显示点击调用成功，另外一个窗口显示回调信息。

iOS 平台

1. 打开 `TRTC 构建配置工具` （可在 Unity 编辑器顶部导航栏找到）
2. 点击 `构建&配置 IOS` ，等待项目生成完成



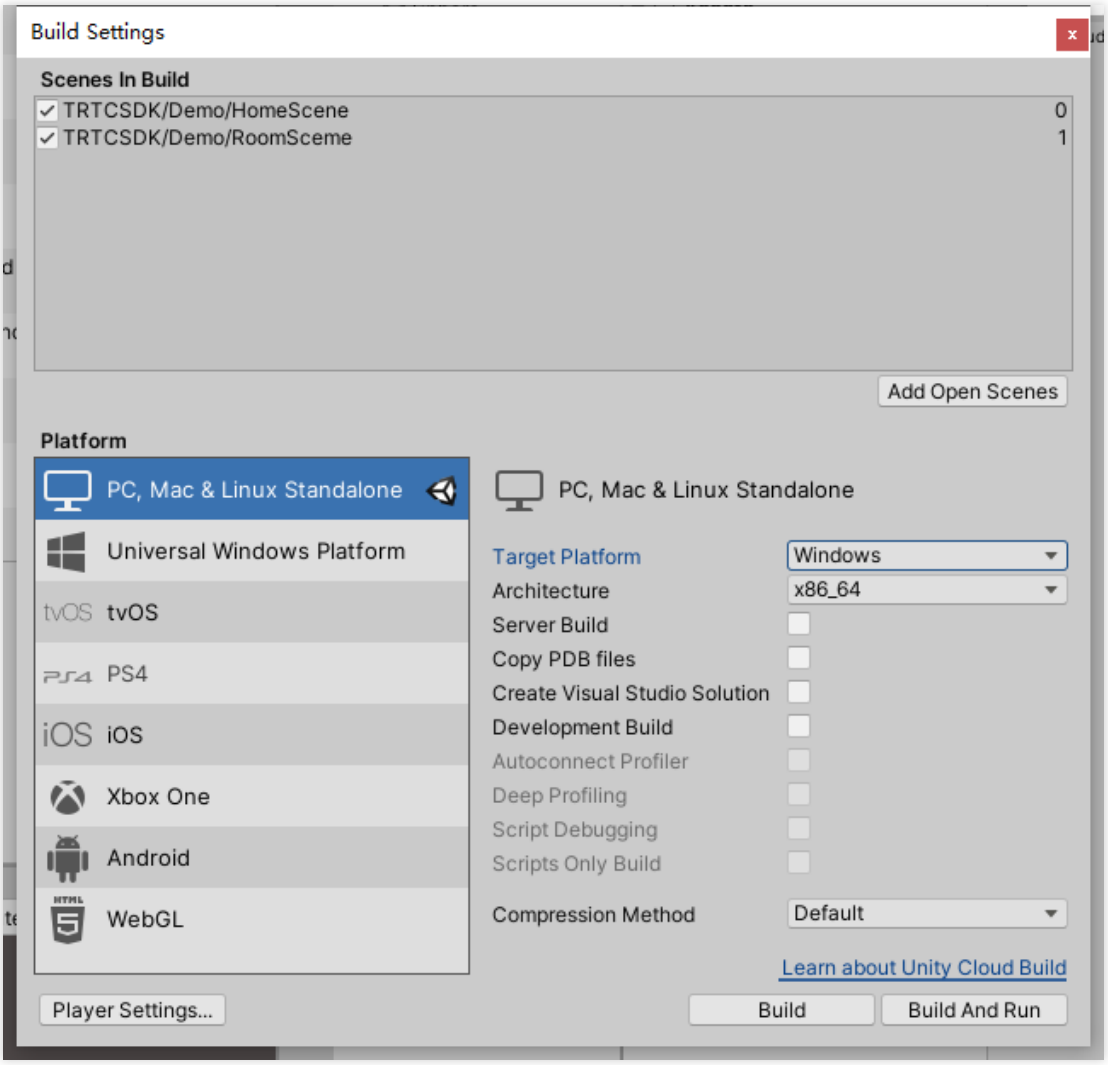
3. 使用 xcode 打开生成好的 Unity-iPhone.xcodeproj 项目
4. 下载TRTC 底层 sdk，单击 General，选择 Frameworks,Libraries,and Embedded Content，单击底下的“+”号图标依次添加所需要动态库 FFmpeg.xcframework、SoundTouch.xcframework，选择 Embed & Sign。



5. 连接 iOS 真机进行调试

Windows 平台

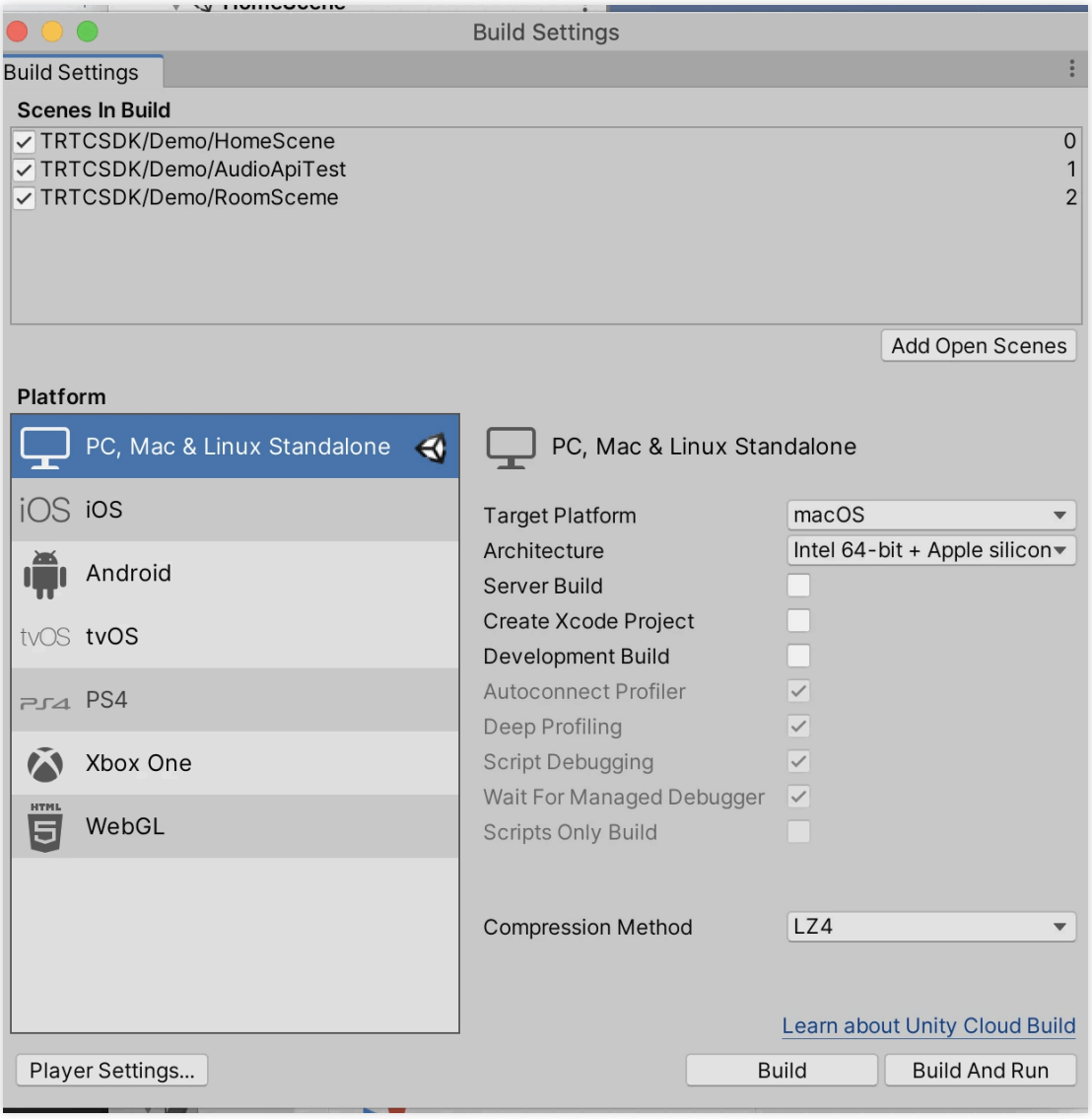
1. 配置 Unity Editor，单击**File** > **Build Setting**，切换至 `PC, Mac & Linux Standalone`，Target Platform 选择 Windows。



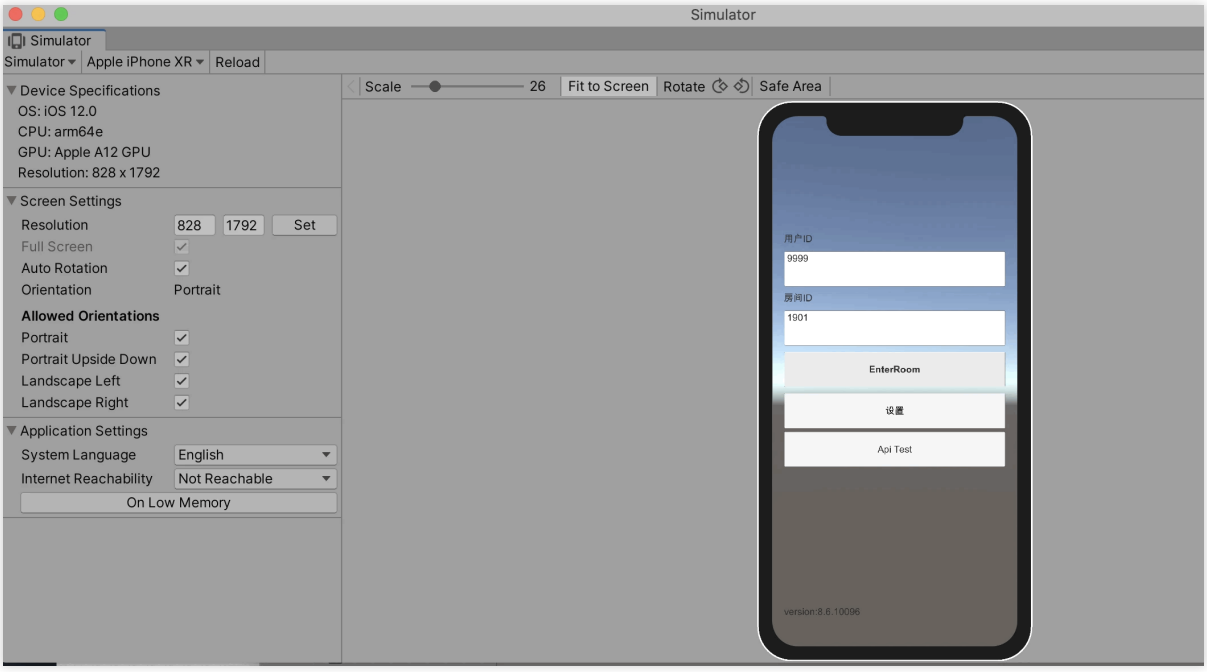
2. 单击**Build And Run**，Demo 就能跑起来。

macOS 平台

1. 配置 Unity Editor，单击**File** > **Build Setting**，切换至 `PC, Mac & Linux Standalone`，Target Platform 选择 macOS。



2. 单击 **Build And Run**，Demo 就能跑起来。
3. 使用 Unity Editor 模拟器运行，先要安装 `Device Simulator Package` 。
4. 单击 **Window** > **General** > **Device Simulator**



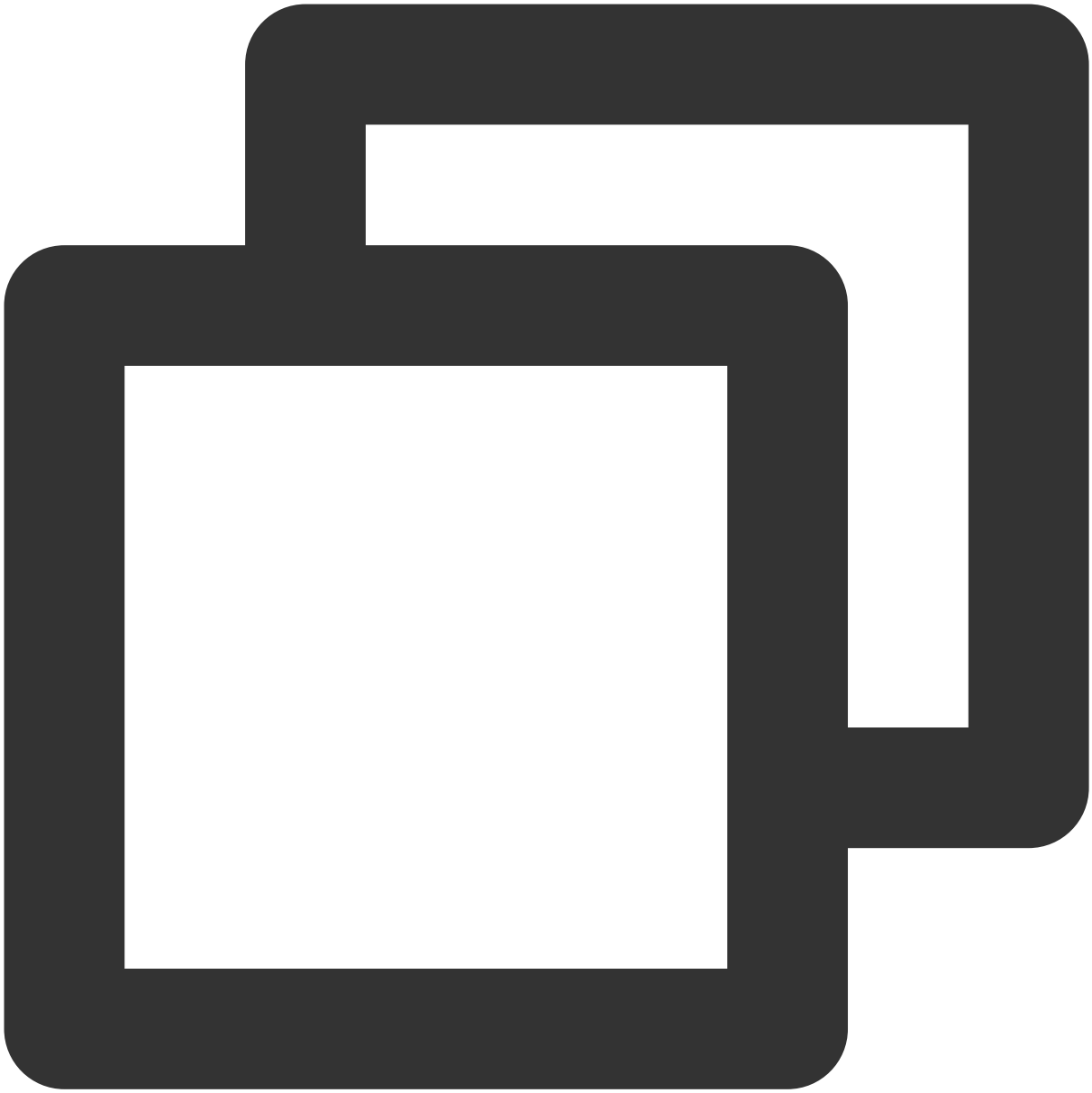
Demo示例

Demo 里面包含了已上线的大部分 API，可以测试和作为调用参考，API 文档参见 SDK API（Unity）。

说明：

UI 可能会有部分调整更新，请以最新版为准。

目录结构



```
├─Assets
├── Editor                          // Unity 编辑器脚本
│    ├── BuildScript.cs             // Unity 编辑器build菜单
│    ├── IosPostProcess.cs          // Unity 编辑器构建ios应用脚本
├── Plugins
│    ├── Android
│    │    ├── AndroidManifest.xml   //Android应用配置文件
├── StreamingAssets                 // Unity Demo 音视频流文件
├── TRTCSDK
     ├── Demo                       // Unity 示例 Demo
     ├── SDK                        // TRTC Unity SDK
```

```
        ├── Implement              // TRTC Unity SDK 实现
        ├── Include                // TRTC Unity SDK 头文件
        └── Plugins                // TRTC Unity SDK 不同平台底层实现
```

React Native

最近更新时间：2024-05-13 10:43:55

本文主要介绍如何快速运行腾讯云 TRTC Demo（React Native）。

环境要求

ReactNative 0.63 及以上版本。
Node & Watchman，node版本需在 v12 以上。
Android 端开发：
Android Studio 3.5及以上版本。
App 要求 Android 4.1及以上版本设备。
iOS & macOS 端开发：
Xcode 11.0及以上版本。
osx 系统版本要求 10.11 及以上版本。
请确保您的项目已设置有效的开发者签名。
环境安装请参见 官方文档。

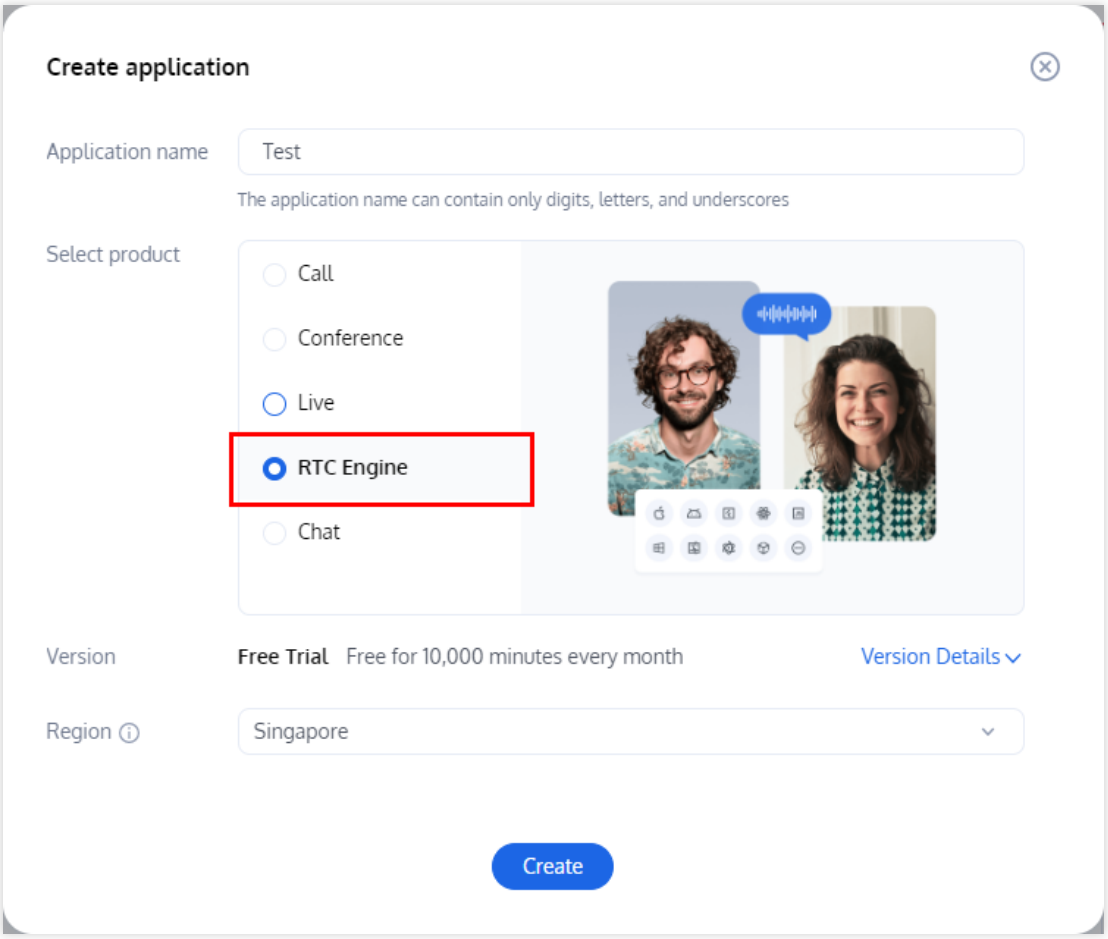
前提条件

您已 注册腾讯云 账号，并完成实名认证。

操作步骤

步骤1：创建新的应用

1. 登录 实时音视频控制台总览页，单击**创建应用**。
2. 在弹窗页面中选择 RTC Engine，输入应用名称，然后单击**创建**。



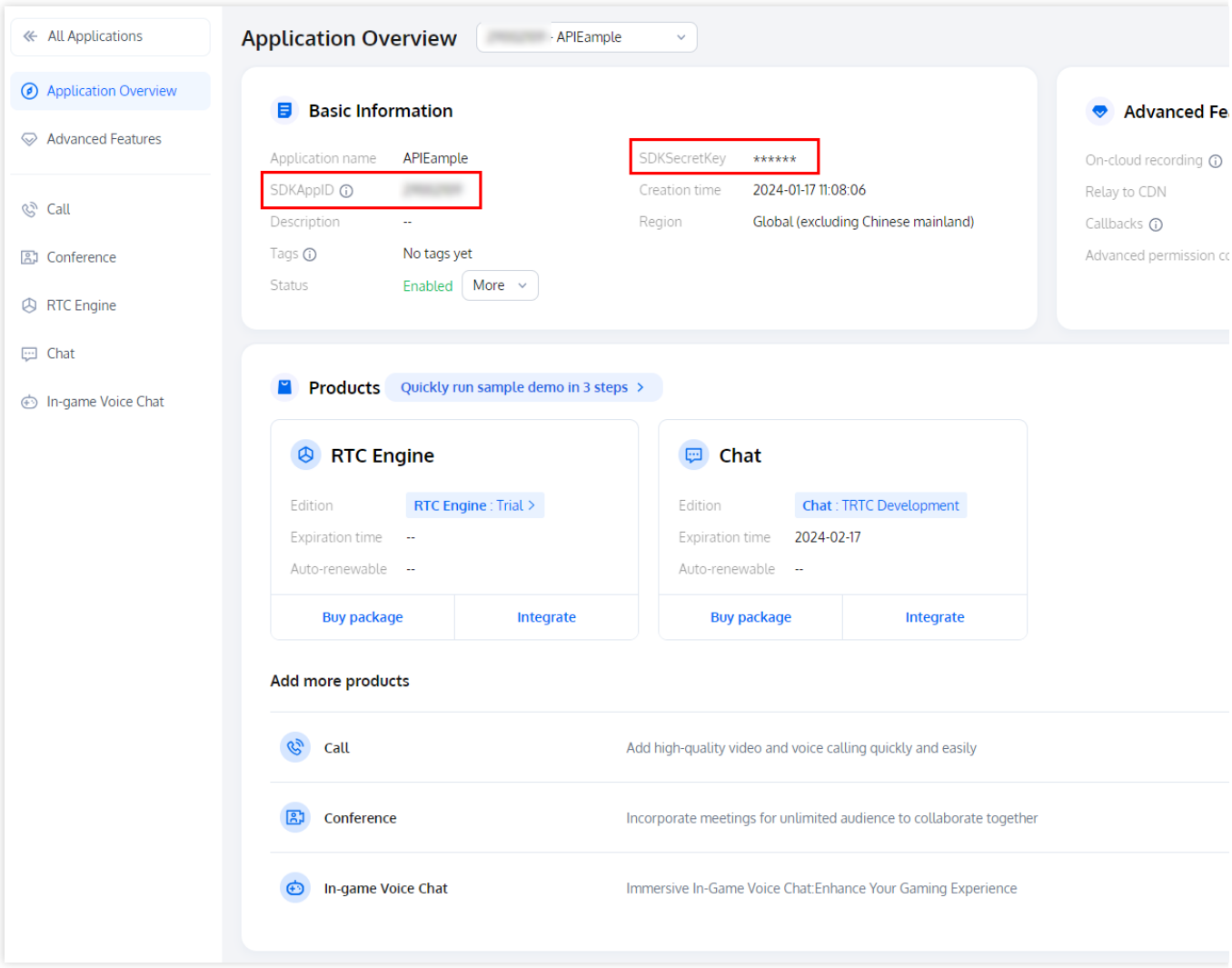
说明：

应用名称只能包含数字、中英文字符和下划线，长度不能超过15个字符。

标签用于标识和组织您在腾讯云的各种资源。例如：企业可能有多个业务部门，每个部门有1个或多个 TRTC 应用，这时，企业可以通过给 TRTC 应用添加标签来标记部门信息。标签并非必选项，您可根据实际业务需求添加或编辑。

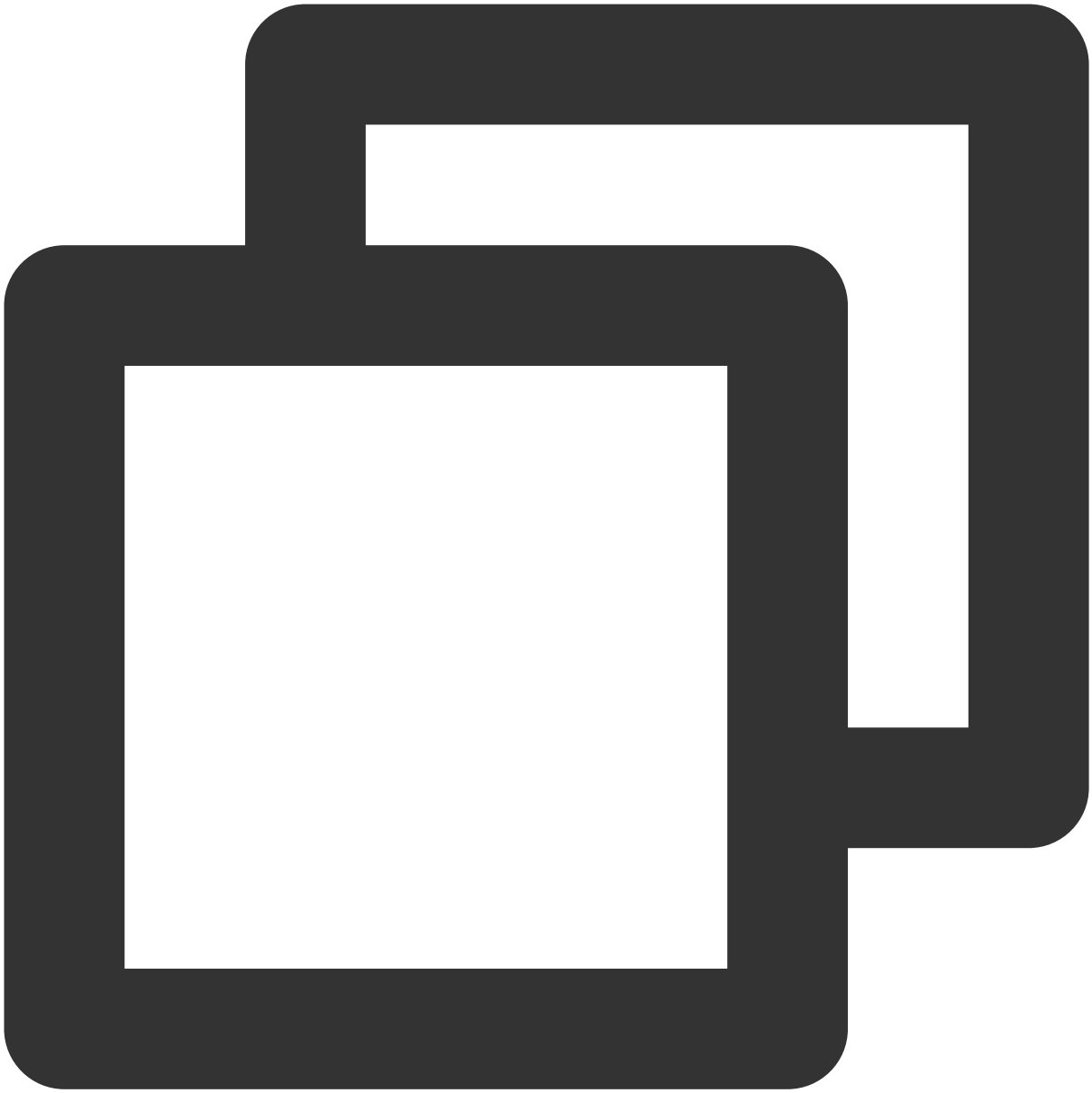
步骤2：获取 SDKAppID 和 SecretKey

在应用创建成功后，您可以在基本信息中获取到您的 `SDKAppID` 和 `SDKSecretKey`，体验 Demo 需要用到这两个信息。



步骤3：下载 SDK 和 Demo 源码

1. 根据实际业务需求，前往 Github 下载对应平台的示例代码。



```
git clone https://github.com/LiteAVSDK/TRTC_ReactNative.git
```

2. 导入SDK的步骤可以参考：React Native 导入SDK

步骤4：配置 Demo 工程文件

1. 找到并打开 `/debug/config.js` 文件。
2. 设置 `SDKAPPID` 和 `SDKSECRETKEY` 参数：

SDKAPPID：默认为 PLACEHOLDER，请设置为实际的 SDKAppID。

SDKSECRETKEY：默认为 PLACEHOLDER，请设置为实际的密钥信息。

说明：

本文提到的生成 UserSig 的方案是在客户端代码中配置 SDKSECRETKEY，该方法中 SDKSECRETKEY 很容易被反编译逆向破解，一旦您的密钥泄露，攻击者就可以盗用您的腾讯云流量，因此**该方法仅适合本地跑通 Demo 和功能调试**。

正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向 App 的接口，在需要 UserSig 时由您的 App 向业务服务器发起请求获取动态 UserSig。更多详情请参见 服务端生成 UserSig。

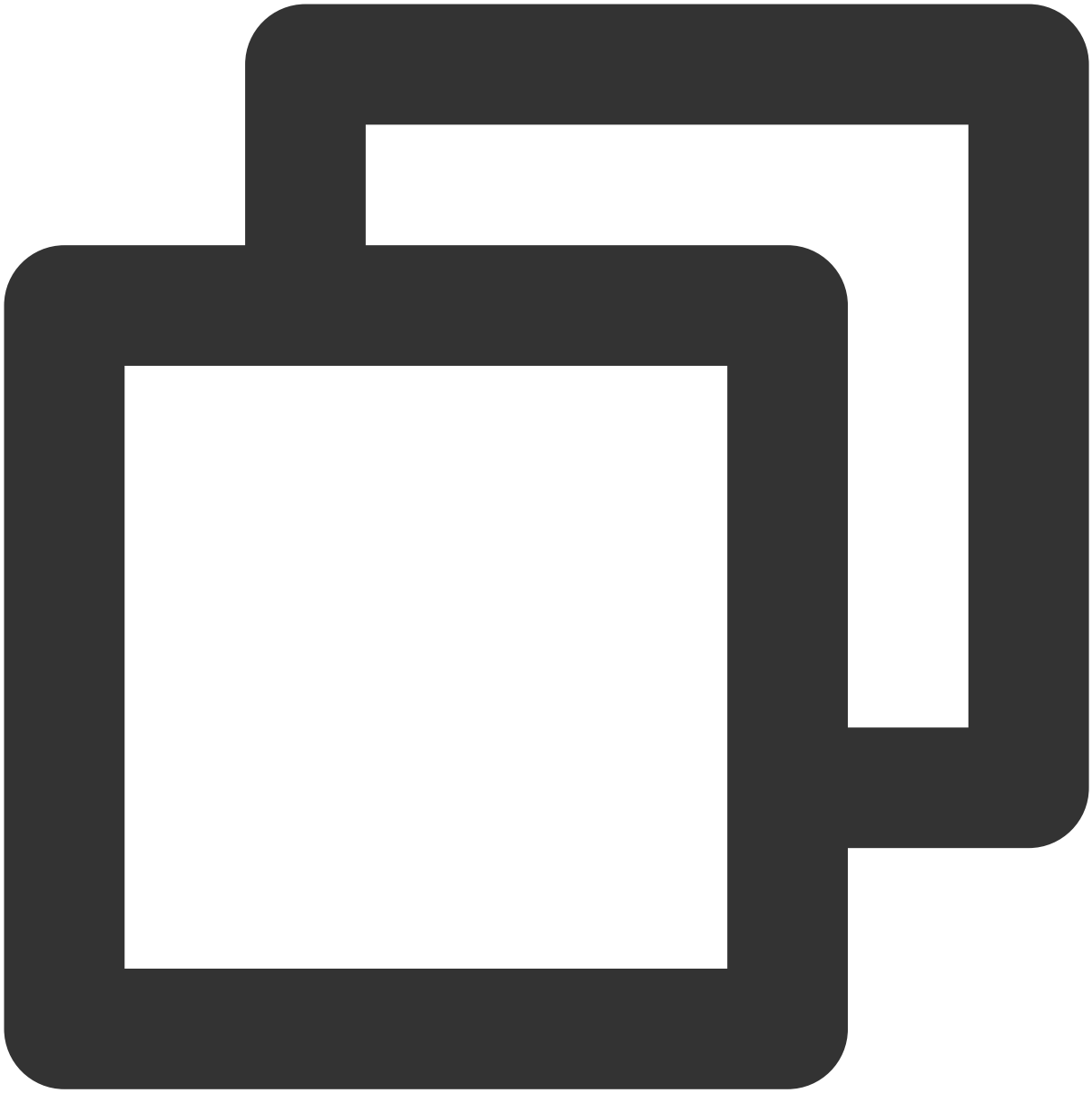
步骤5：权限配置

需要配置 App 权限才能运行。

Android 端

iOS 端

1. 在 `AndroidManifest.xml` 中配置 App 的权限，TRTC SDK 需要以下权限：



```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-feature android:name="android.hardware.camera" />
```

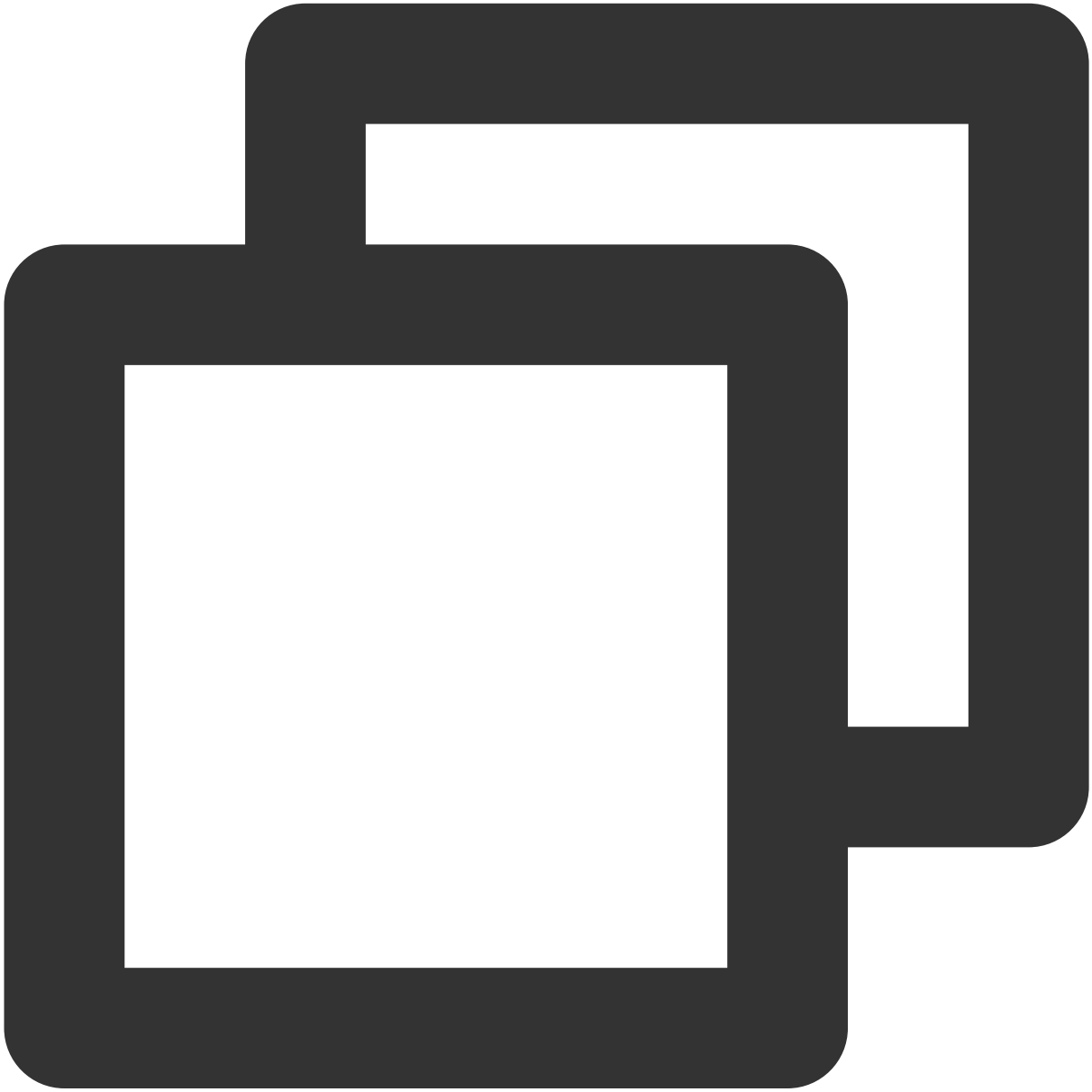
版权所有：腾讯云计算（北京）有限责任公司

```
<uses-feature android:name="android.hardware.camera.autofocus" />
```

注意：

请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

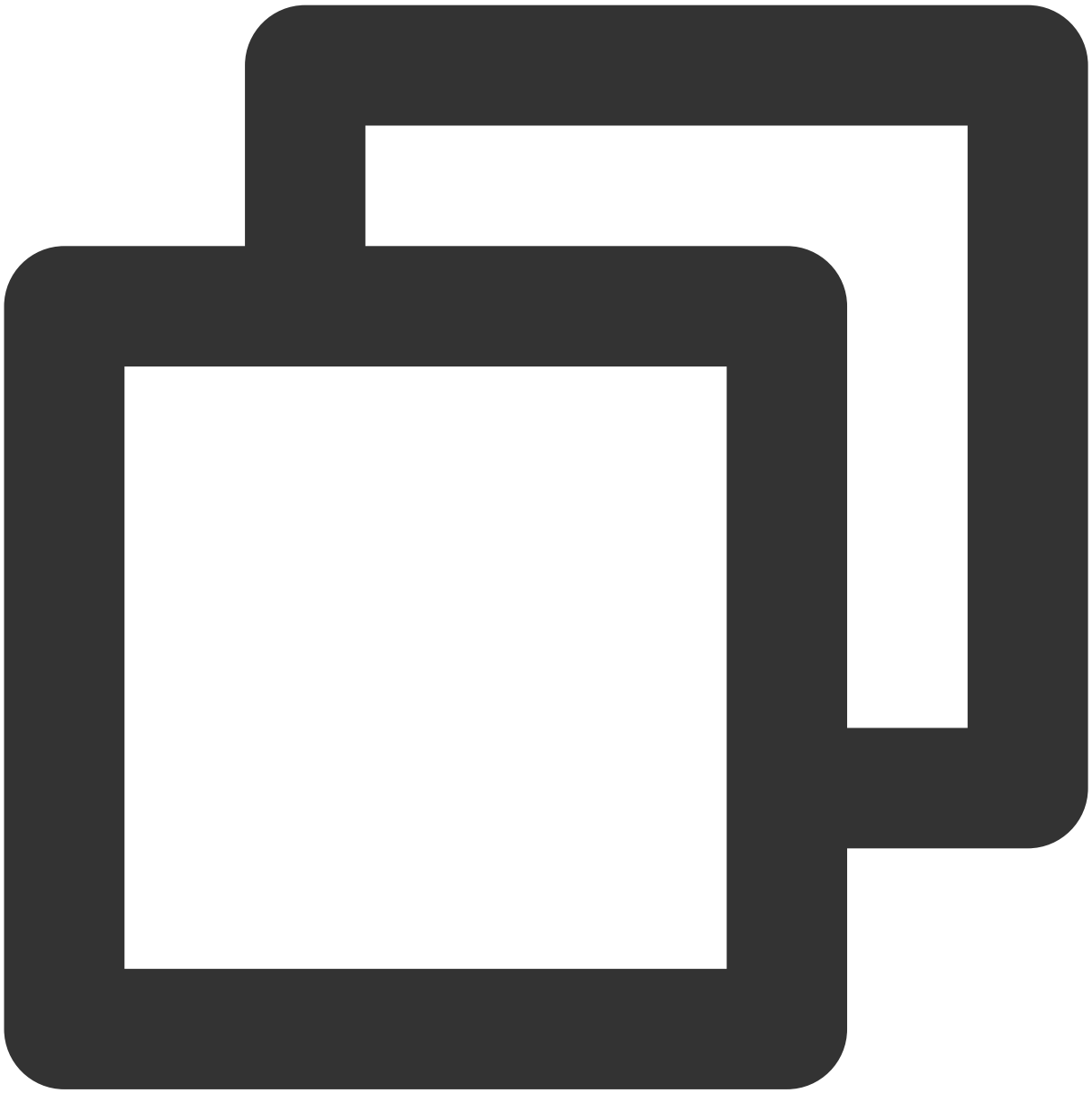
2. Android 端音视频权限需要手动申请。



```
if (Platform.OS === 'android') {
  await PermissionsAndroid.requestMultiple([
 PermissionsAndroid.PERMISSIONS.RECORD_AUDIO, //音频需要
 PermissionsAndroid.PERMISSIONS.CAMERA, // 视频需要
  ]);
```

```
}
```

在 `Info.plist` 中配置 App 的权限，TRTC SDK 需要以下权限：



```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

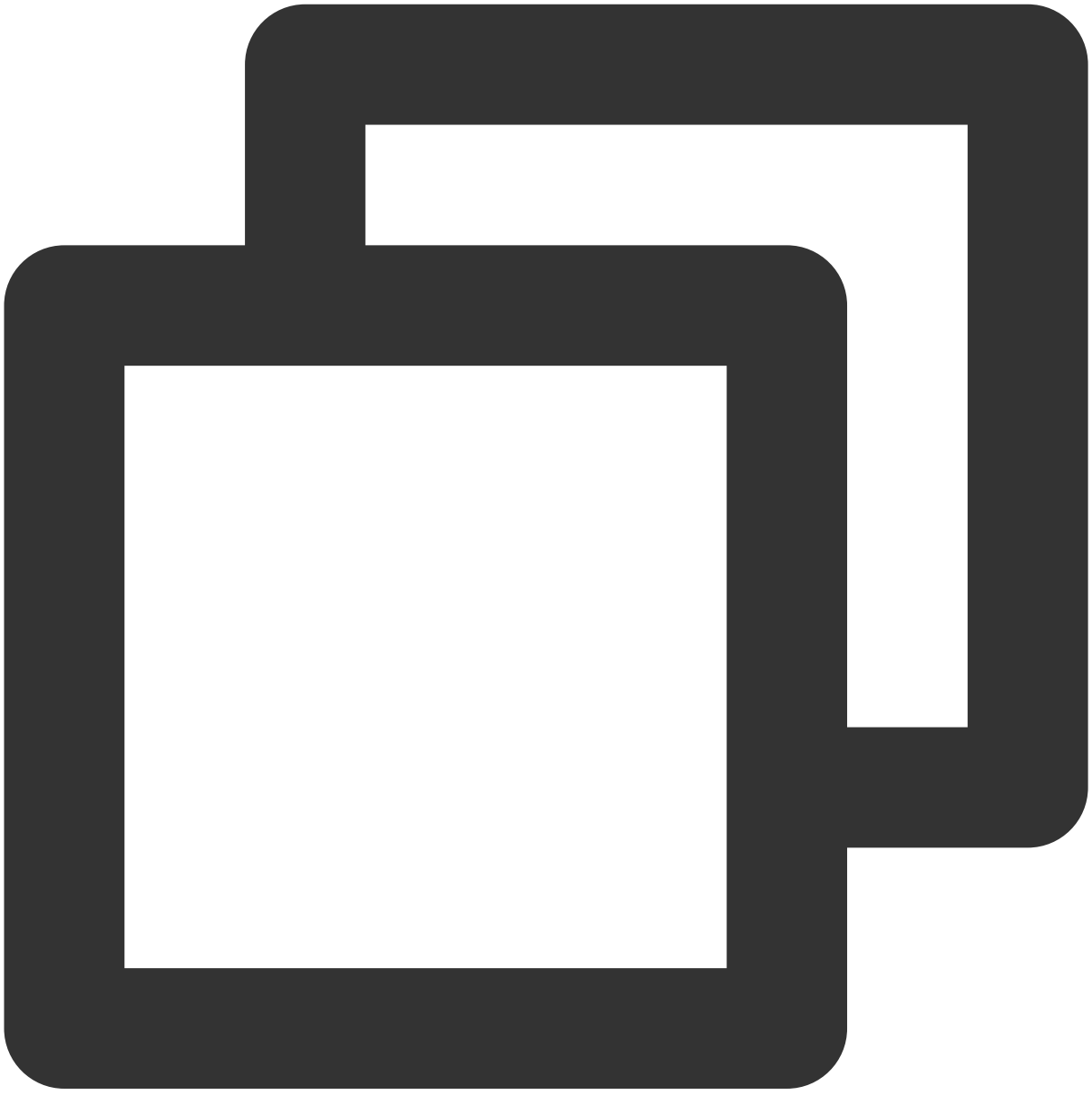
步骤6：编译运行

启动 Metro，在您的 React Native 项目目录下运行 `npx react-native start` 。

Android 端

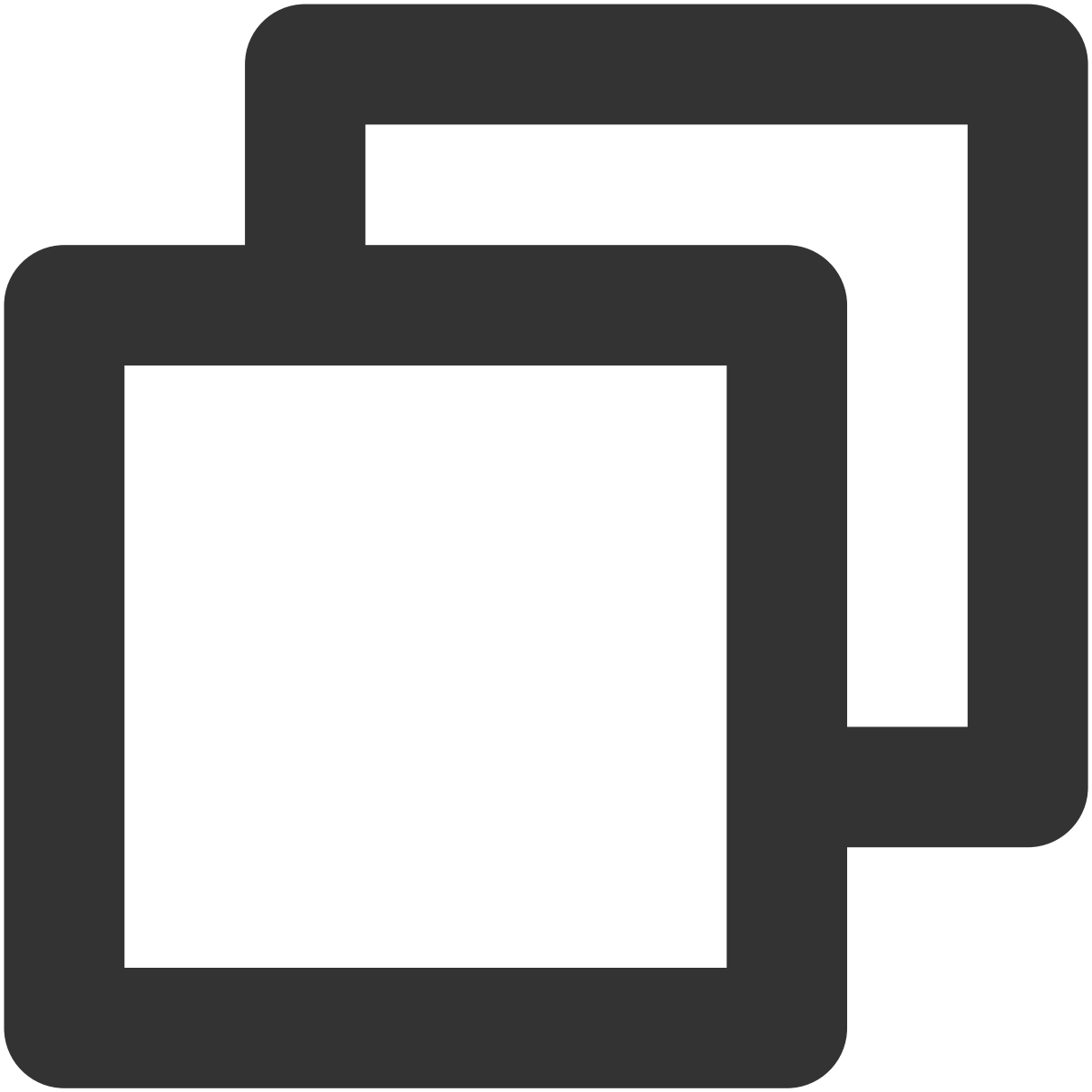
iOS 端

新开窗口，启动开发调试：



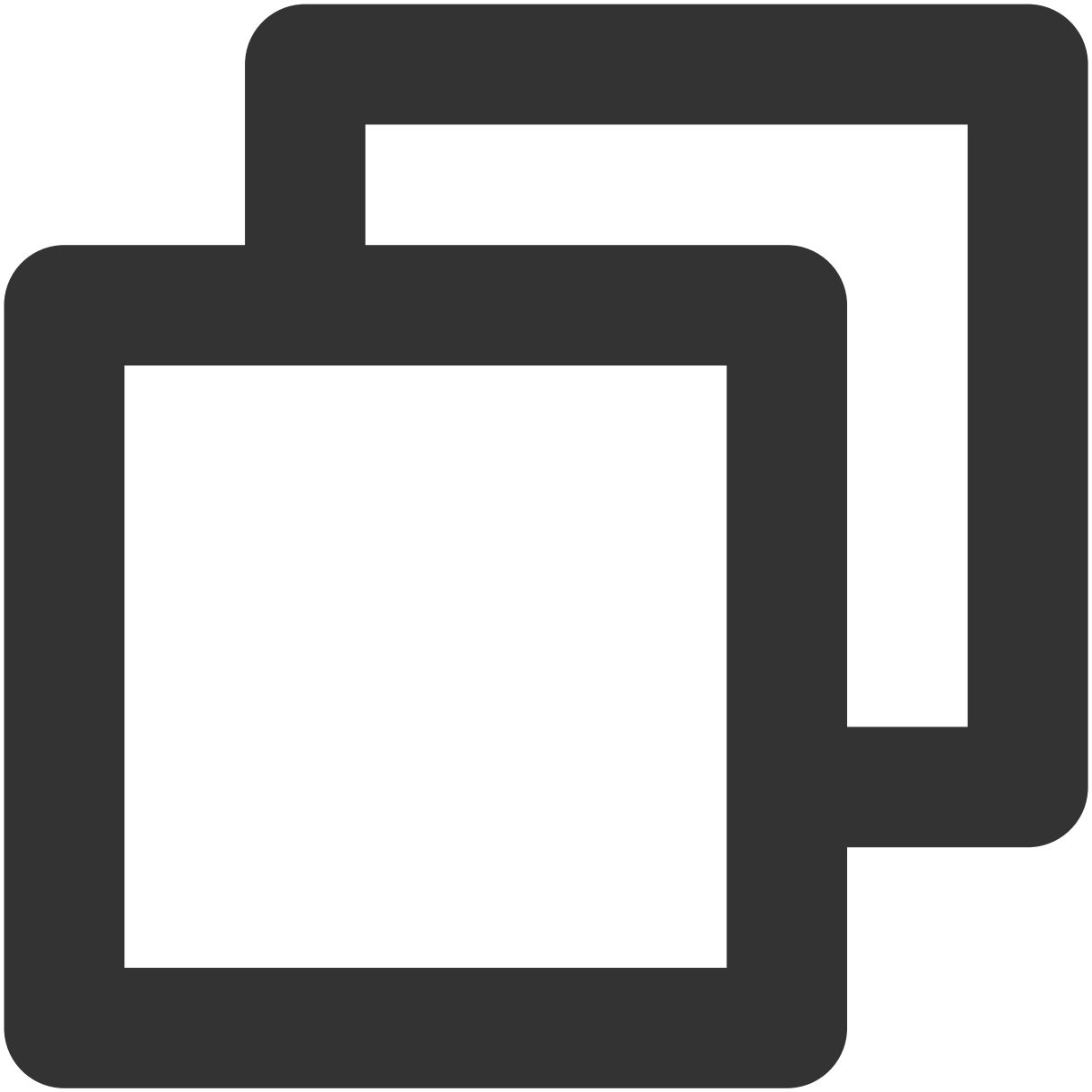
```
npx react-native run-android
```

1. 在 Demo iOS 目录里执行 `pod install` 安装依赖。
2. 在 Demo 目录下启动 Metros。



```
npx react-native start
```

3. 在 Demo 目录下新开窗口，启动开发调试（如果报错请打开 xcode 编译调试）。



```
npx react-native run-ios
```

API-使用指引
SDK 快速开始
Android

最近更新时间：2024-07-18 15:17:24

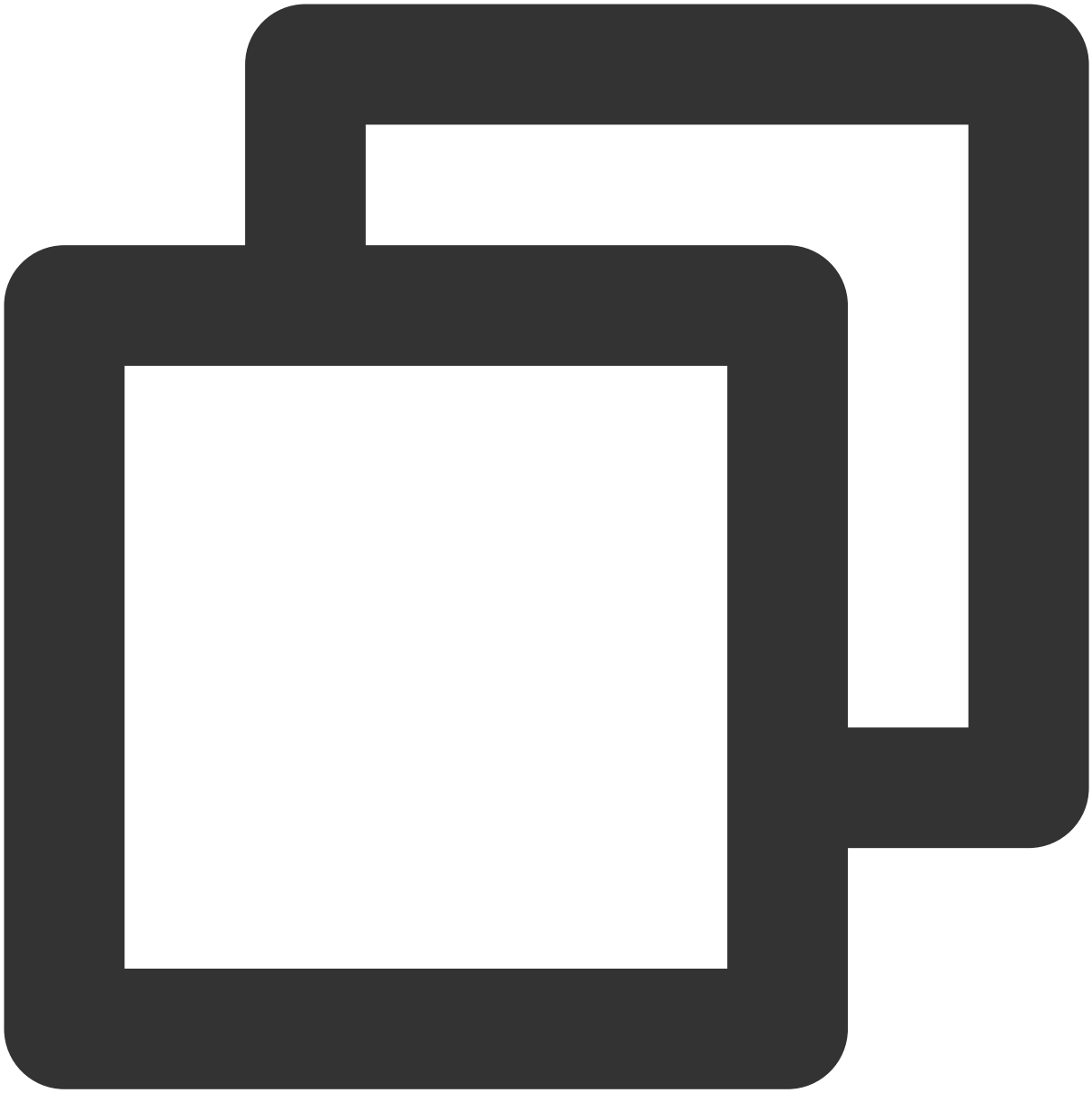
本文将介绍如何快速完成 Android RTC Engine 的接入，实现一个基本的音视频通话。

环境准备

Android Studio 3.5+。
Android 4.1（SDK API 16）及以上系统。

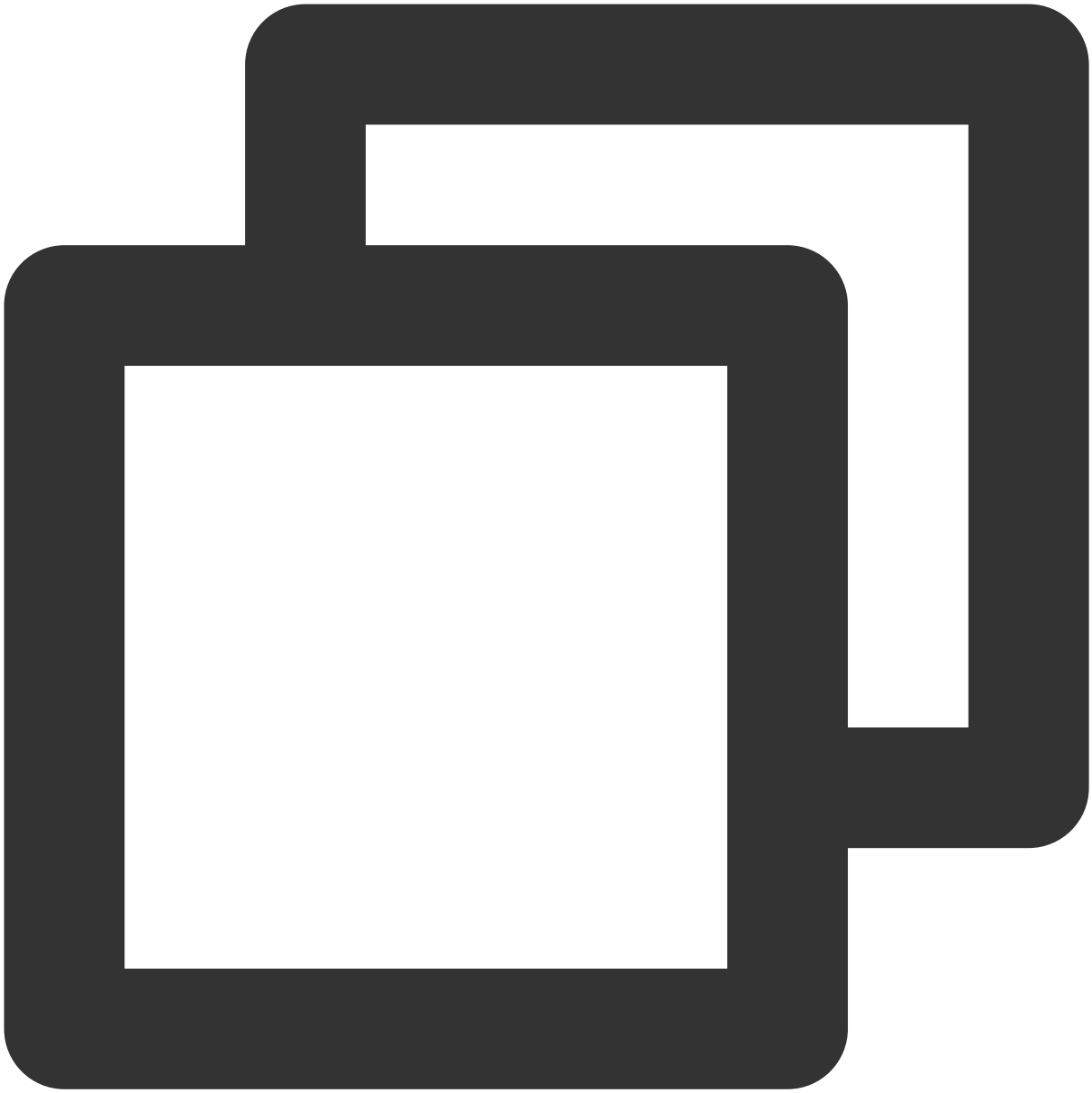
步骤1. 导入 SDK

1. 在 `app/build.gradle` 文件中添加 **TRTC SDK** 的依赖。



```
dependencies {
     implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'
}
```

2. 在 **defaultConfig** 中指定 App 使用的 **CPU** 架构。

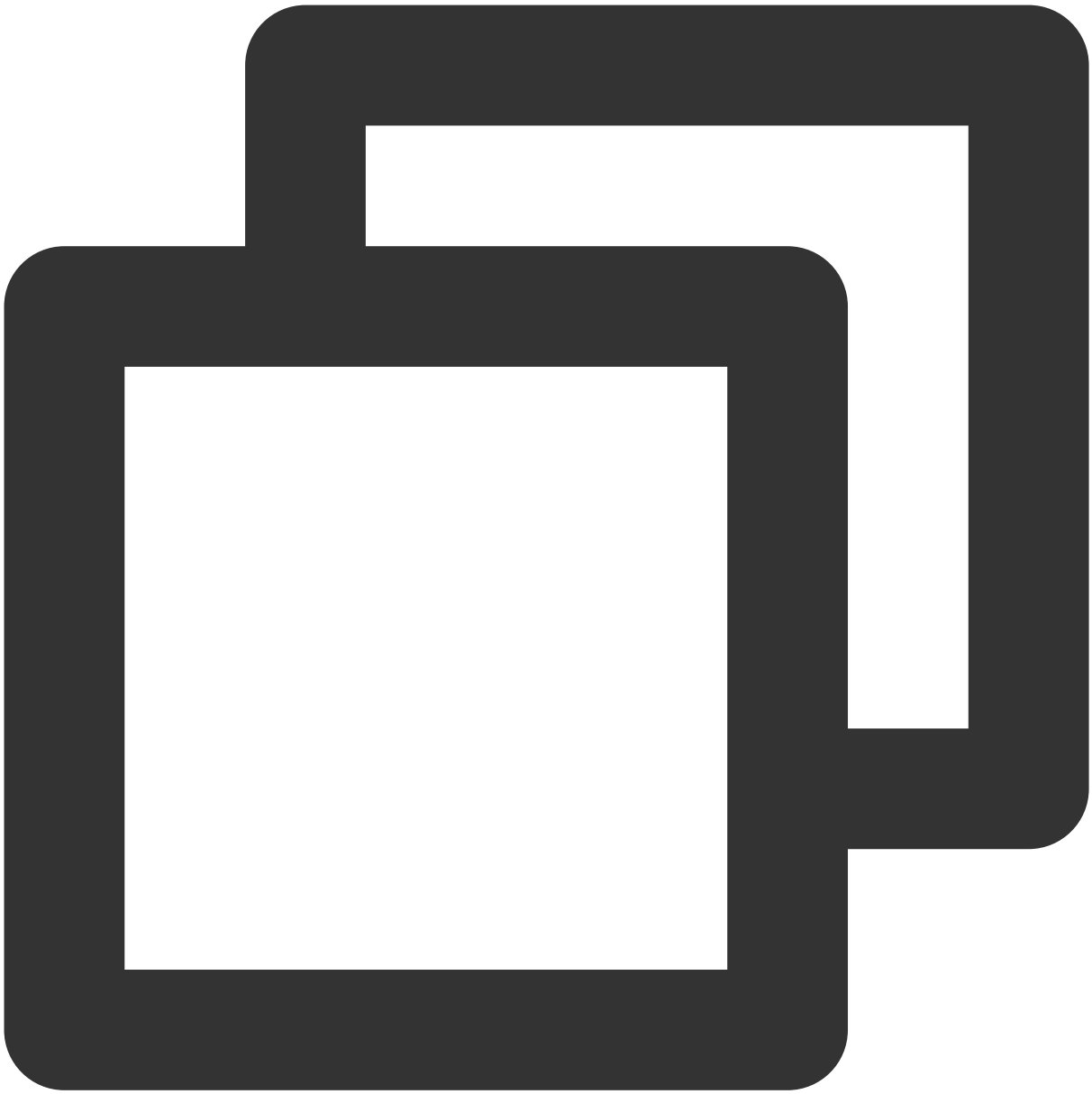


```
defaultConfig {
      ndk {
              abiFilters "armeabi-v7a", "arm64-v8a"
      }
}
```

完成以上配置后，单击 `Sync Now` 将自动将SDK集成到目标工程中。

步骤2. 配置项目

1. 在 `AndroidManifest.xml` 文件中配置 **TRTC SDK** 需要的权限。

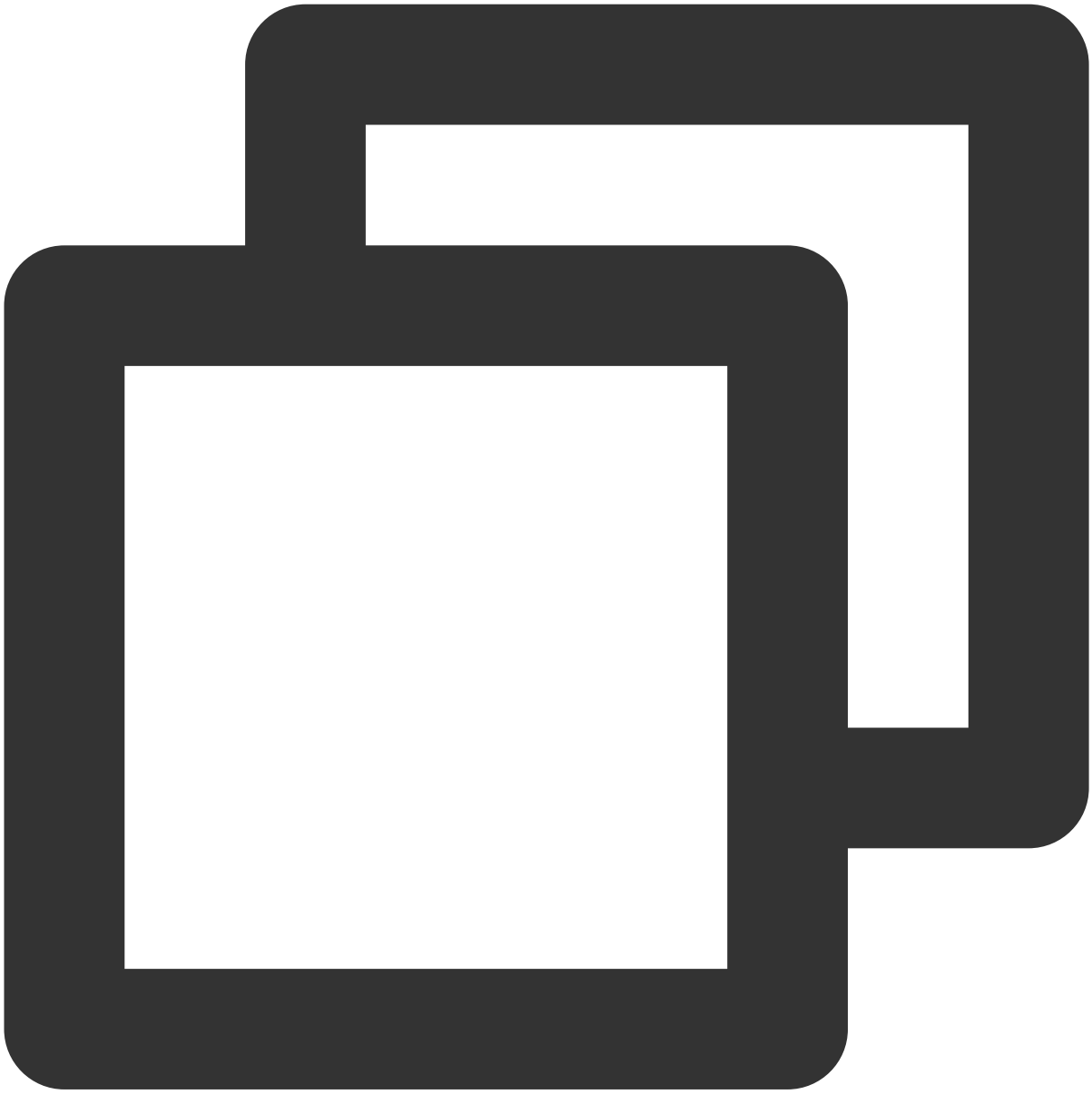


```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

注意：

请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

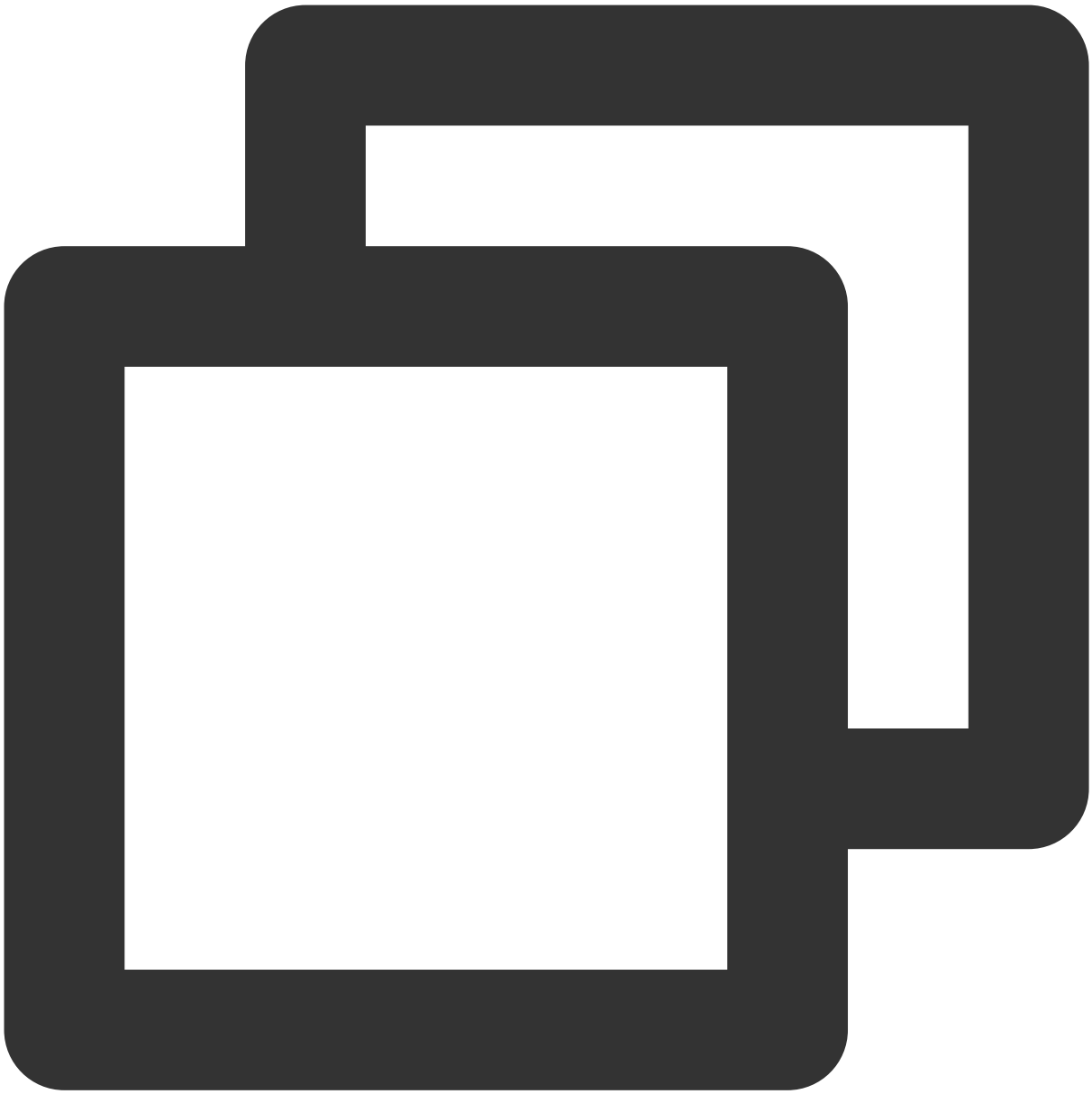
2. 在 `proguard-rules.pro` 文件中，将 **TRTC SDK** 相关类加入不混淆名单。



```
-keep class com.tencent.** { *; }
```

步骤3. 创建 TRTC 实例

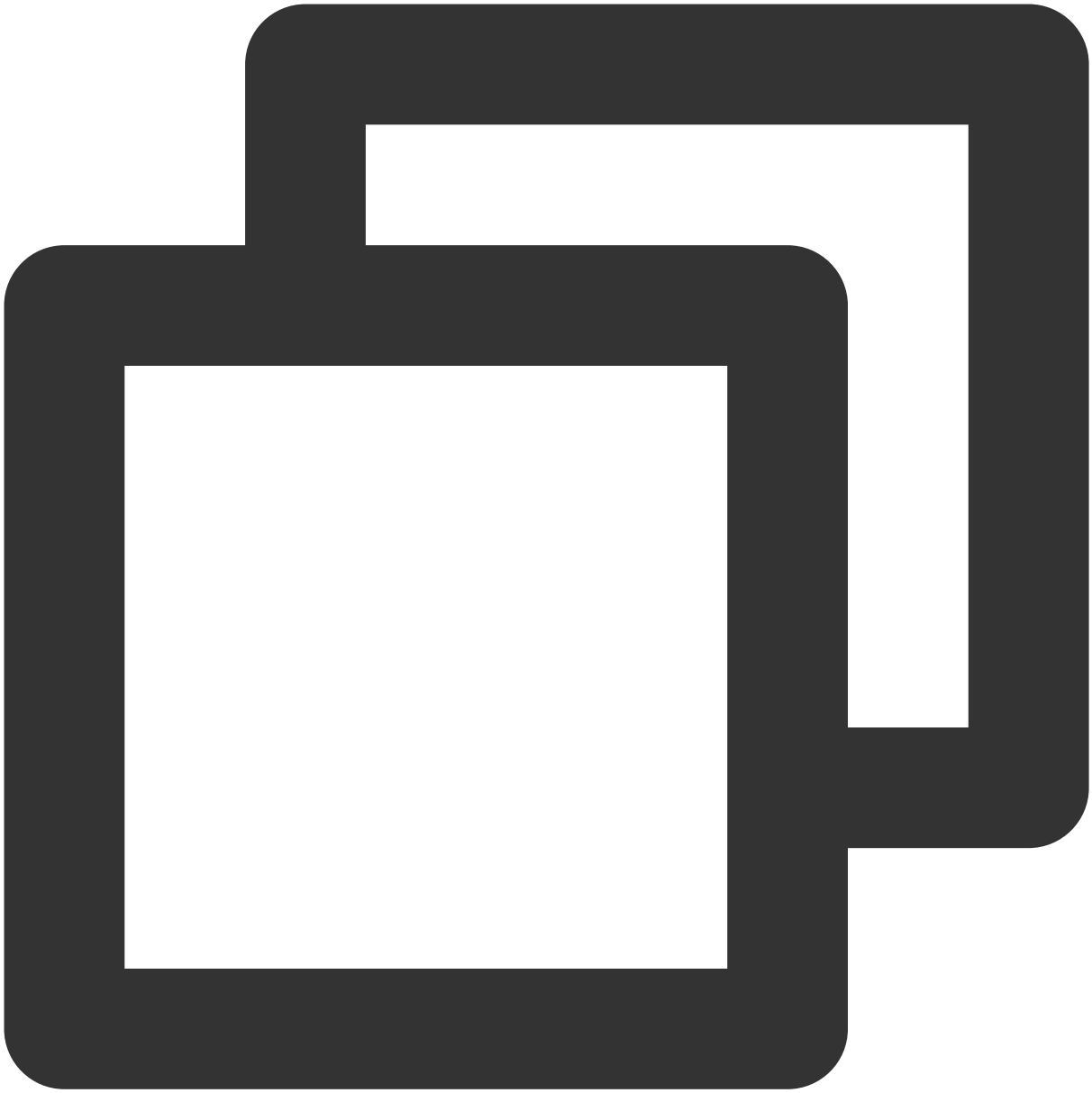
1. 声明成员变量



```
private final static String TAG = MainActivity.class.getSimpleName();
private static final int REQUEST_CAMERA_AND_MICROPHONE = 1;

private TRTCCloud mCloud; // Declare the mCloud member variable
```

2. 调用初始化接口创建 TRTC 实例并设置事件回调接口。



```
// Create trtc instance(singleton)  and set up event listeners
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
mCloud.setListener(new TRTCCloudListener() {

    // Listen to the "onError" event, and print logs for errors such as "Camera is
    @Override
    public void onError(int errCode, String errMsg, Bundle extraInfo) {
        super.onError(errCode, errMsg, extraInfo);
        if (errCode == TXLiteAVCode.ERR_CAMERA_NOT_AUTHORIZED) {
            Log.d(TAG, "Current application is not authorized to use the camera");
        }
```

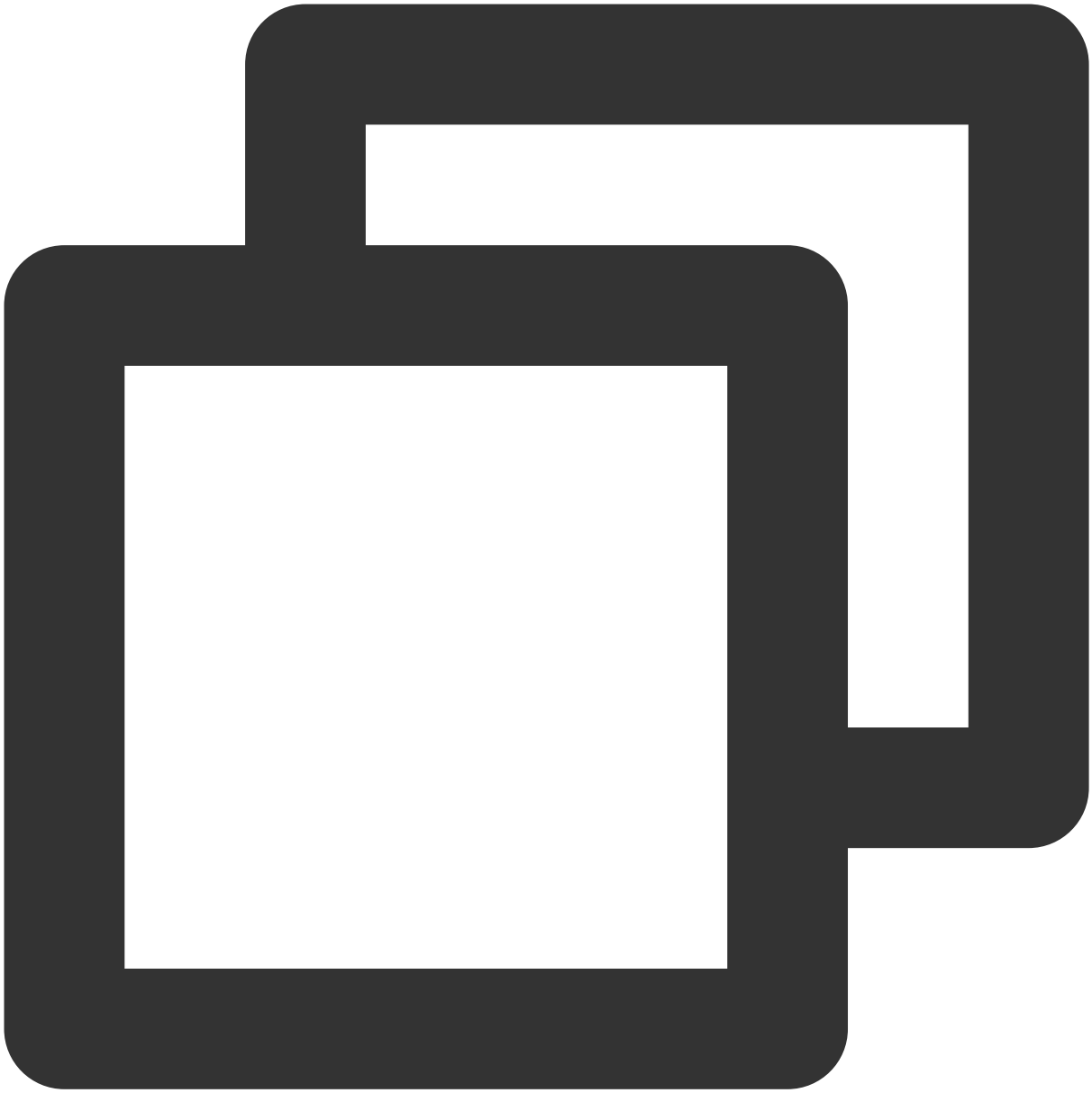
```
    }

    // Listen for the `onEnterRoom` event of the SDK and learn whether the room is
    @Override
    public void onEnterRoom(long result) {
        super.onEnterRoom(result);
        if (result > 0) {
            Log.d(TAG, "Enter room succeed");
        } else {
            Log.d(TAG, "Enter room failed");
        }

    }
});
```

步骤4. 进入房间

1. 请求 **CAMERA** 和 **MICROPHONE** 权限。



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // ... Other codes

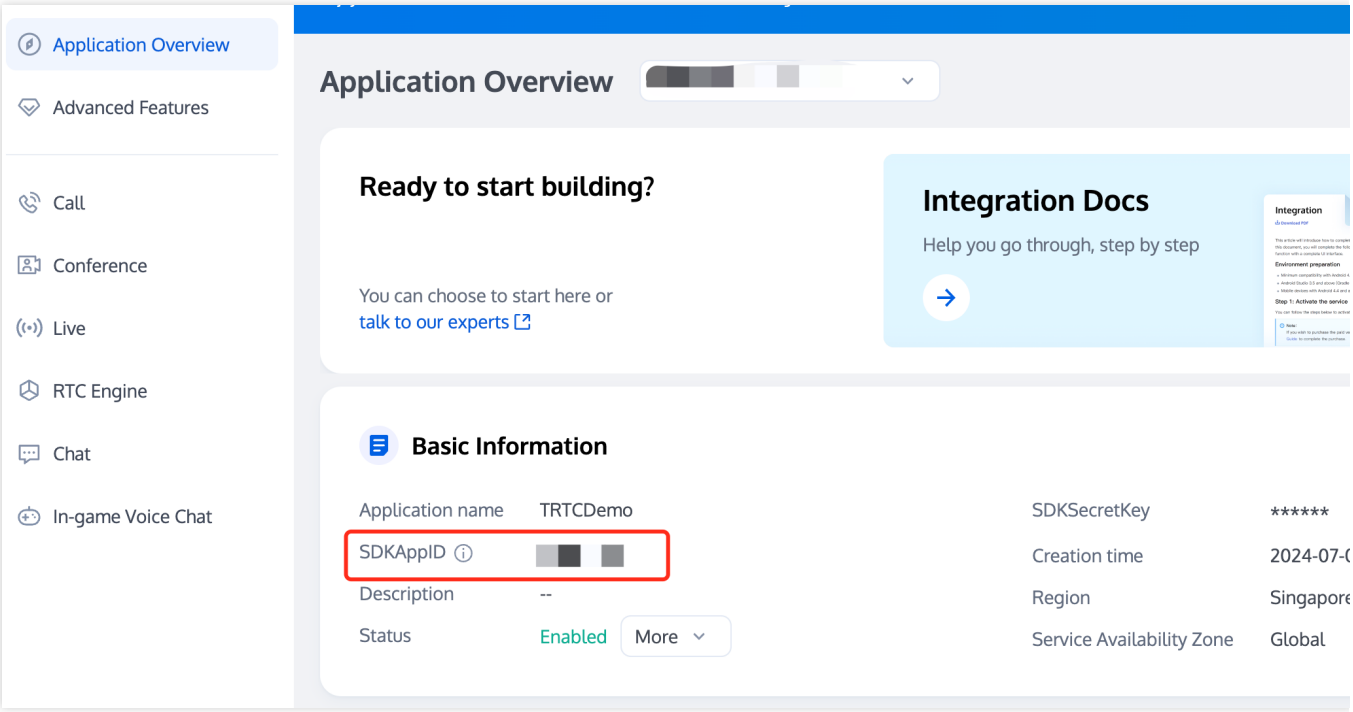
    // Request camera and microphone permissions
    requestCameraAndMicrophonePermission();
}

private void requestCameraAndMicrophonePermission() {
```

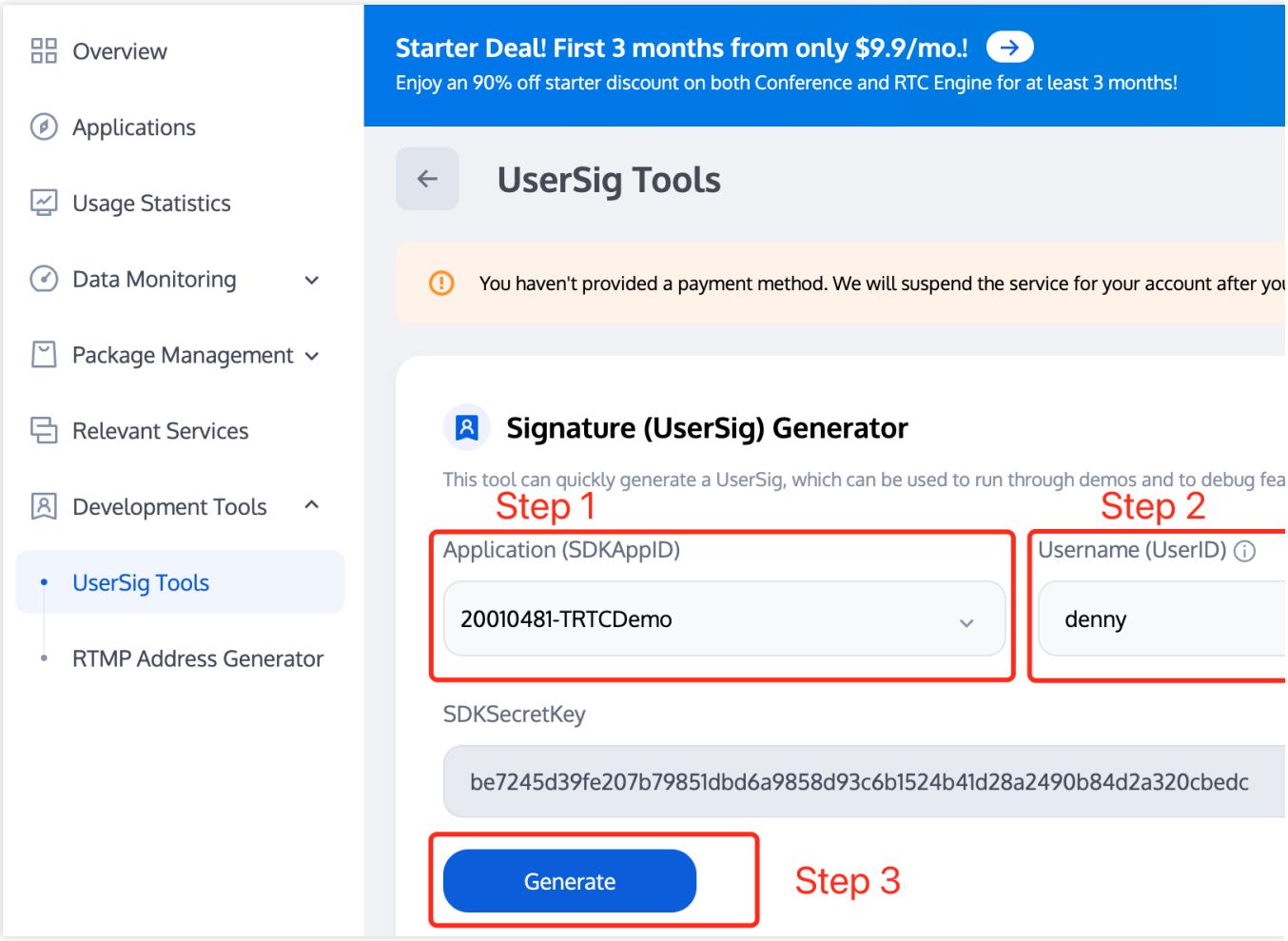
```
    String[] permissions = {android.Manifest.permission.CAMERA, android.Manifest.pe
    ActivityCompat.requestPermissions(this, permissions, REQUEST_CAMERA_AND_MICROPH
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissio
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CAMERA_AND_MICROPHONE) {
        boolean allPermissionsGranted = true;
        for (int grantResult : grantResults) {
            if (grantResult != PackageManager.PERMISSION_GRANTED) {
                allPermissionsGranted = false;
                break;
            }
        }
        if (allPermissionsGranted) {
            // All permissions are granted, you can start using the camera and micr
            // Here to create TRTC instances, enter the room, publish audio and vid
        } else {
            // Show a message to the user that the permissions are required to use
            Toast.makeText(this, "Camera and Microphone permissions are required",
        }
    }
}
```

2. 在Tencent RTC 控制台中单击 `Create Application` ，在 **Application Overview** 中获取 **SDKAppID** 。

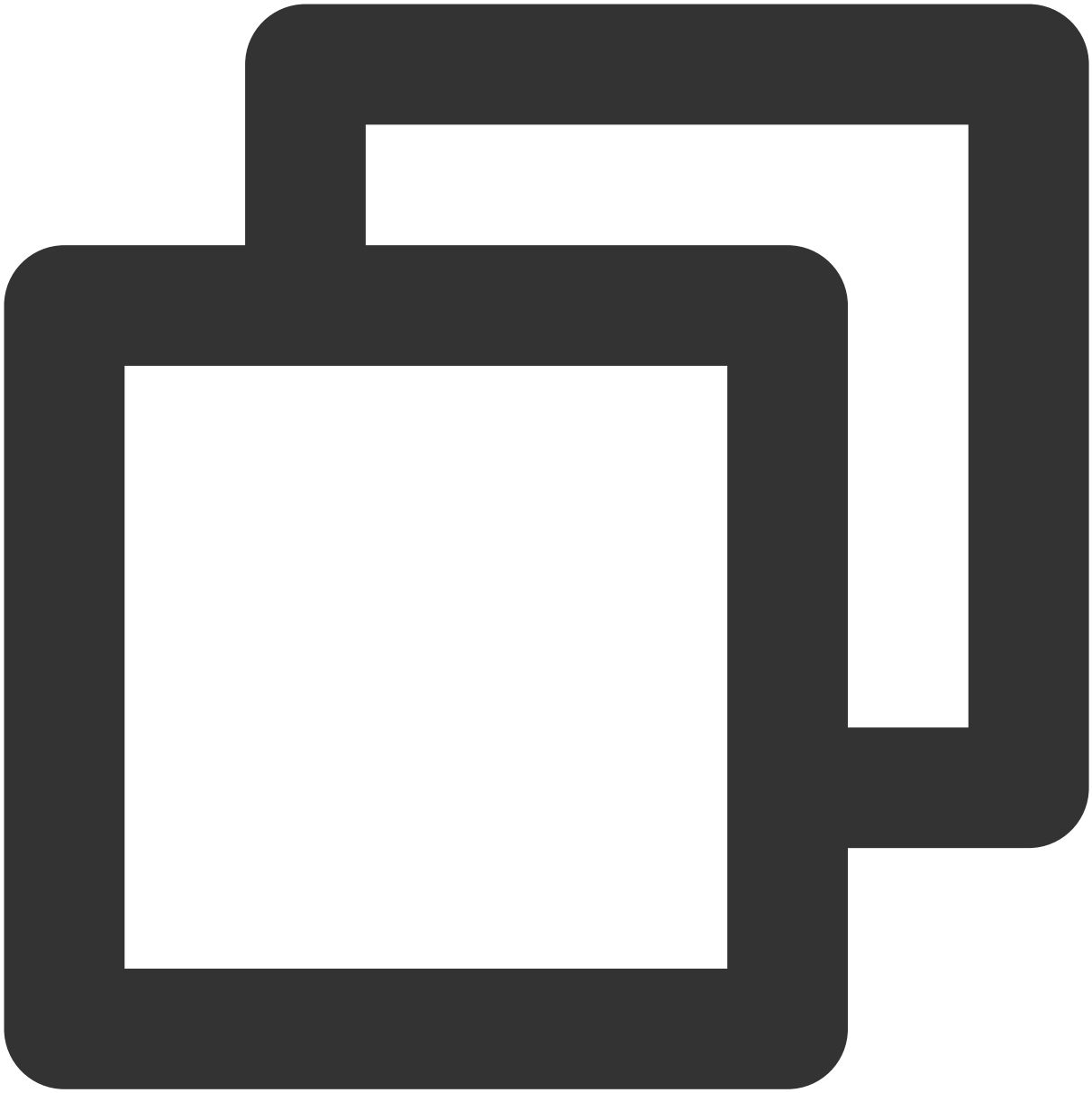


3. 在UserSig Tools中下来选择 **SDKAppID**，并输入您自己的**用户名（UserID）**，单击 `Generate` 即可获取您自己的 **UserSig**。



4. 设置进房参数 **TRTCParams** 后，调用接口 `enterRoom` 函数即可进入房间。

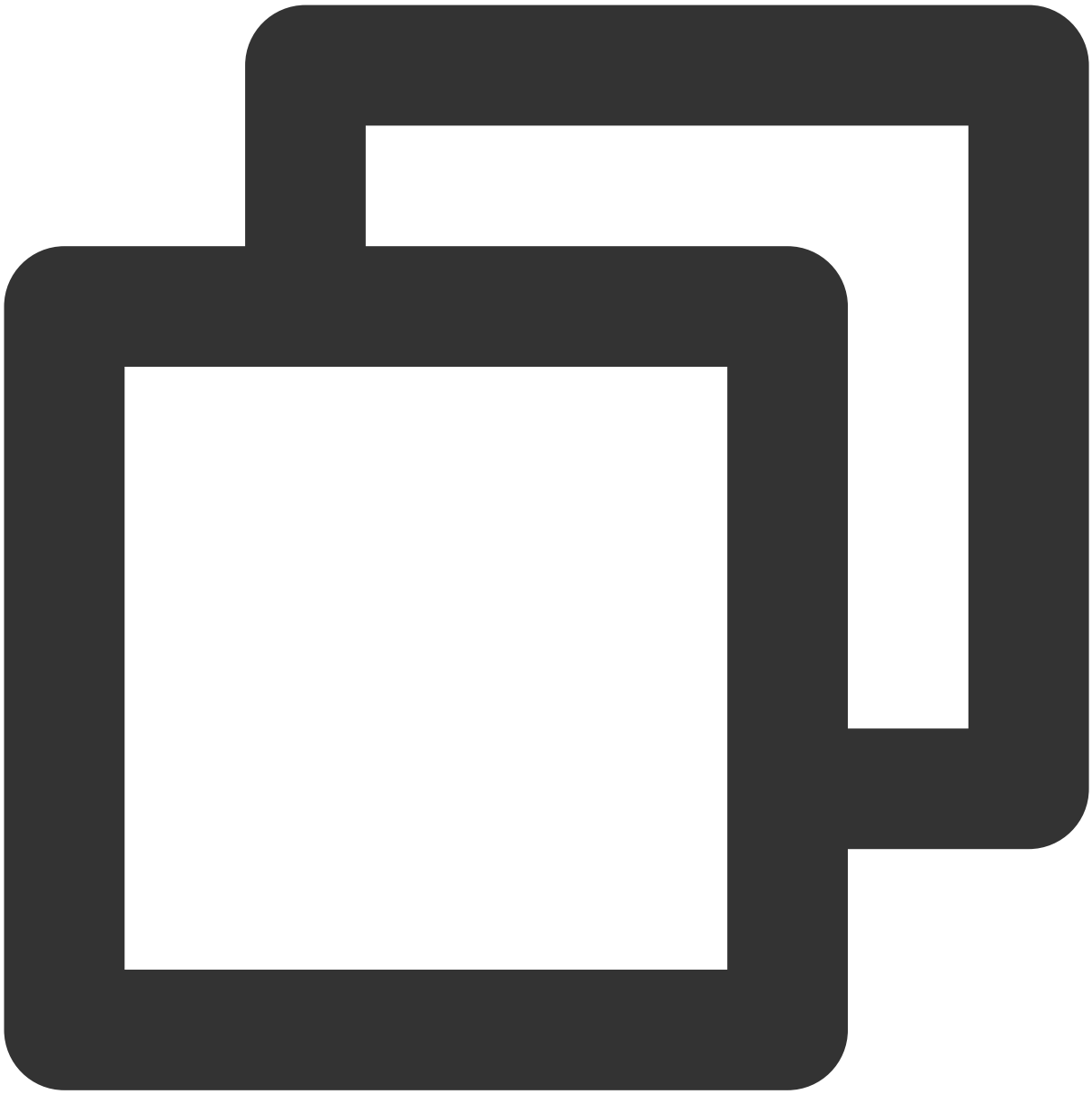
主播角色：



```
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams trtcParams = new TRTCCloudDef.TRTCParams();
trtcParams.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
trtcParams.userId = "denny";       // Please replace with your own userid
trtcParams.roomId = 123321;        // Please replace with your own room number
trtcParams.userSig = "xxx";        // Please replace with your own userSig
trtcParams.role = TRTCCloudDef.TRTCRoleAnchor;

// If your application scenario is a video call between several people, please use
mCloud.enterRoom(trtcParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

观众角色：



```
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams trtcParams = new TRTCCloudDef.TRTCParams();
trtcParams.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
trtcParams.userId = "denny";       // Please replace with your own userid
trtcParams.roomId = 123321;        // Please replace with your own room number
trtcParams.userSig = "xxx";        // Please replace with your own userSig
trtcParams.role = TRTCCloudDef.TRTCRoleAudience;

// If your application scenario is a video call between several people, please use
```

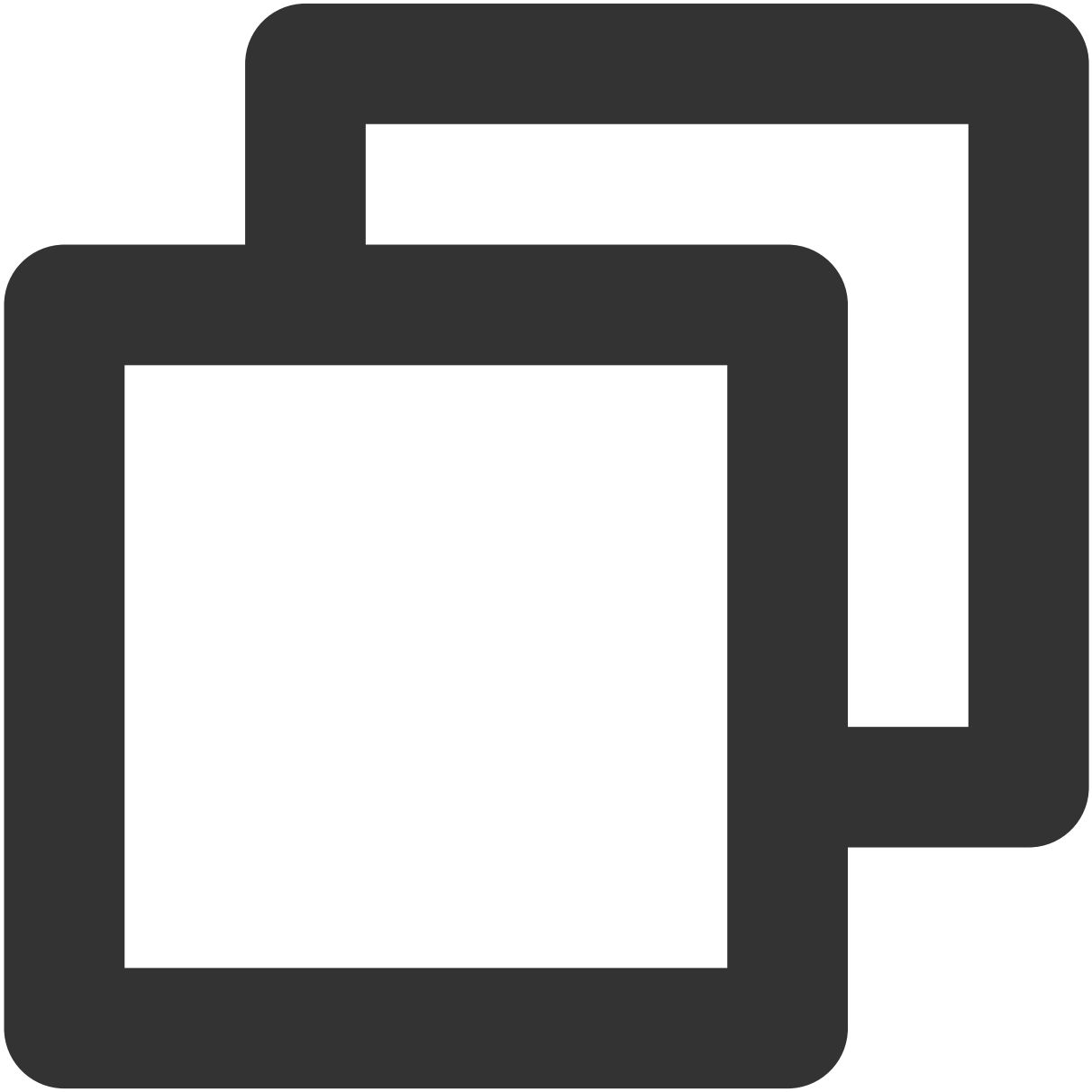
```
mCloud.enterRoom(trtcParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

注意：

如果以**观众角色**进入房间，**sdkAppId** 和 **roomId** 需要与主播端相同，而 **userId** 和 **userSig** 需要替换为自己的值。

步骤5. 打开摄像头

1. 在对应的 `.xml` 文件中添加，如以下代码所示。

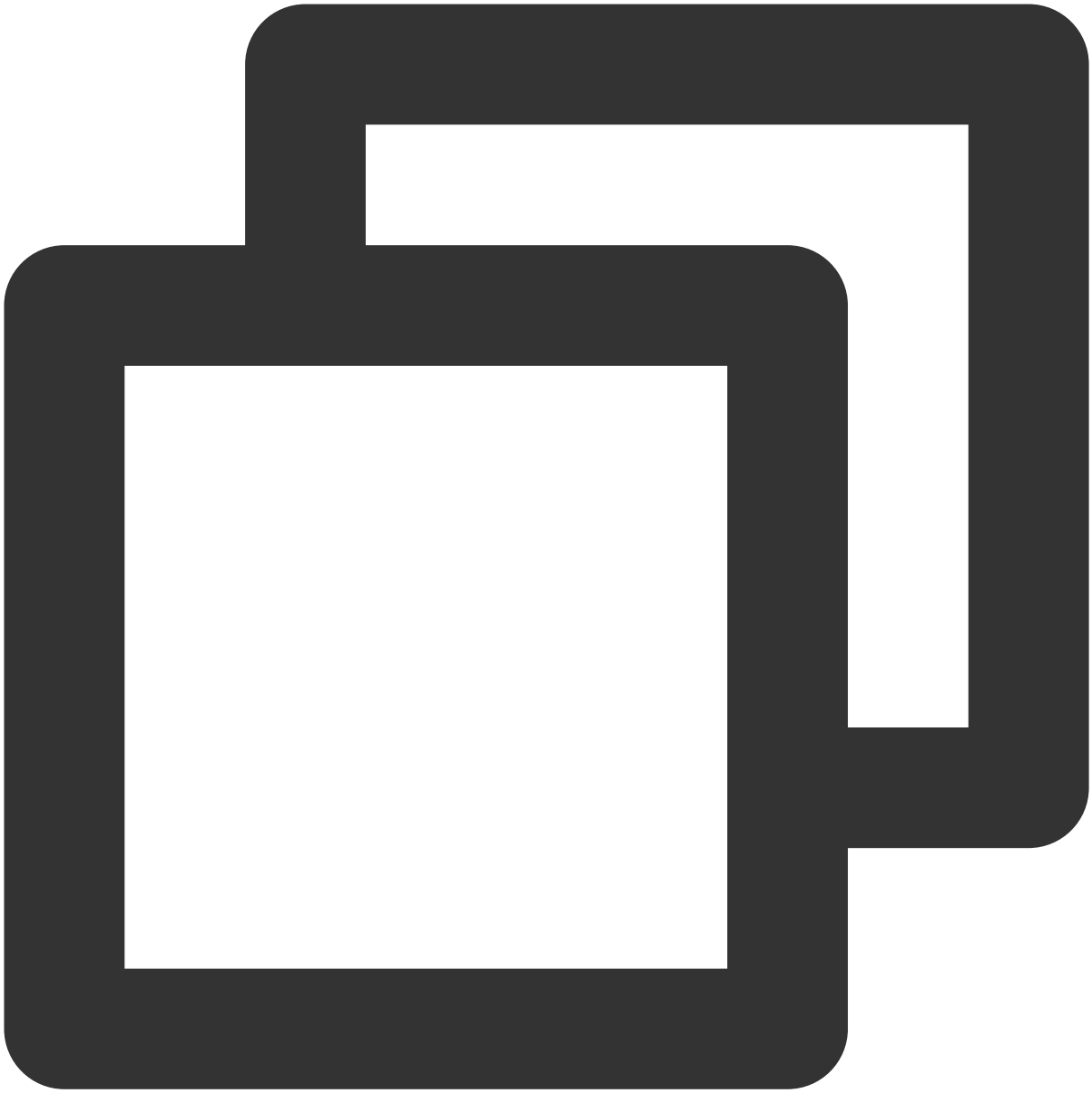


```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.an
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.tencent.rtmp.ui.TXCloudVideoView
        android:id="@+id/txcvv_main_local"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

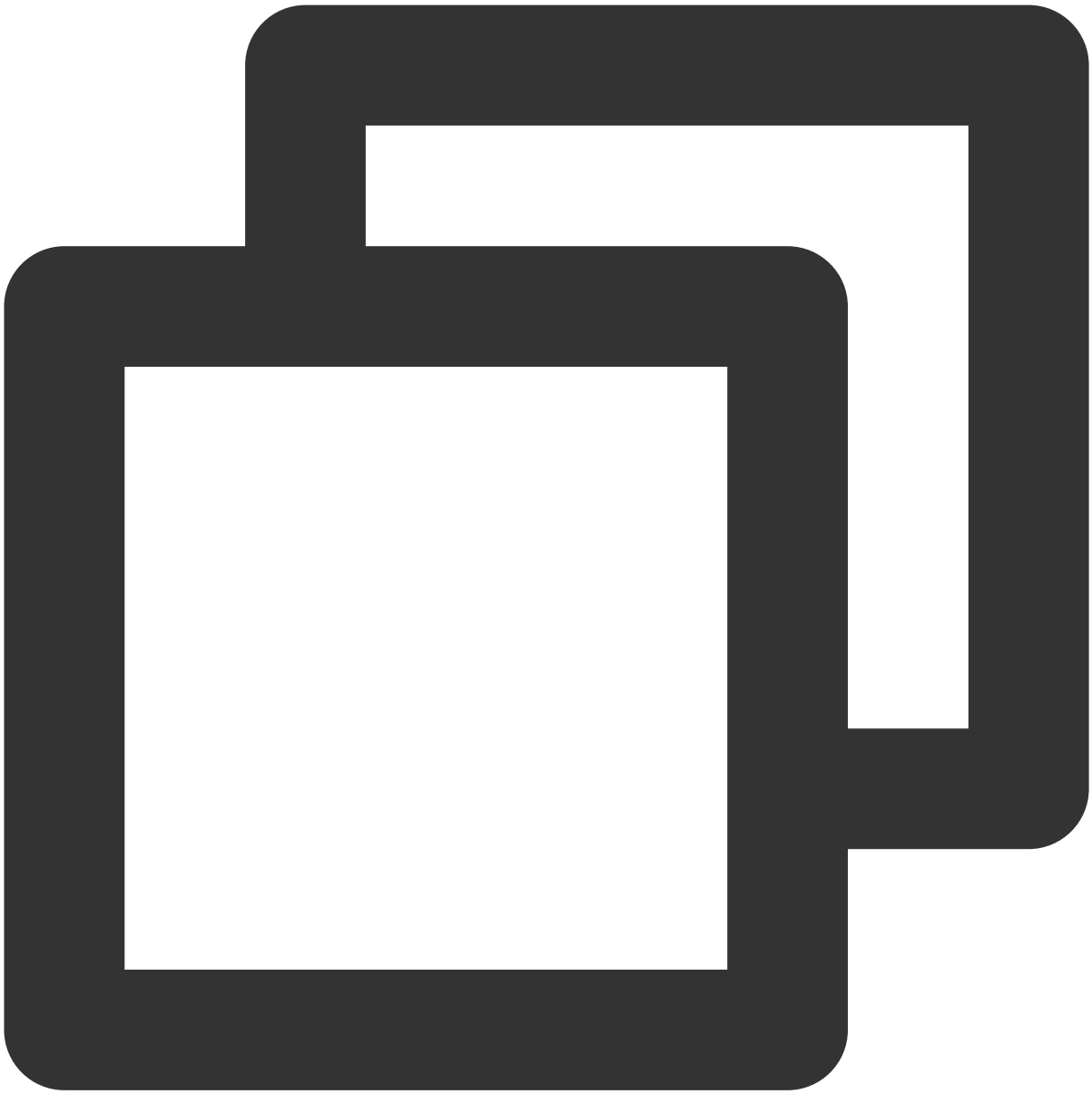
2. 在调用接口 `startLocalPreview` 打开摄像头预览之前，您可以通过通过调用接口 `setLocalRenderParams` 来设置本地预览的渲染参数。



```
// Set the preview mode of the local screen
TRTCCloudDef.TRTCRenderParams trtcRenderParams = new TRTCCloudDef.TRTCRenderParams(
trtcRenderParams.fillMode   = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;
trtcRenderParams.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;
mCloud.setLocalRenderParams(trtcRenderParams);

// Start a preview of the local camera
TXCloudVideoView cameraVideo = findViewById(R.id.txcvv_main_local);
mCloud.startLocalPreview(true, cameraVideo);
```

3. 您可以通过调用接口 `TXDeviceManager` 来完成**“切换前后摄像头”**、**“设置对焦模式”**、**“打开”**等操作。



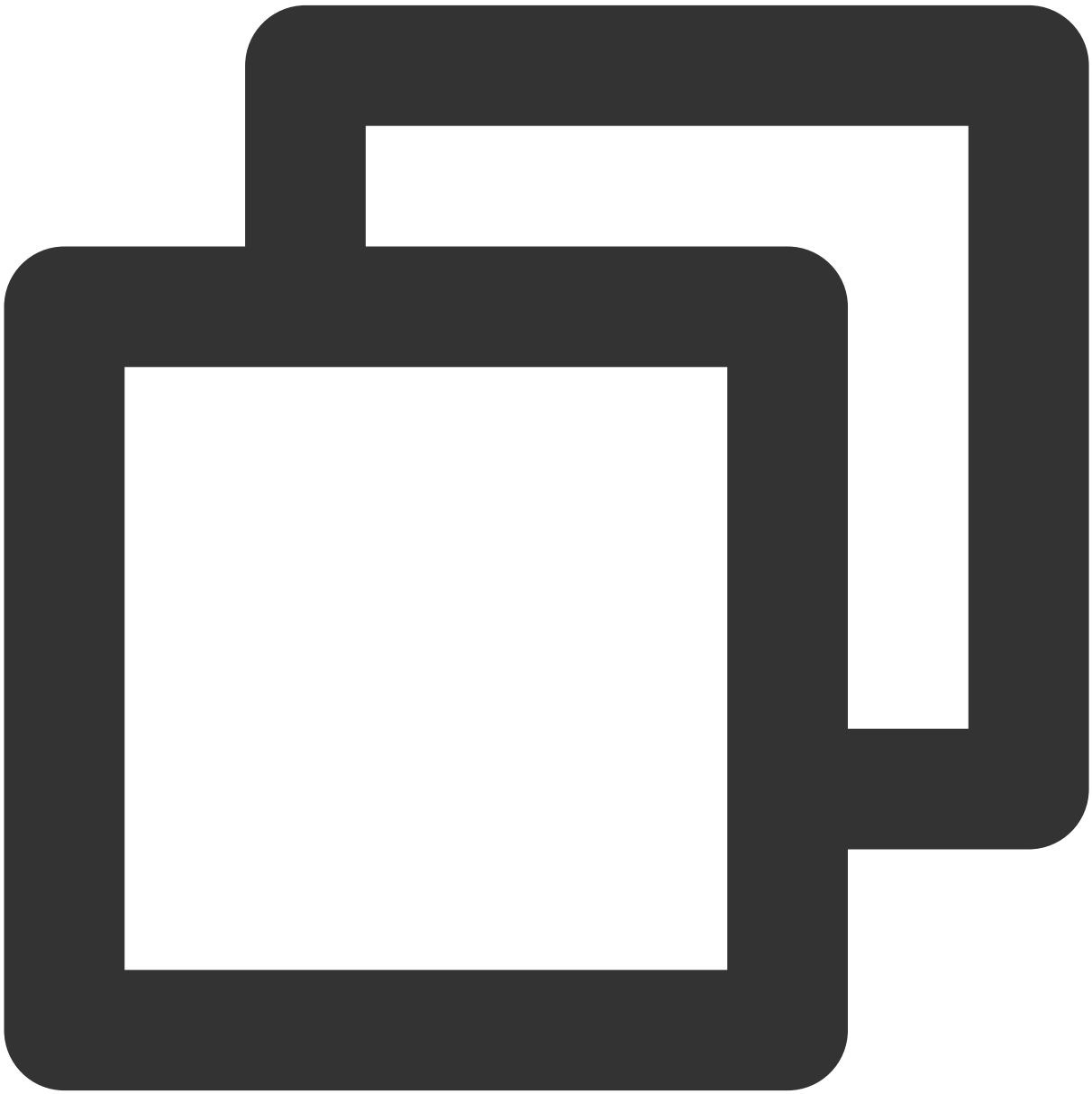
```
// Enable auto focus and turn on the flash using the TXDeviceManager
TXDeviceManager manager = mCloud.getDeviceManager();
if (manager.isAutoFocusEnabled()) {
    manager.enableCameraAutoFocus(true);
}
manager.enableCameraTorch(true);
```

说明：

摄像头默认打开前置摄像头，如果您需要使用后置摄像头，调用 `manager.switchCamera(false)` 即可打开后置摄像头。

步骤6. 打开麦克风

您可以调用 `startLocalAudio` 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式，建议根据您的需求**选择以下其中一个适合您项目的模式**。



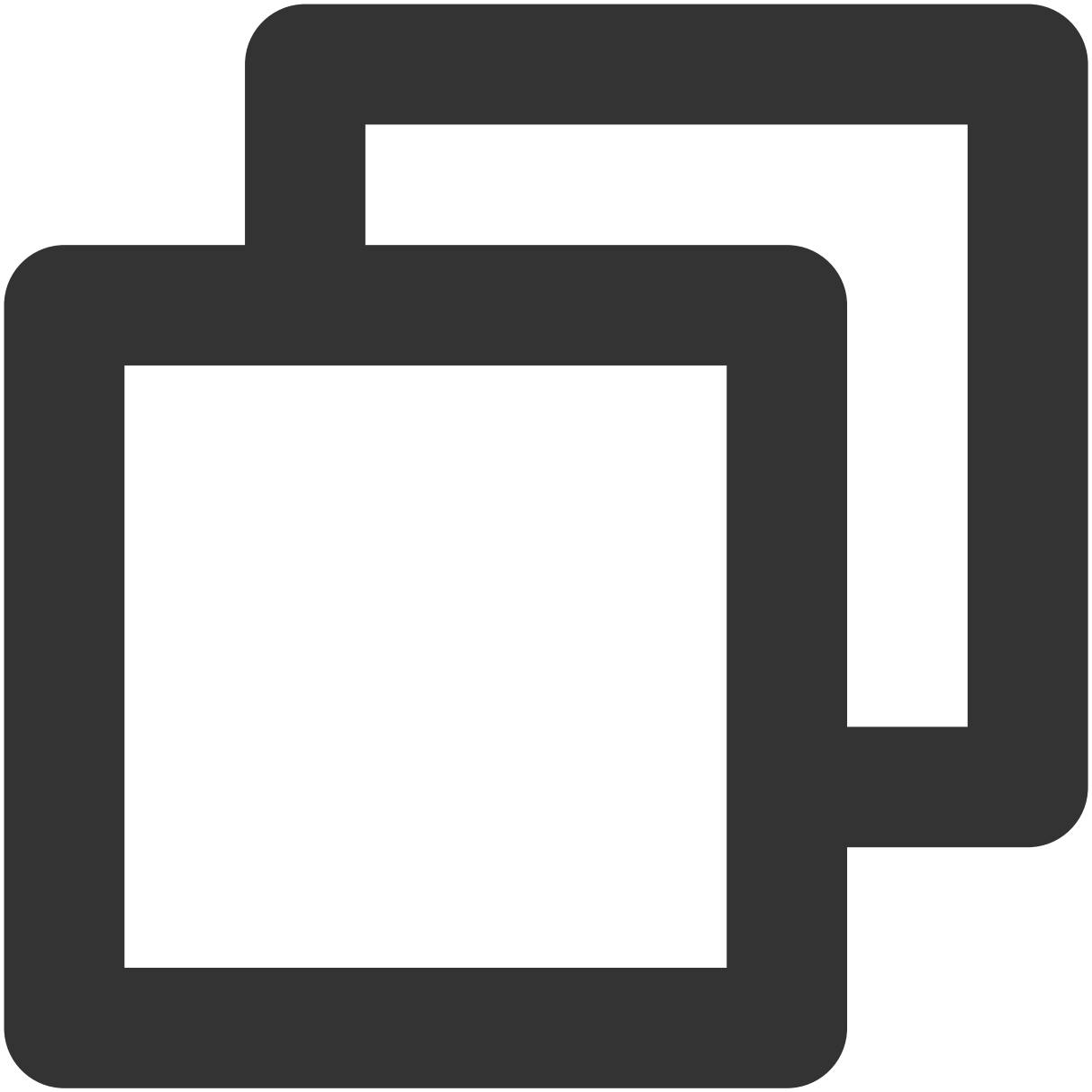
```
// Enable microphone acquisition and set the current scene to: Voice mode
```

```
// For high noise suppression capability, strong and weak network resistance
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH );

// Enable microphone acquisition, and set the current scene to: Music mode (
// For high fidelity acquisition, low sound quality loss, recommended to use with p
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
```

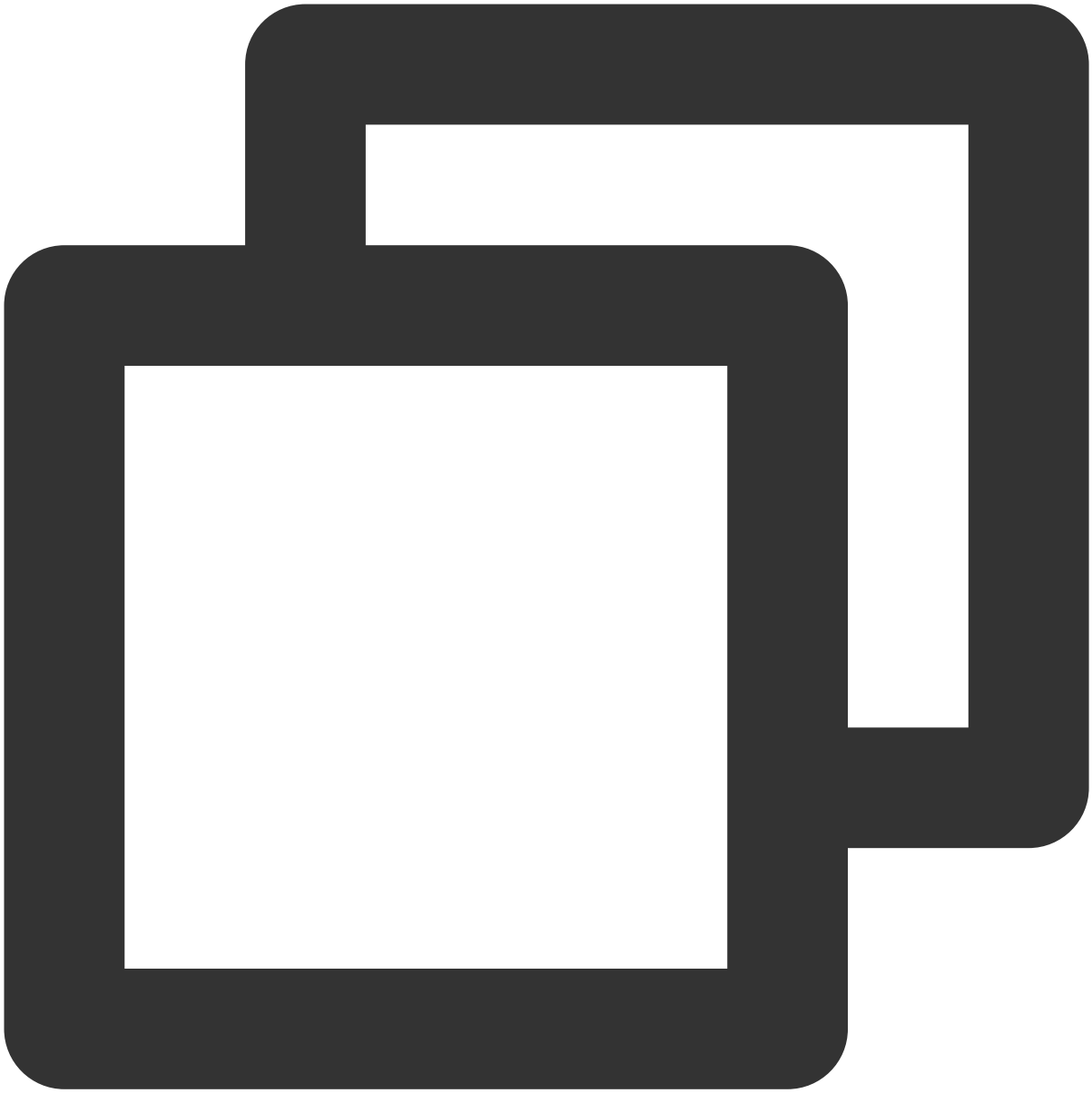
步骤7. 播放/停止视频流

创建一个新项目，按照**步骤1-4**以**观众角色**进入 **denny** 的房间后，您可以通过调用接口 `startRemoteView` 来播放远端用户的视频画面。



```
// Play denny's camera footage
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraView)
```

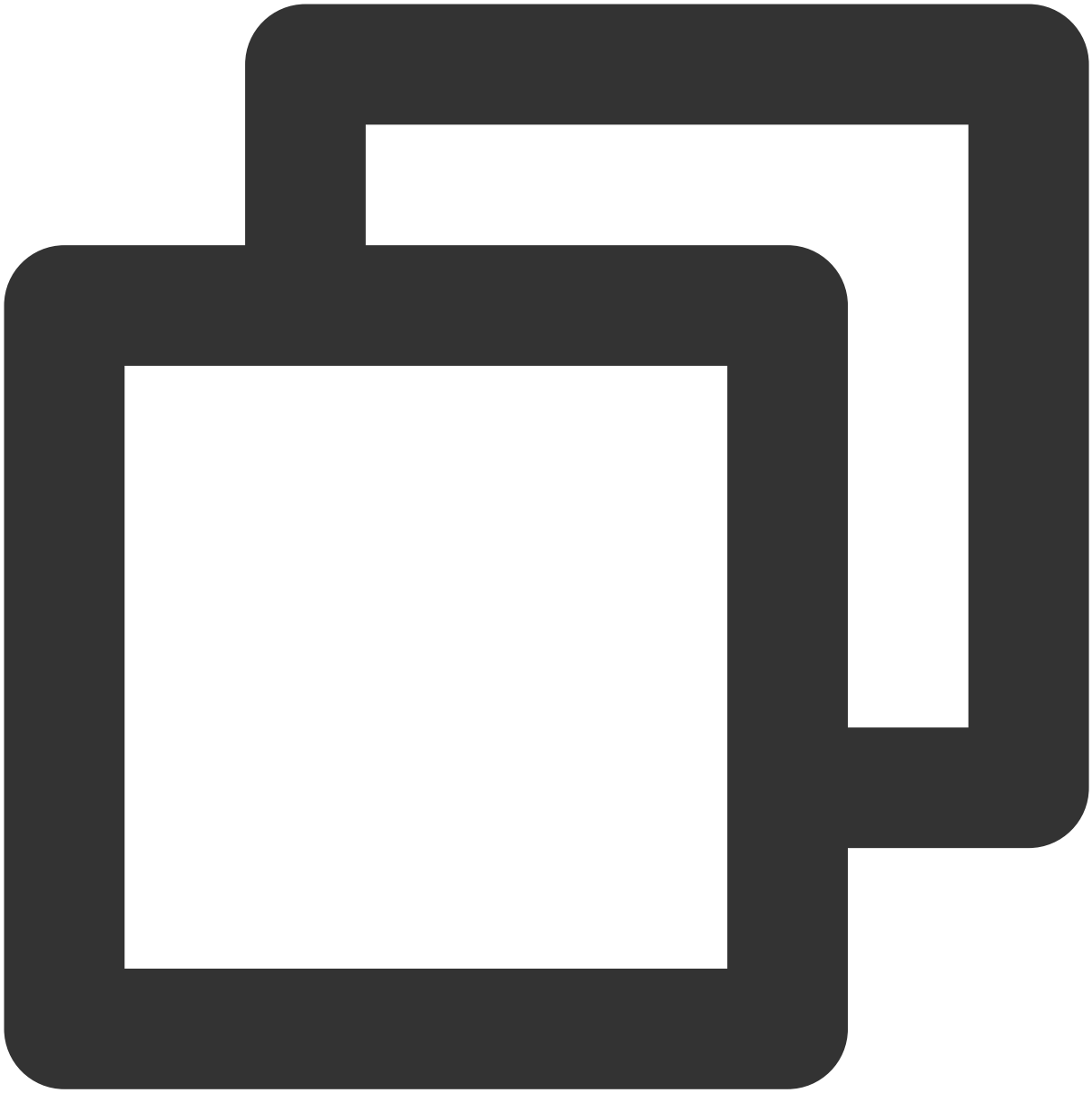
之后，您通过调用接口 `stopRemoteView` 停止某个远端用户的视频，也可以通过接口 `stopAllRemoteView` 停止播放所有远端用户的视频。



```
// Stop denny's camera footage
mCloud.stopRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraView);
// Stop all camera footages
mCloud.stopAllRemoteView();
```

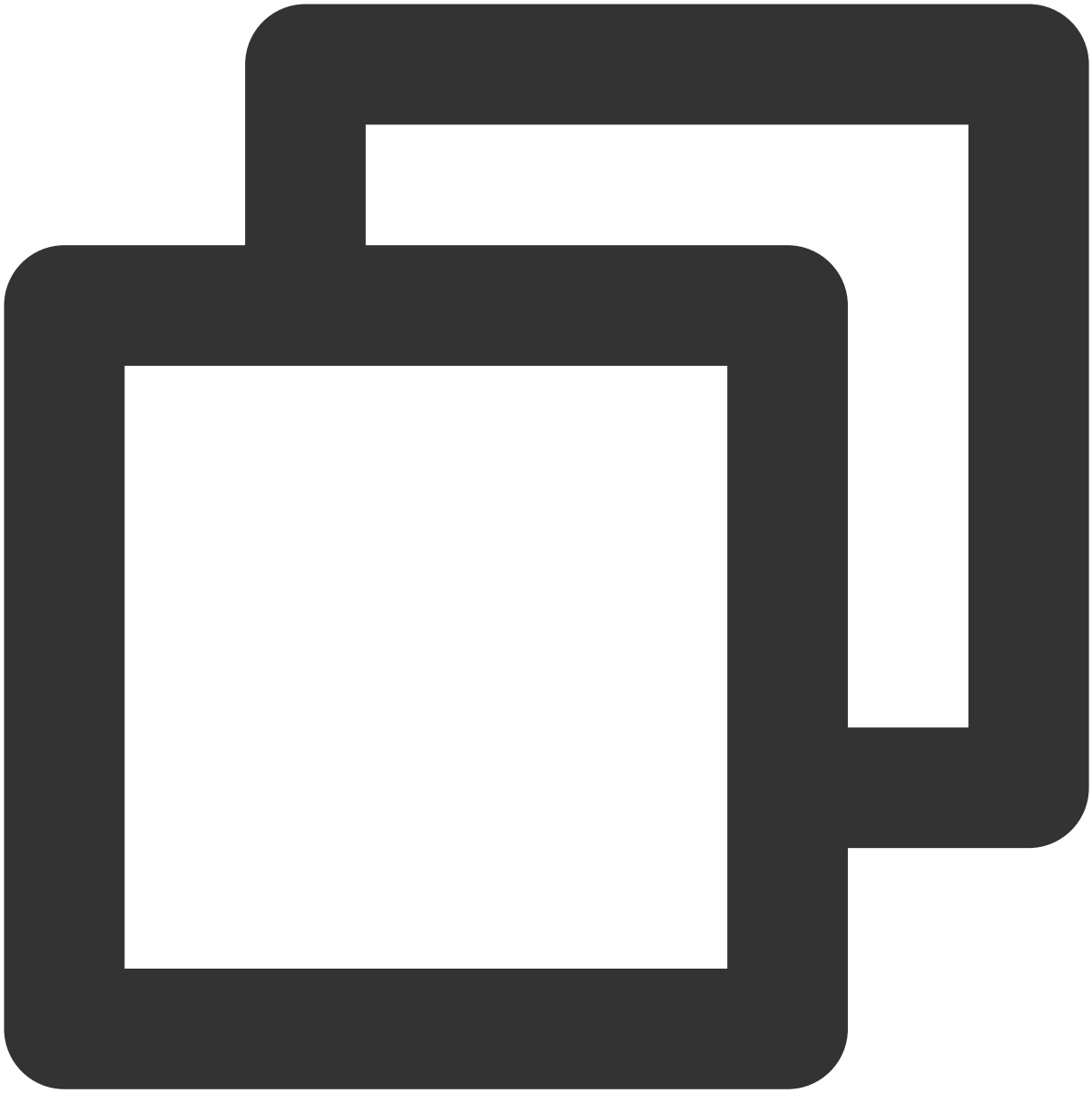
步骤8. 播放/停止音频流

您可以通过调用接口 `muteRemoteAudio("denny", true)` 来静音远端用户 **denny** 的声音。



```
// Mute user with id denny
mCloud.muteRemoteAudio("denny", true);
```

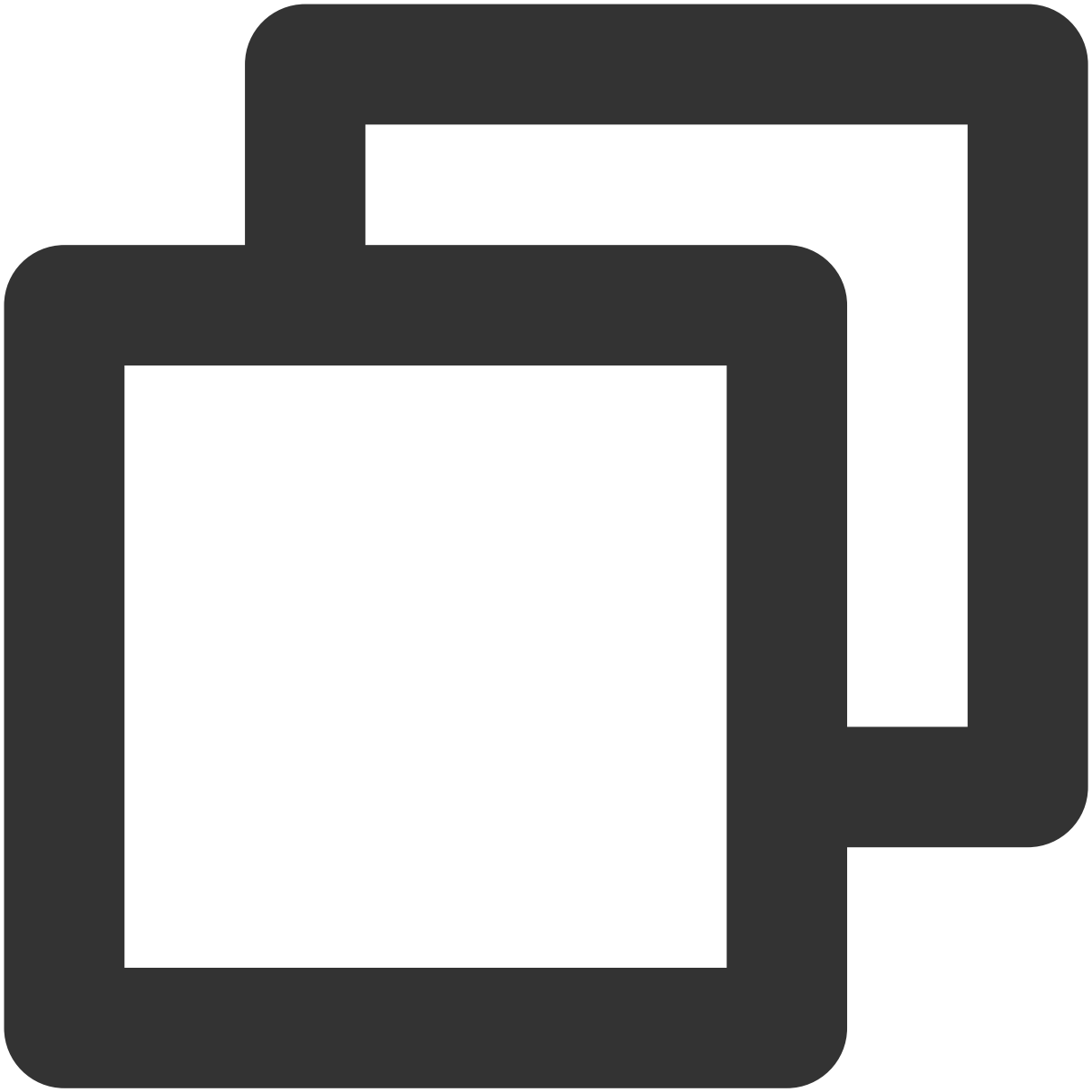
之后也可以通过调用接口 `muteRemoteAudio("denny", false)` 来解除对他的静音。



```
// Unmute user with id denny
mCloud.muteRemoteAudio("denny", false);
```

步骤9. 退出房间

您可以调用接口 `exitRoom` 即可退出当前的房间，SDK 会在退房结束后通过 `onExitRoom(int reason)` 回调事件通知您。



```
// Exit current room
mCloud.exitRoom();

// Listen for the `onExitRoom` callback to get the reason for room exit
@Override
public void onExitRoom(int reason) {
    if (reason == 0) {
        Log.d(TAG, "Exit current room by calling the 'exitRoom' api of sdk ...");
    } else if (reason == 1) {
        Log.d(TAG, "Kicked out of the current room by server through the restful ap
    } else if (reason == 2) {
```

```
            Log.d(TAG, "Current room is dissolved by server through the restful api..."
        }
    }
```

常见问题

您可以在 API 参考 查看所有函数列表及其描述。
如果您的接入和使用中遇到问题，请参见 常见问题。

联系我们

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com。

Mac

最近更新时间：2024-07-18 15:20:46

本文将介绍如何快速完成 Mac RTC Engine 的接入，实现一个基本的音视频通话。

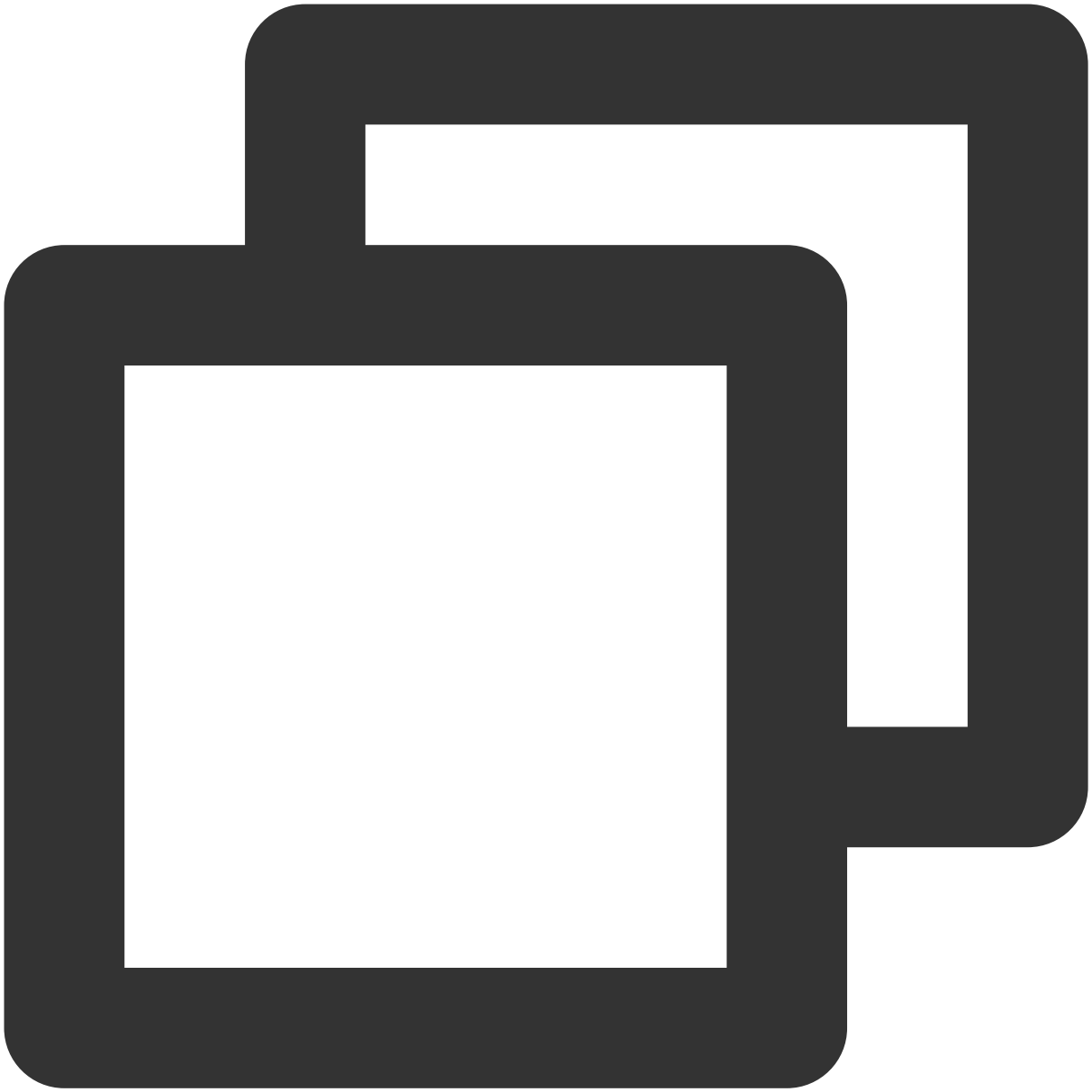
环境准备

Xcode 9.0+。
OS X10.10+ 的 Mac 真机。
项目已配置有效的开发者签名。

步骤1. 导入 SDK

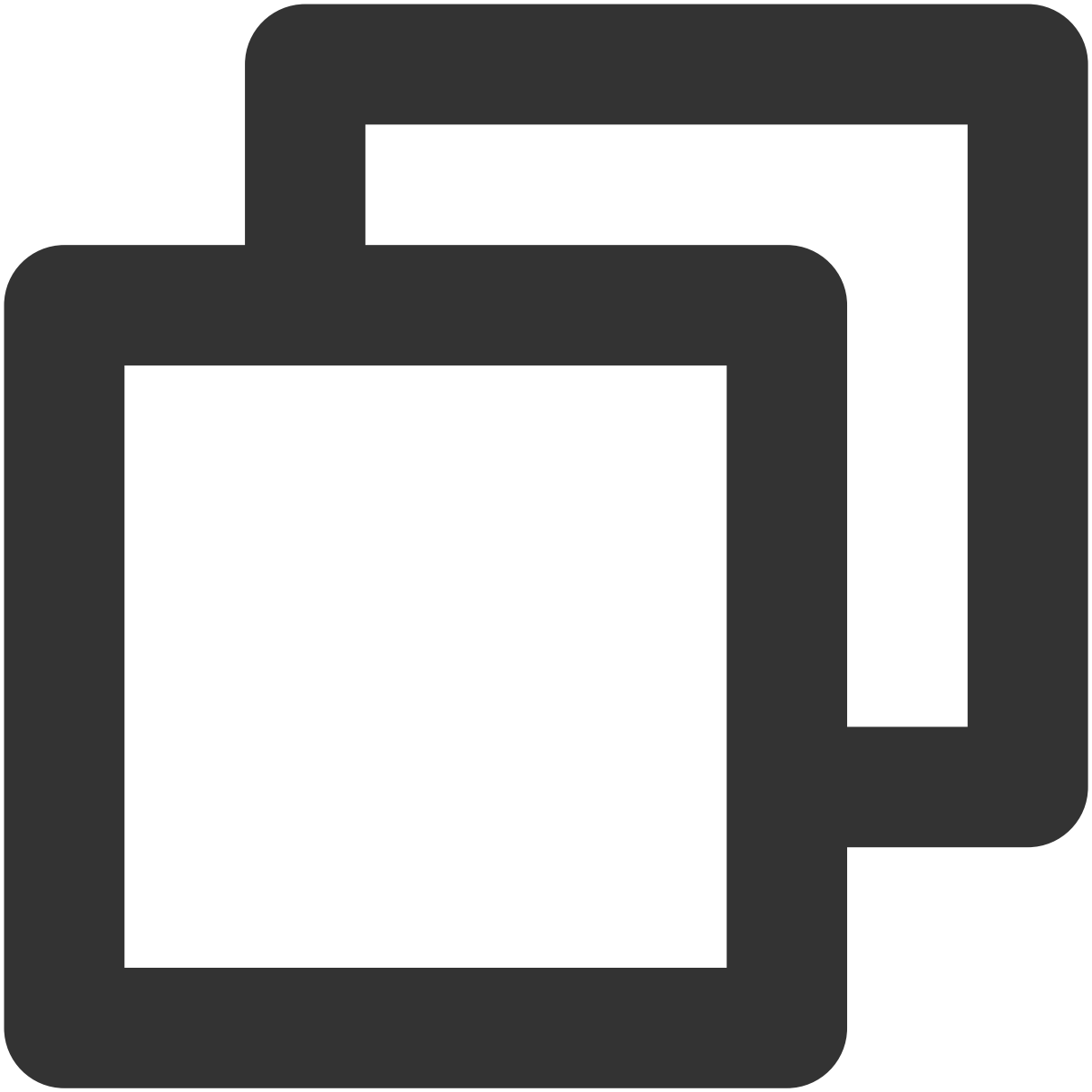
1. 在终端窗口输入以下命令安装 CocoaPods，如果您已经完成 CocoaPods 的安装，可以跳过该步。

版权所有：腾讯云计算（北京）有限责任公司



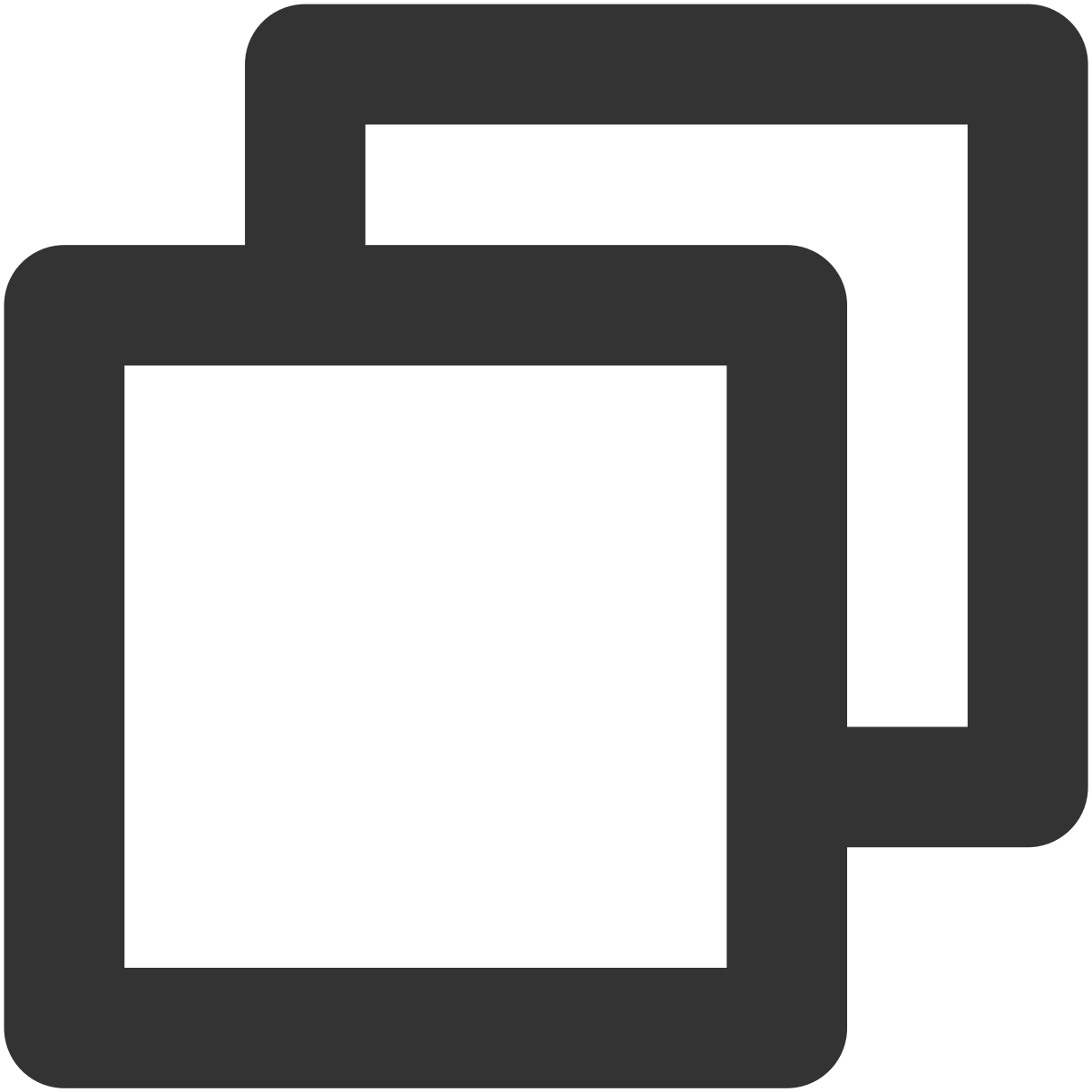
```
sudo gem install cocoapods
```

2. 在终端窗口中进入 **TRTCDemo** 所在的路径，输入以下命令为您的项目创建 **Podfile** 文件。



```
pod init
```

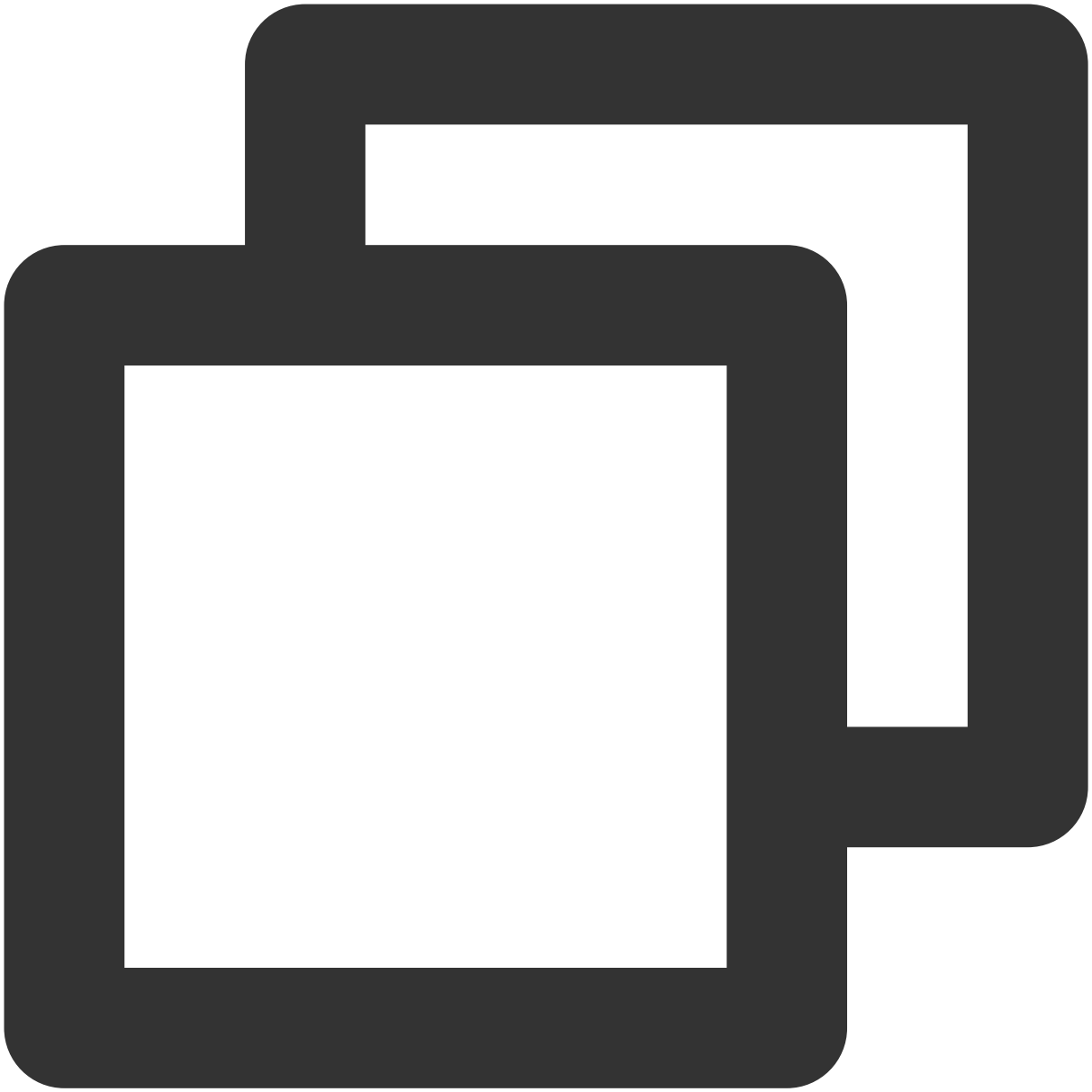
3. 编辑 Podfile 文件为如下内容，将 **Your Target** 修改为您自己的项目名称。



```
platform :osx, '10.10'

target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd
end
```

4. 在终端窗口中输入以下命令更新本地库文件，并安装 SDK 。



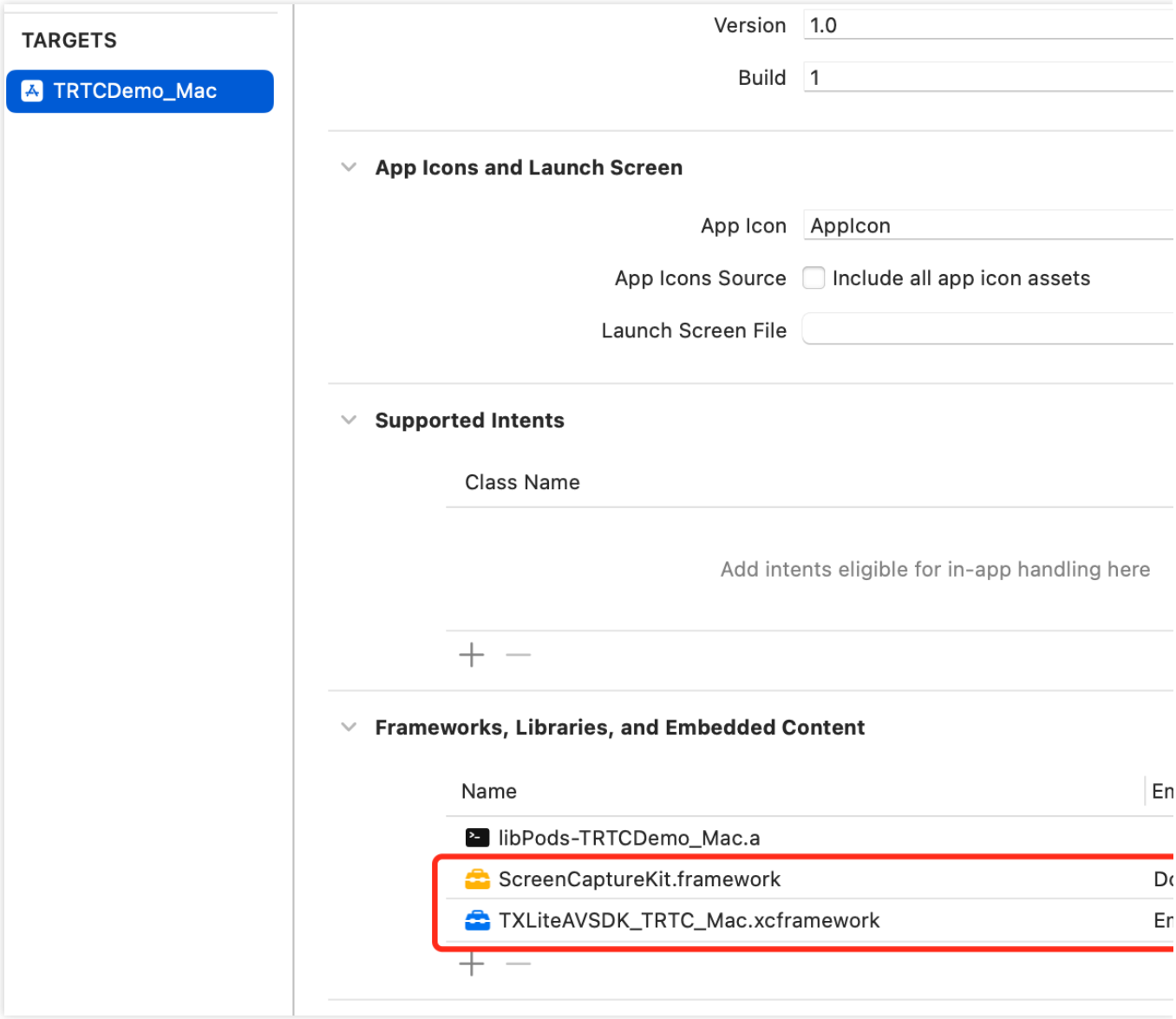
```
pod install
```

说明：

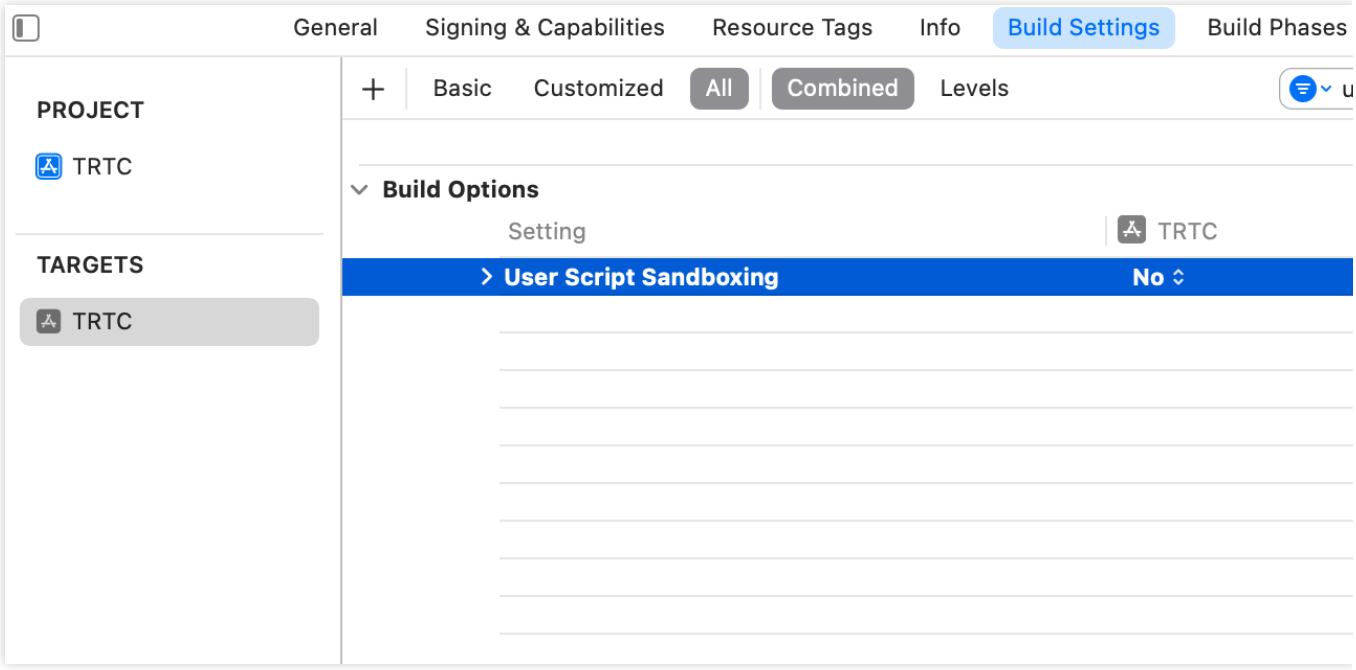
pod 命令执行完后，会生成集成了 SDK 的 **.xcworkspace** 后缀的工程文件，双击打开**.xcworkspace** 文件进行后续工作。

步骤2. 配置项目

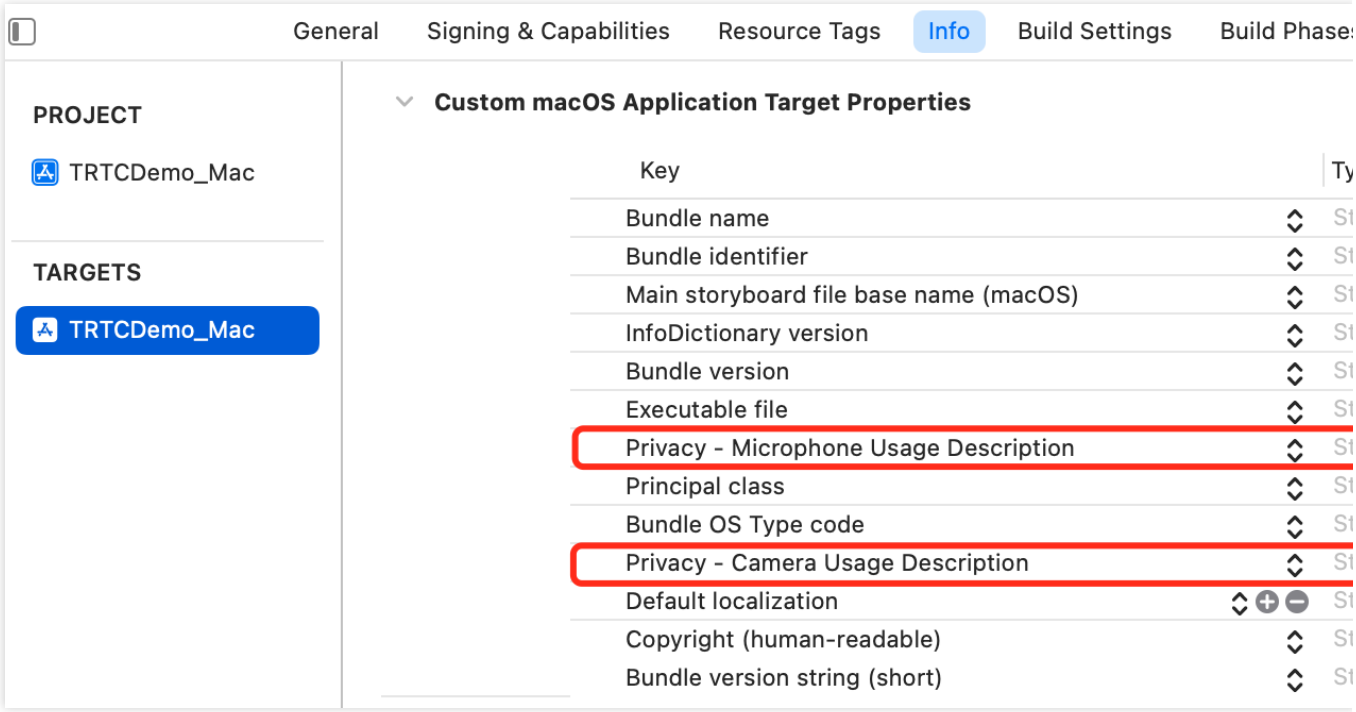
1. 打开 `.xcworkspace` 文件后，在 `General` 中的 **Frameworks, Libraries, and Embedded Content** 部分添加 **TXLiteAVSDK_TRTC.xcframework** 和 **ScreenCaptureKit.framework** 。



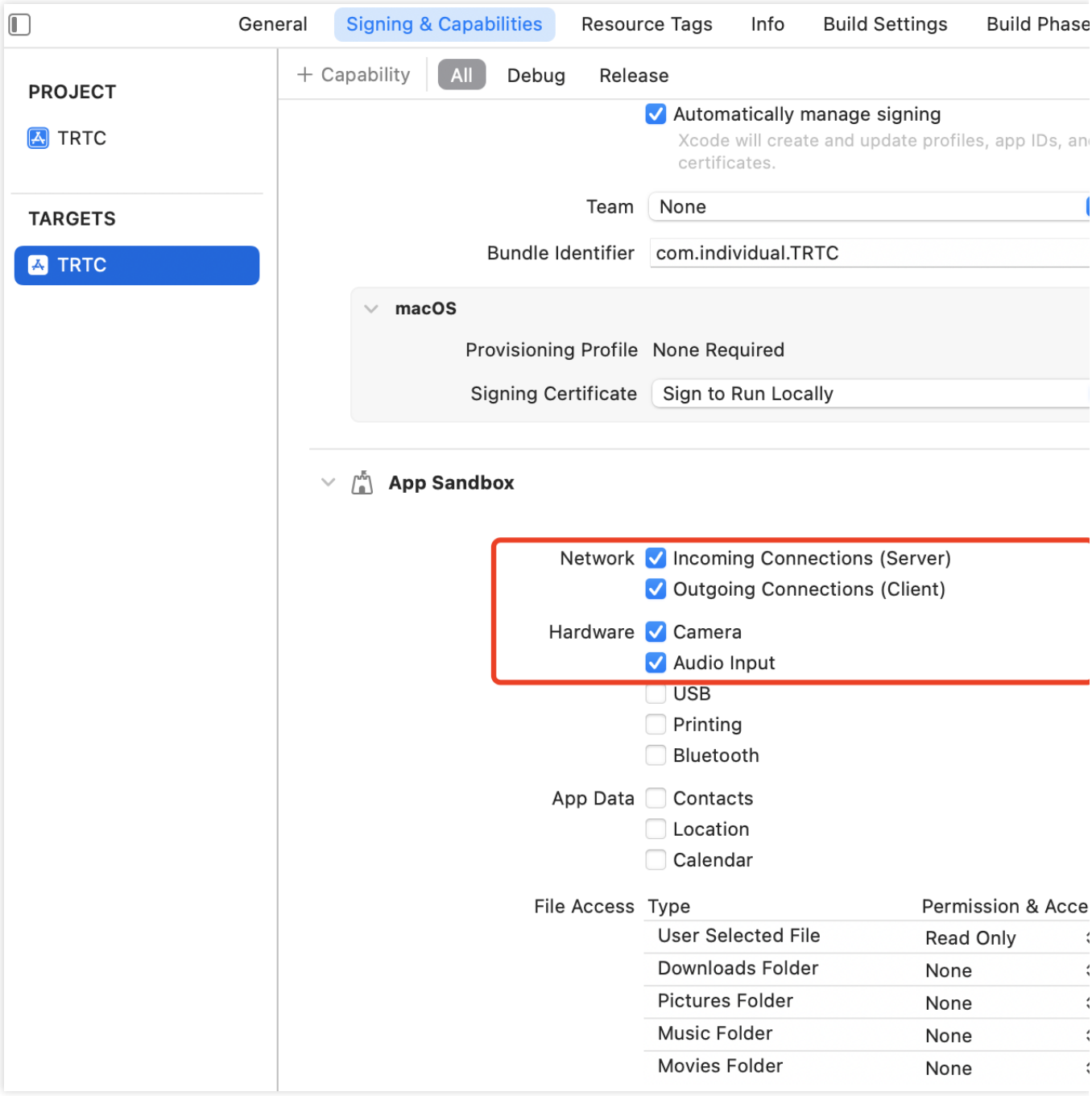
2. 在 `Build Settings` 中搜索 **User Script Sandboxing**，将其值置为 `No` 。



3. 在 `Info.plist` 中添加 **Privacy-Microphone Usage Description** 和 **Privacy-Microphone Usage Description**，并填入 **Microphone/Camera** 使用的目标提示语，获取麦克风和摄像头的使用权限。

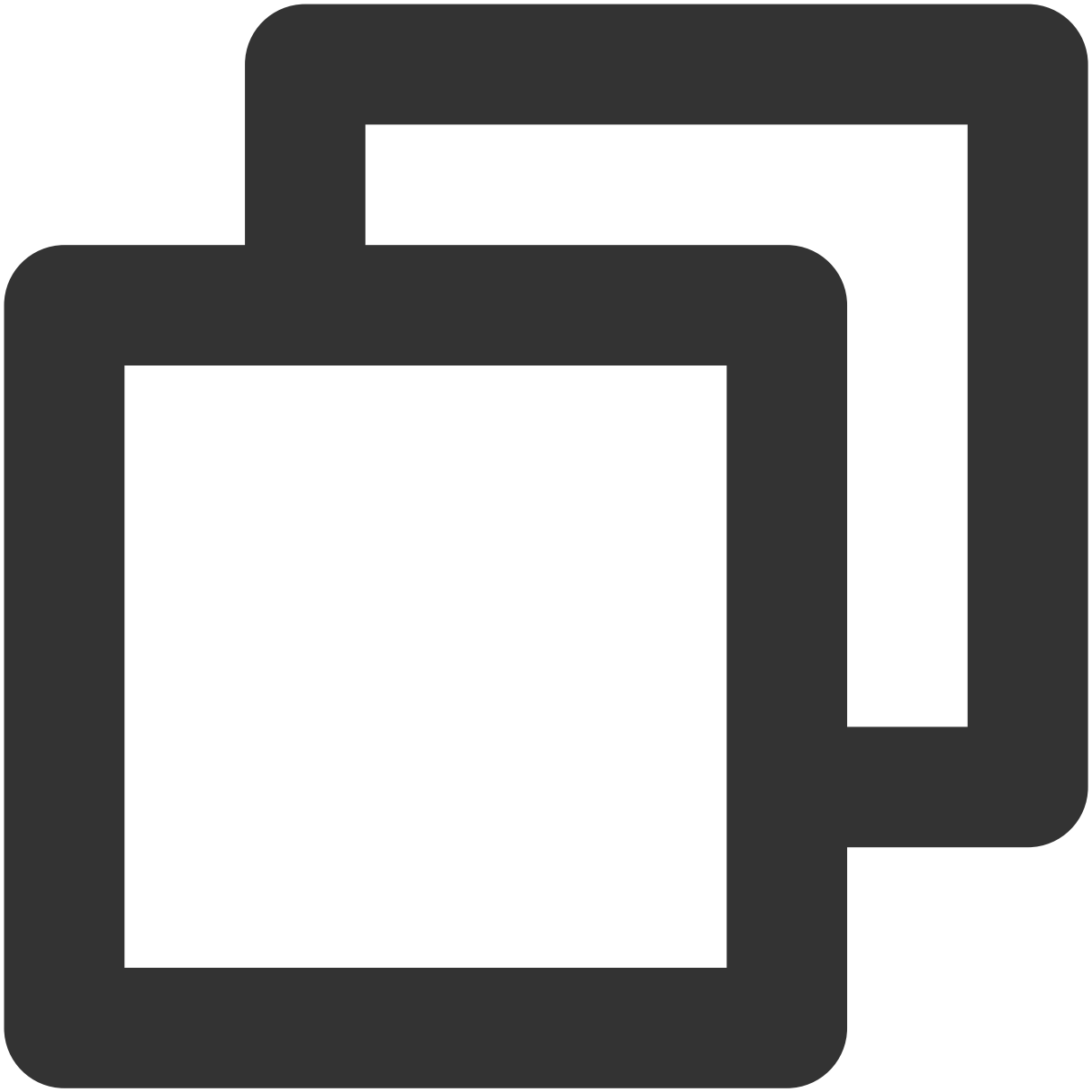


4. 在 `Signing & Capabilities` 中 **App Sandbox** 部分勾选以下内容。



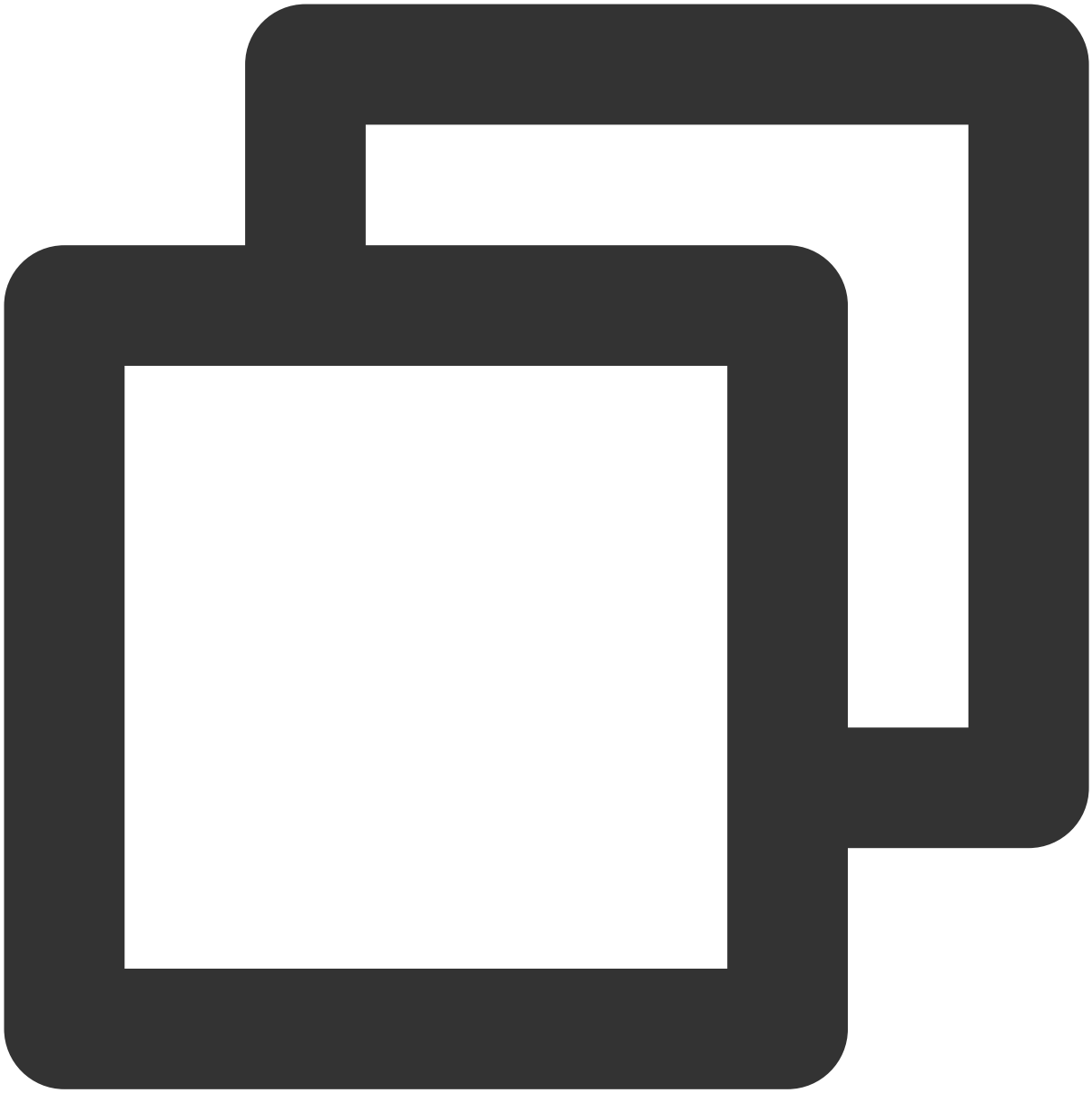
步骤3. 创建 TRTC 实例

1. 在 `AppDelegate.h` 文件中添加对 SDK 的模块引用：



```
@import TXLiteAVSDK_TRTC_Mac;
```

2. 在 `AppDelegate.h` 文件中添加以下属性，并声明 `toastTip:` 方法。



```
#import <Cocoa/Cocoa.h>
@import TXLiteAVSDK_TRTC_Mac;

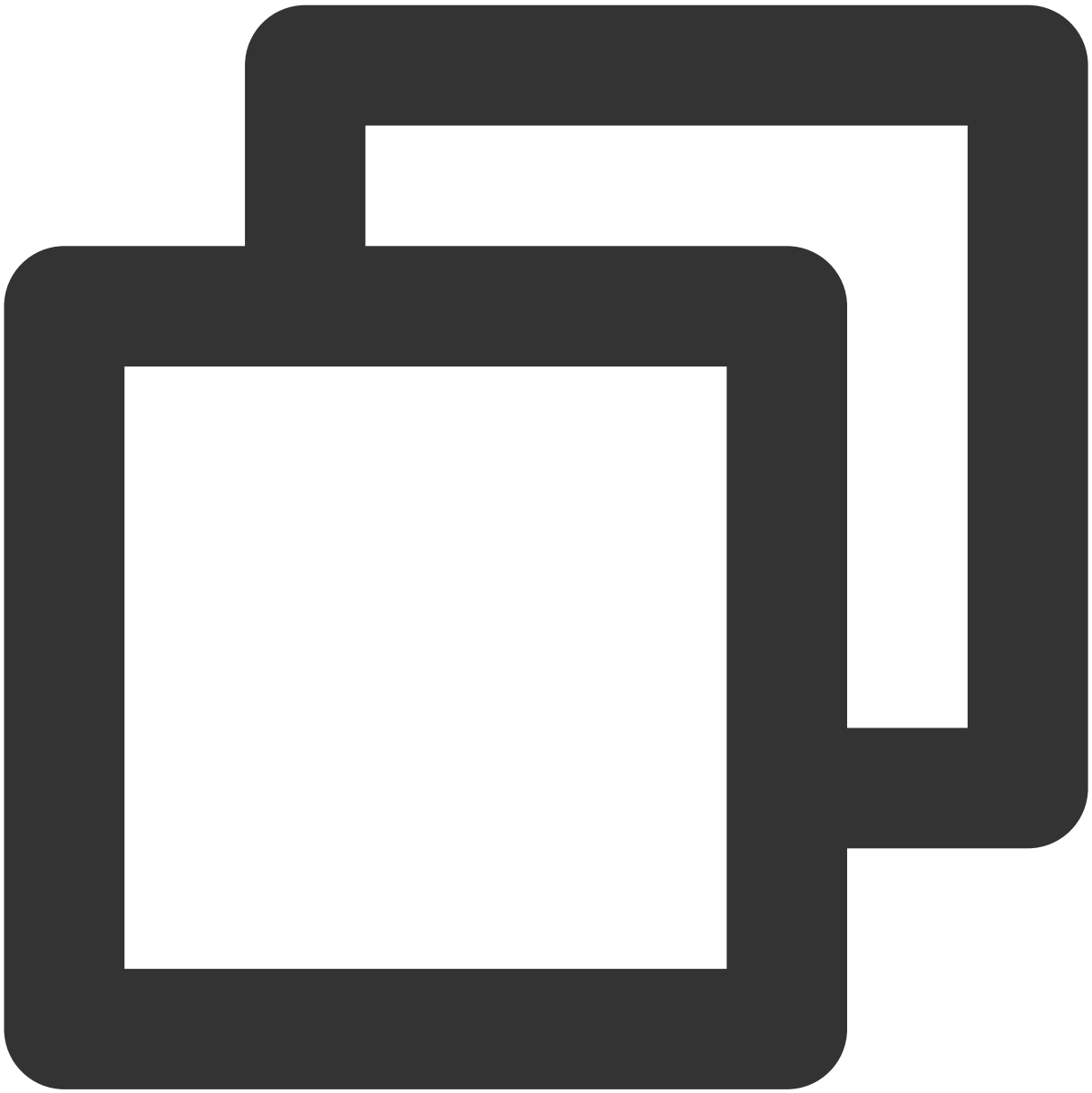
@interface AppDelegate : NSObject <NSApplicationDelegate>

@property (nonatomic, strong) NSWindow *window; // Add window property
@property (nonatomic, strong) TRTCCloud *trtcCloud; // Add trtcCloud property
@property (nonatomic, strong) NSView *localCameraVideoView; // Add localCameraVideo

- (void)toastTip:(NSString *)tip; // Declare the toastTip: method
```

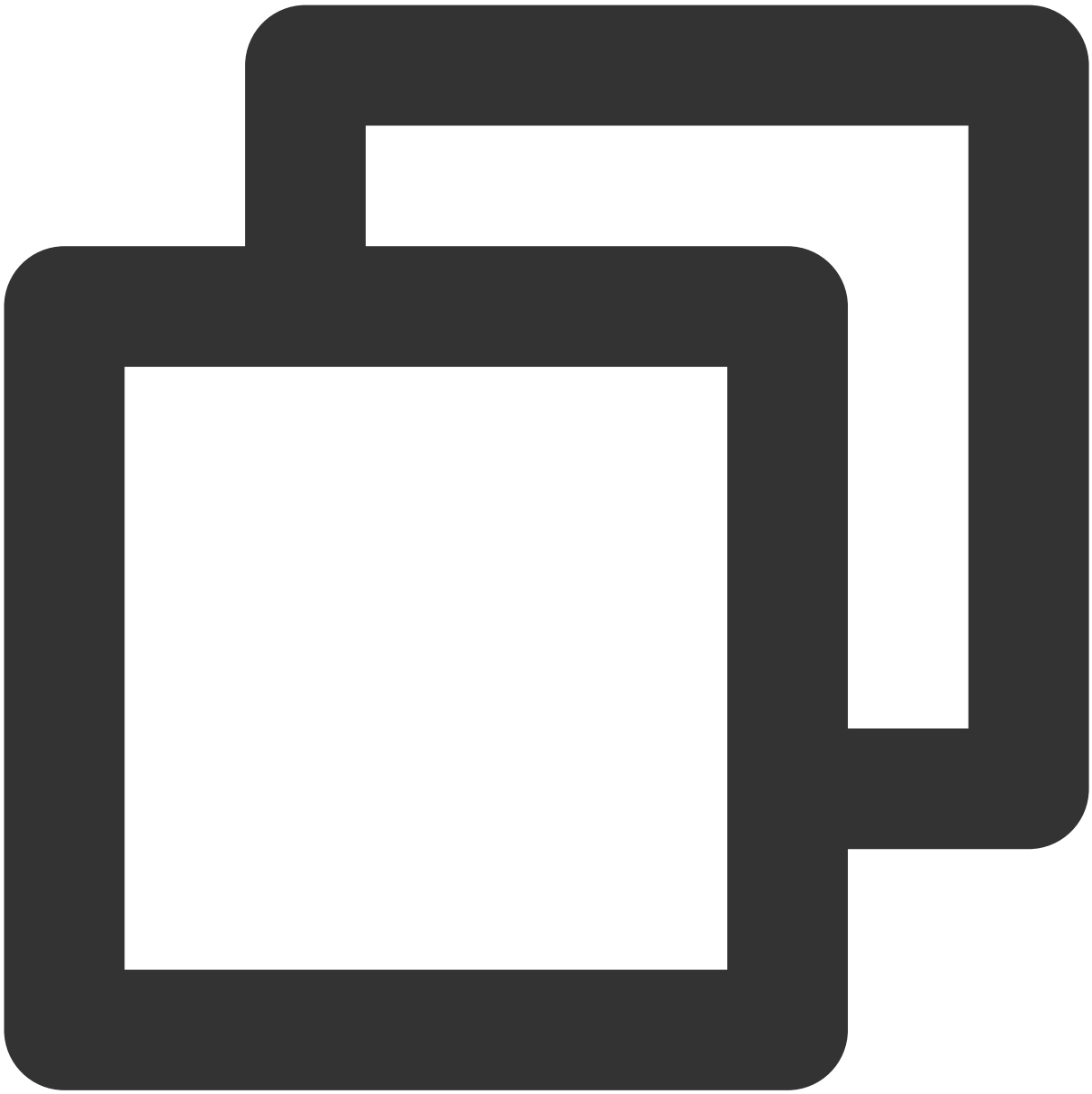
```
@end
```

3. 在 `AppDelegate.m` 文件中实现 `toastTip:` 方法。



```
// Implement toastTip: method
- (void)toastTip:(NSString *)tip {
    NSAlert *alert = [[NSAlert alloc] init];
    [alert setMessageText:tip];
    [alert runModal];
}
```

4. 在 `applicationDidFinishLaunching()` 方法中调用初始化接口创建 TRTC 实例，并设置事件回调接口。



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    // Override point for customization after application launch.

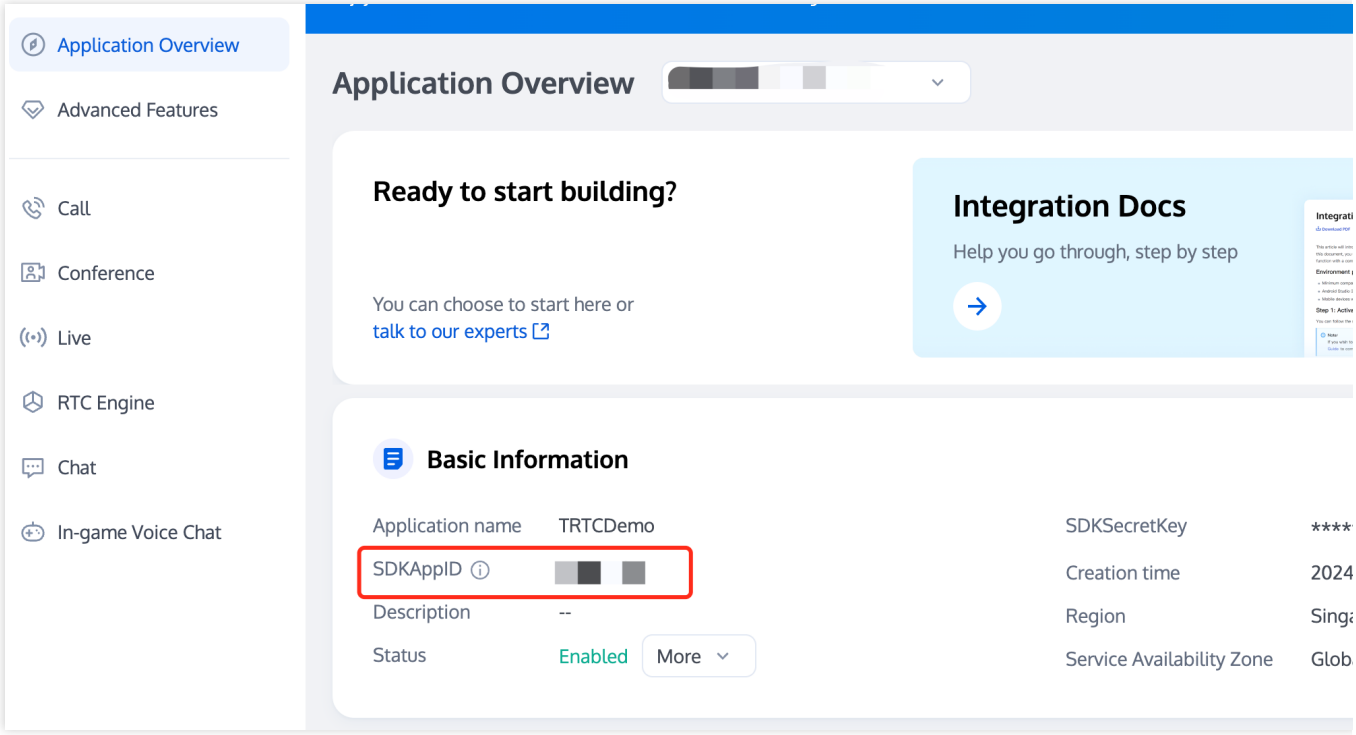
    // Create trtc instance(singleton)  and set up event listeners
    _trtcCloud = [TRTCCloud sharedInstance];
    _trtcCloud.delegate = self;

    return YES;
}
```

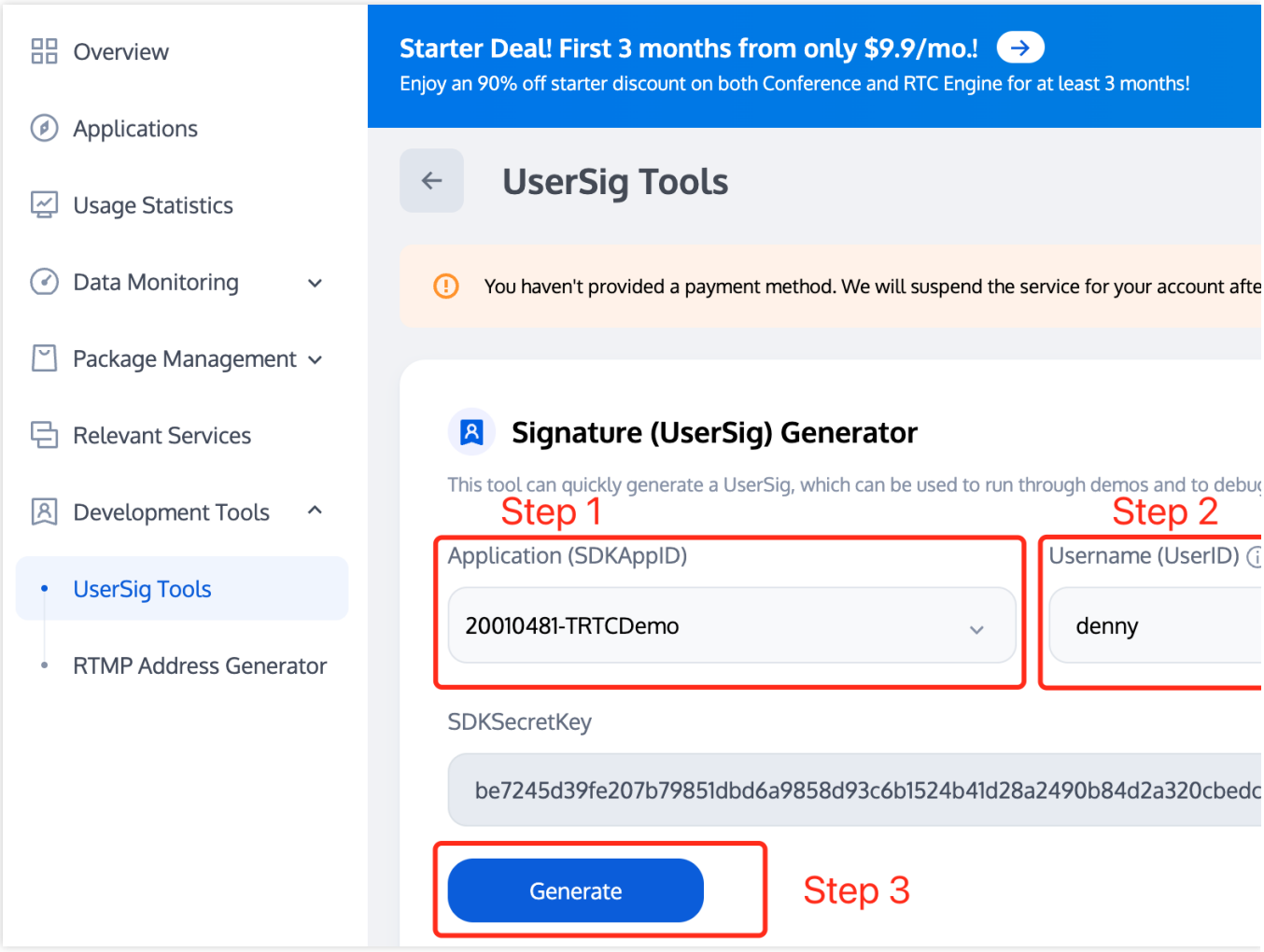
```
// Listen to the "onError" event, and print logs for errors such as "Camera is not
- (void)onError:(TXLiteAVError)errCode
         errMsg:(nullable NSString *)errMsg
        extInfo:(nullable NSDictionary *)extInfo{
    if (ERR_CAMERA_NOT_AUTHORIZED == errCode) {
        NSString *errorInfo = @"Current application is not authorized to use the ca
        errorInfo = [errorInfo stringByAppendingString : errMsg];
        [self toastTip:errorInfo];
    }
}
```

步骤4. 进入房间

1. 在Tencent RTC 控制台中单击 `Create Application` ，在 **Application Overview** 中获取 **SDKAppID** 。

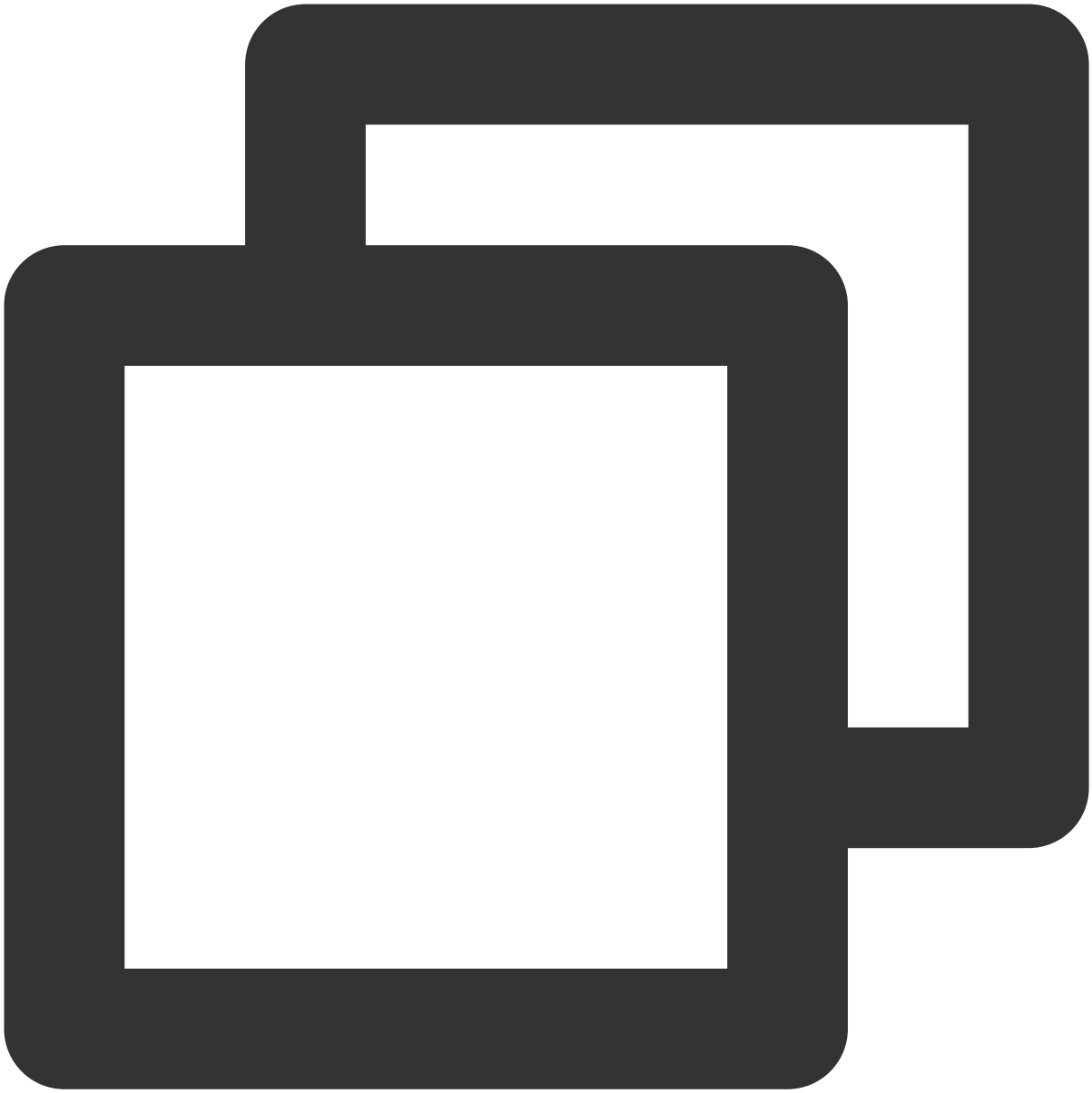


2. 在UserSig Tools中下来选择 **SDKAppID** ，并输入您自己的**用户名（UserID）**，单击 `Generate` 即可获取您自己的 **UserSig** 。



3. 在 `didFinishLaunchingWithOptions()` 方法中设置进房参数 **TRTCParams** 后，调用 `enterRoom` 接口函数即可进入房间。

主播角色：



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    // Override point for customization after application launch.

    // ...Other codes

    // Please replace each field in TRTCParams with your own parameters
    TRTCParams *trtcParams = [[TRTCParams alloc] init];
    trtcParams.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
    trtcParams.roomId = 123321; // Please replace with your own room number
    trtcParams.userId = @"denny";   // Please replace with your own userid
    trtcParams.userSig = @""; // Please replace with your own userSig
```

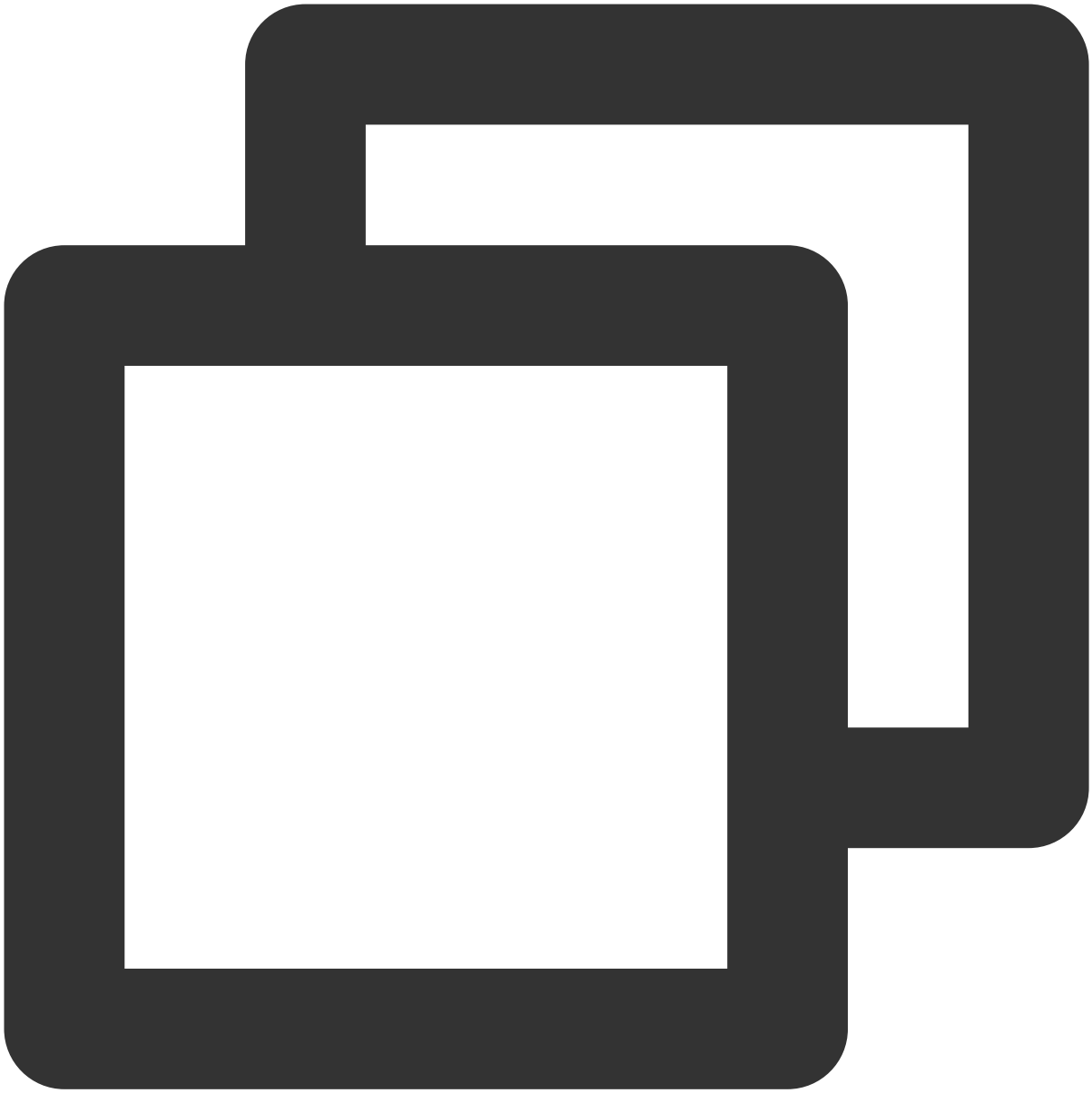
```
    trtcParams.role = TRTCRoleAnchor;

    // If your application scenario is a video call between several people, please
    [self.trtcCloud enterRoom:trtcParams appScene:TRTCAppSceneLIVE];

    return YES;
}

// Listen for the `onEnterRoom` event of the SDK and learn whether the room is succ
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        [self toastTip:@"Enter room succeed!"];
    } else {
        [self toastTip:@"Enter room failed!"];
    }
}
```

观众角色：



```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    // Override point for customization after application launch.

    // ...Other codes

    // Please replace each field in TRTCParams with your own parameters
    TRTCParams *trtcParams = [[TRTCParams alloc] init];
    trtcParams.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
    trtcParams.roomId = 123321; // Please replace with your own room number
    trtcParams.userId = @"denny";   // Please replace with your own userid
    trtcParams.userSig = @""; // Please replace with your own userSig
```

```
    trtcParams.role = TRTCRoleAudience;

    // If your application scenario is a video call between several people, please
    [self.trtcCloud enterRoom:trtcParams appScene:TRTCAppSceneLIVE];

    return YES;
}

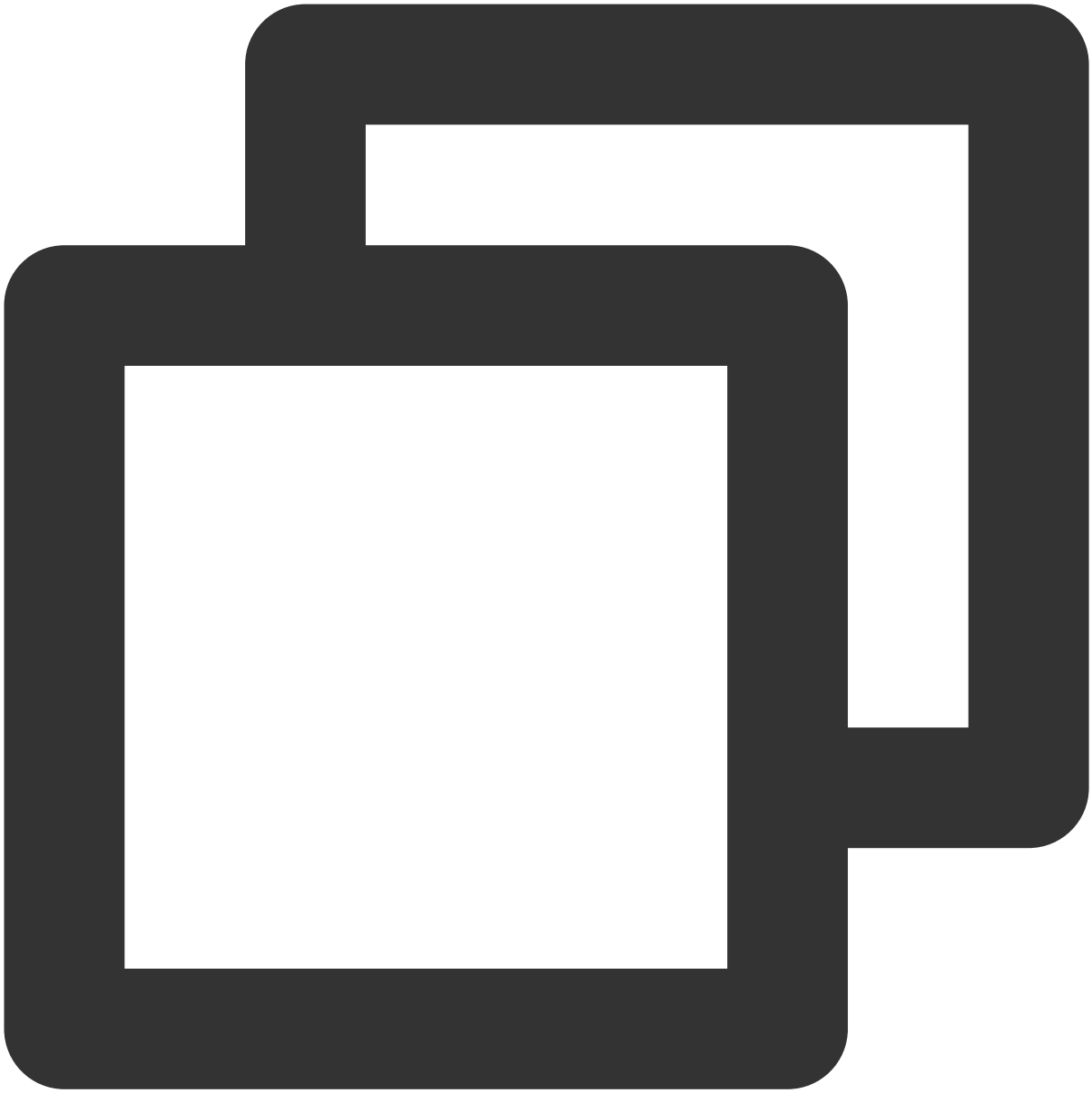
// Listen for the `onEnterRoom` event of the SDK and learn whether the room is succ
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        [self toastTip:@"Enter room succeed!"];
    } else {
        [self toastTip:@"Enter room failed!"];
    }
}
```

注意：

如果以**观众角色**进入房间，**sdkAppId** 和 **roomId** 需要与主播端相同，而**userId** 和 **userSig** 需要替换为自己的值。

步骤5. 打开摄像头

在 `didFinishLaunchingWithOptions()` 方法中初始化 **localCameraVideoView** ，并调用 `setLocalRenderParams` 接口来设置本地预览的渲染参数。



```
// Create a window
self.window = [[NSWindow alloc] initWithContentRect:NSMakeRect(0, 0, 800, 600) styl
[self.window center];
[self.window setTitle:@"TRTCDemo_Mac"];
[self.window makeKeyAndOrderFront:nil];
self.window.releasedWhenClosed = NO;

// Initialize localCameraVideoView
self.localCameraVideoView = [[NSView alloc] initWithFrame:NSMakeRect(0, 0, 300, 300
[self.window.contentView addSubview:self.localCameraVideoView];
```

版权所有：腾讯云计算（北京）有限责任公司

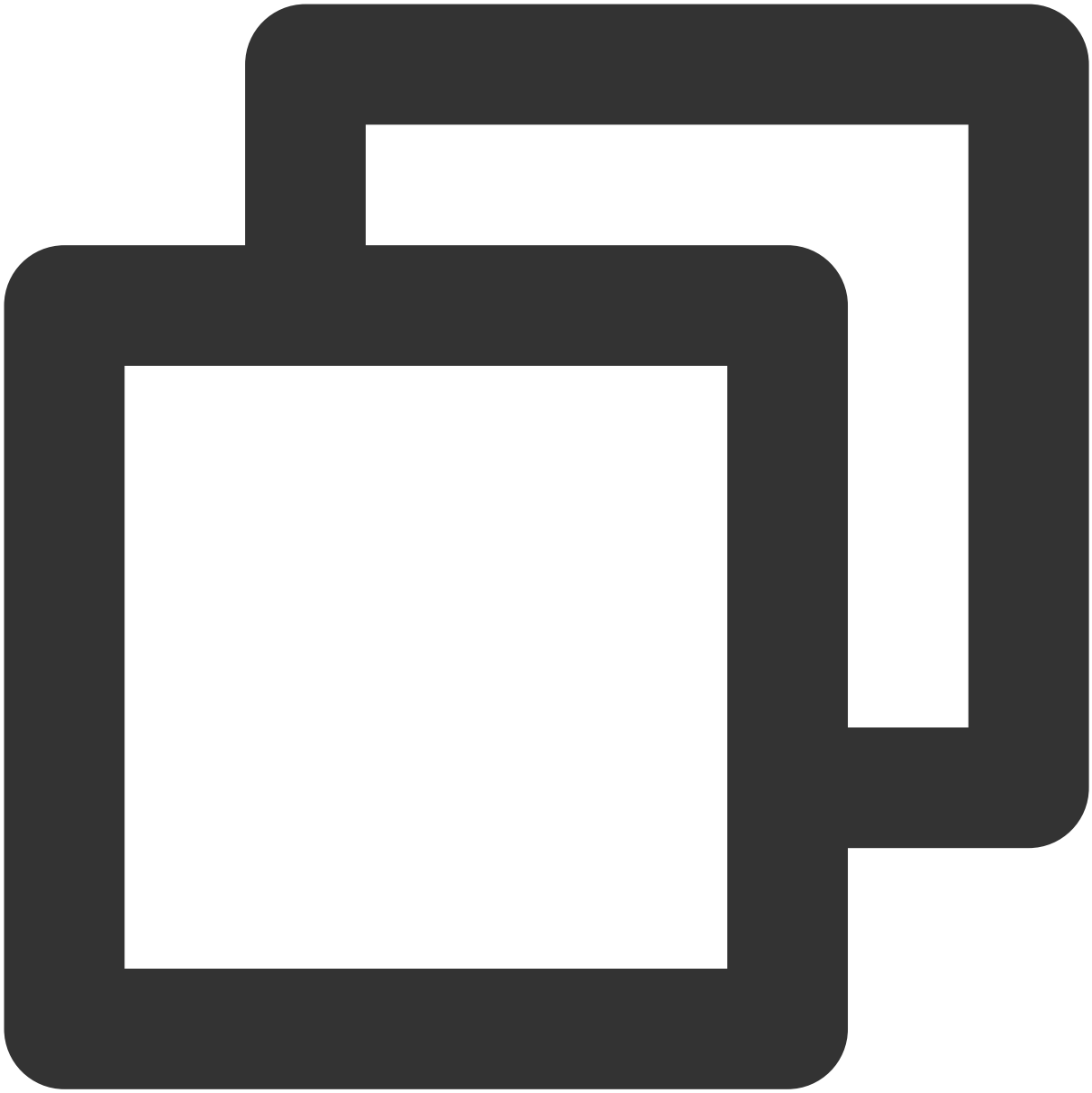
```
// Adjust the localCameraVideoView frame to match the size of the window
self.localCameraVideoView.frame = self.window.contentView.bounds;

// Set the preview mode of the local screen
TRTCRenderParams *trtcRenderParams = [[TRTCRenderParams alloc] init];
trtcRenderParams.fillMode   = TRTCVideoFillMode_Fill;
trtcRenderParams.mirrorType = TRTCVideoMirrorTypeAuto;
[self.trtcCloud setLocalRenderParams:trtcRenderParams];

// Start a preview of the local camera
[_trtcCloud startLocalPreview:_localCameraVideoView];
```

步骤6. 打开麦克风

您可以调用 `startLocalAudio` 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式，建议根据您的需求**选择以下其中一个适合您项目的模式**。

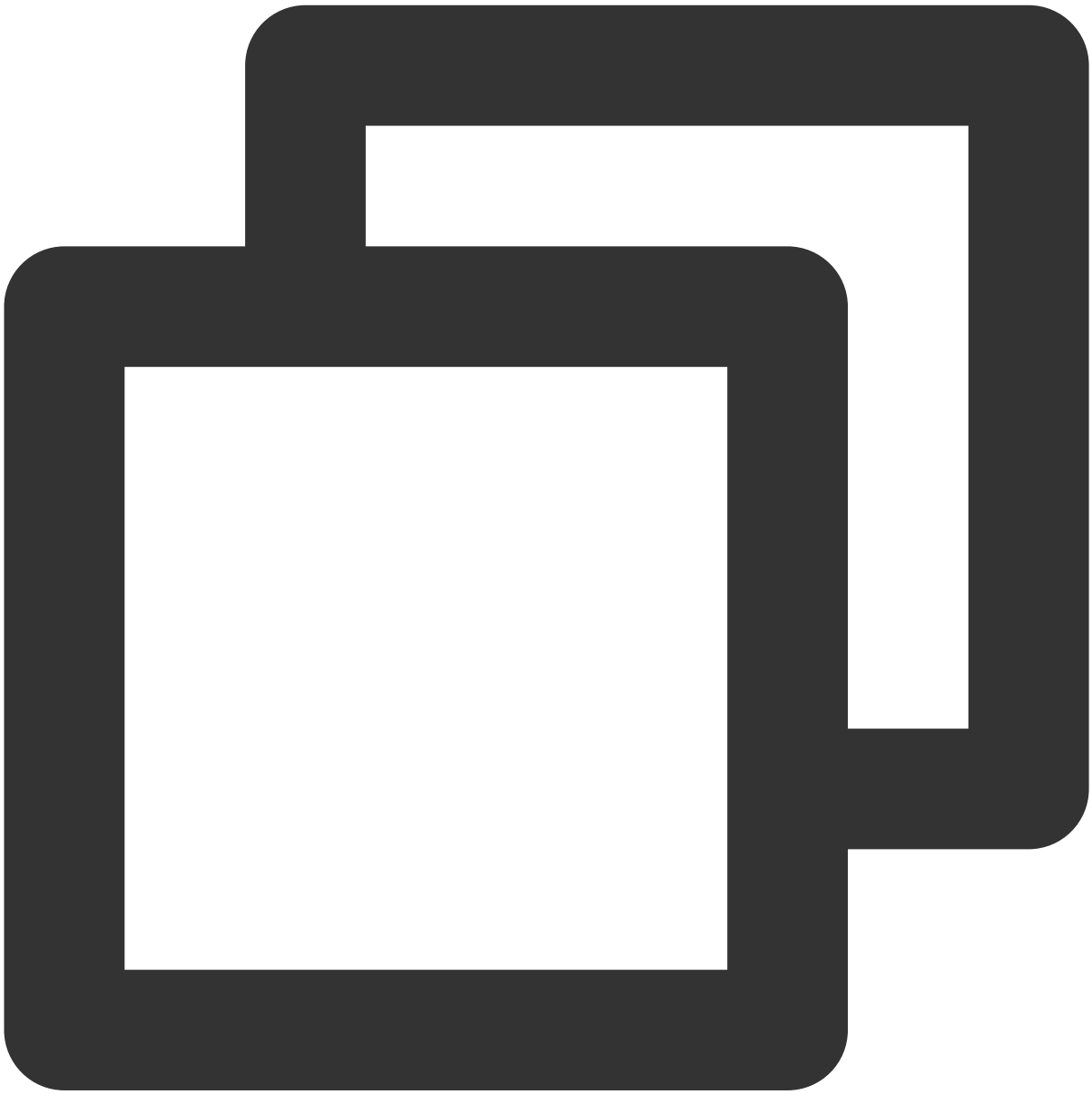


```
// Enable microphone acquisition and set the current scene to: Voice mode
// For high noise suppression capability, strong and weak network resistance
[self.trtcCloud startLocalAudio:TRTCAudioQualitySpeech];

// Enable microphone acquisition, and set the current scene to: Music mode
// For high fidelity acquisition, low sound quality loss, recommended to use with p
[self.trtcCloud startLocalAudio:TRTCAudioQualityMusic];
```

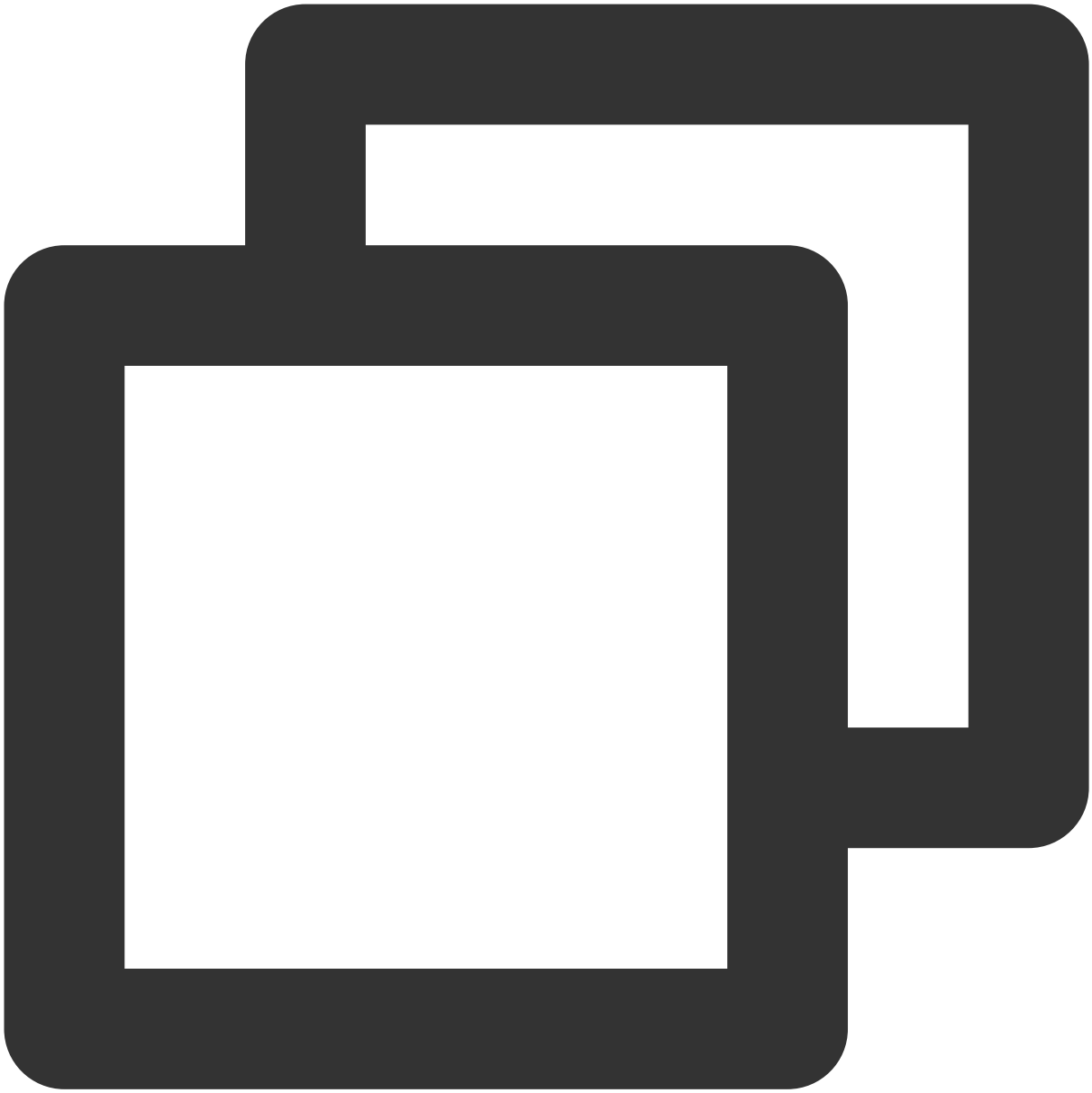
步骤7. 播放/停止视频流

按照**步骤1-4**以**观众角色**进入 **denny** 的房间后，您可以通过调用接口 `startRemoteView` 来播放远端用户的视频画面。



```
// Play denny's camera footage
[self.trtcCloud startRemoteView:@"denny" streamType:TRTCVideoStreamTypeBig view:cam
```

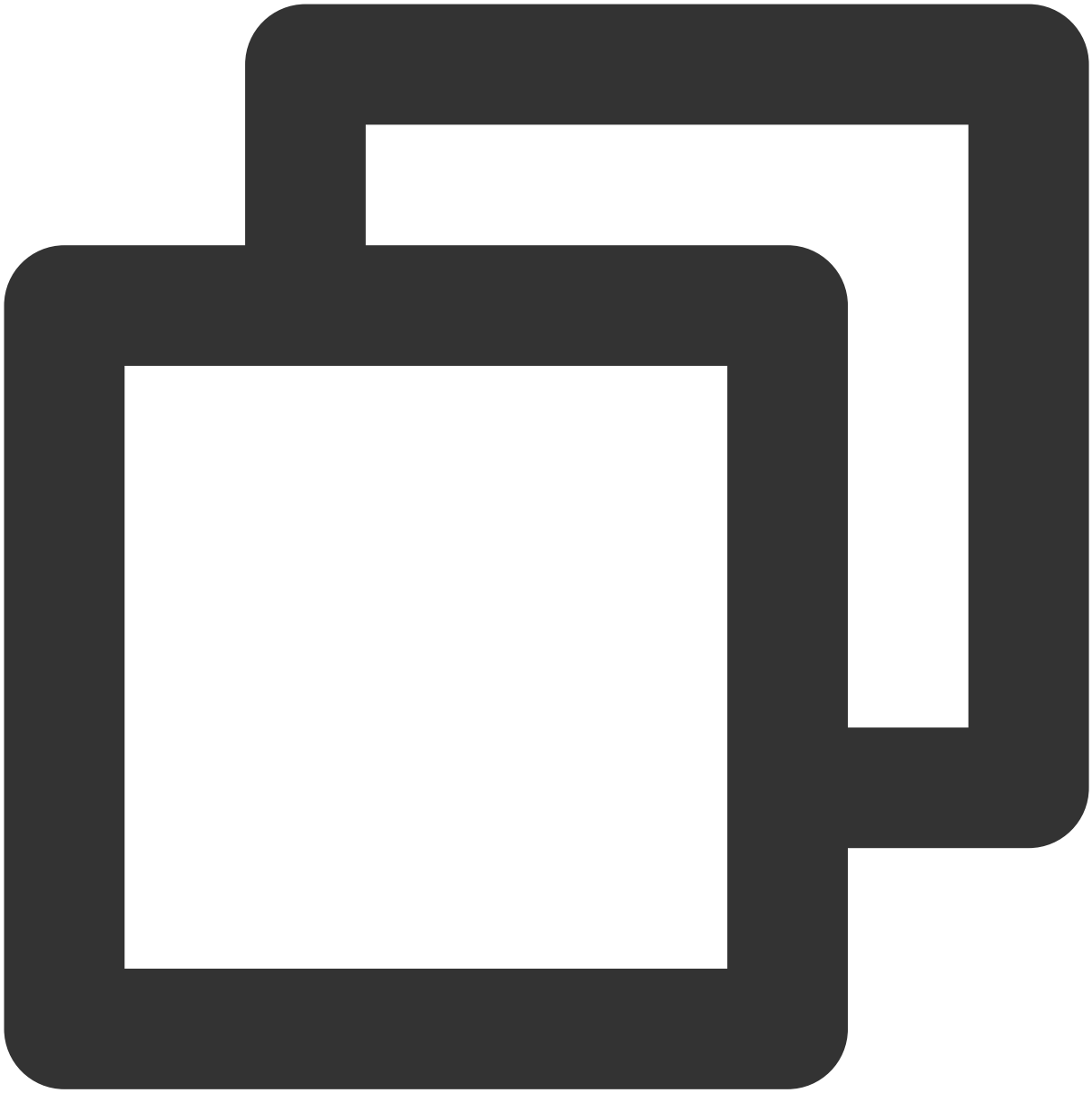
之后，您通过调用 `stopRemoteView` 接口停止某个远端用户的视频，也可以通过接口 `stopAllRemoteView` 停止播放所有远端用户的视频。



```
// Stop denny's camera footage
[self.trtcCloud stopRemoteView:@"denny" streamType:TRTCVideoStreamTypeBig view:came
// Stop all camera footages
[self.trtcCloud stopAllRemoteView];
```

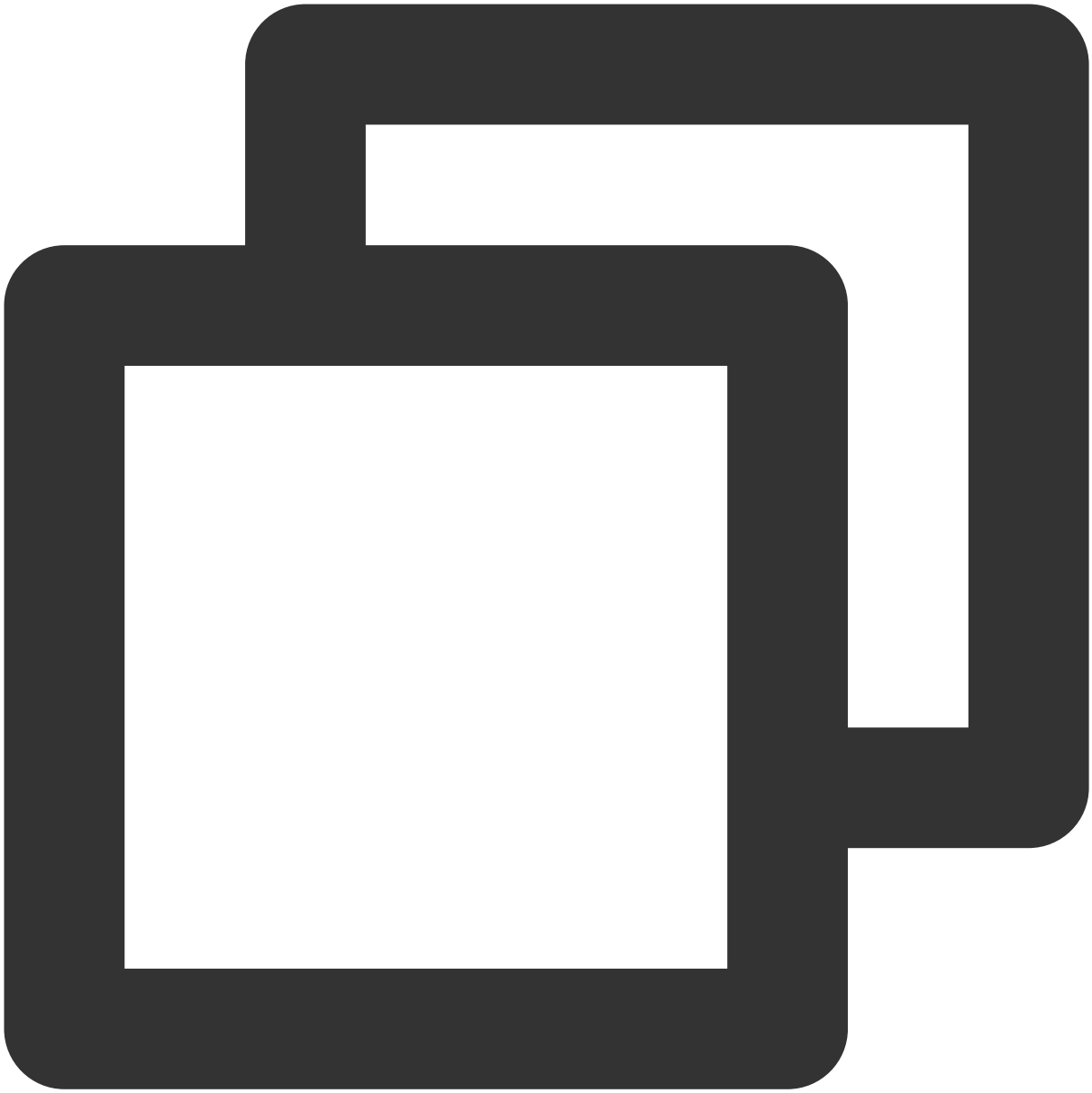
步骤8. 播放/停止音频流

您可以通过调用接口 `muteRemoteAudio("denny", true)` 来静音远端用户 **denny** 的声音。



```
// Mute user with id denny
[self.trtcCloud muteRemoteAudio:@"denny" mute:YES];
```

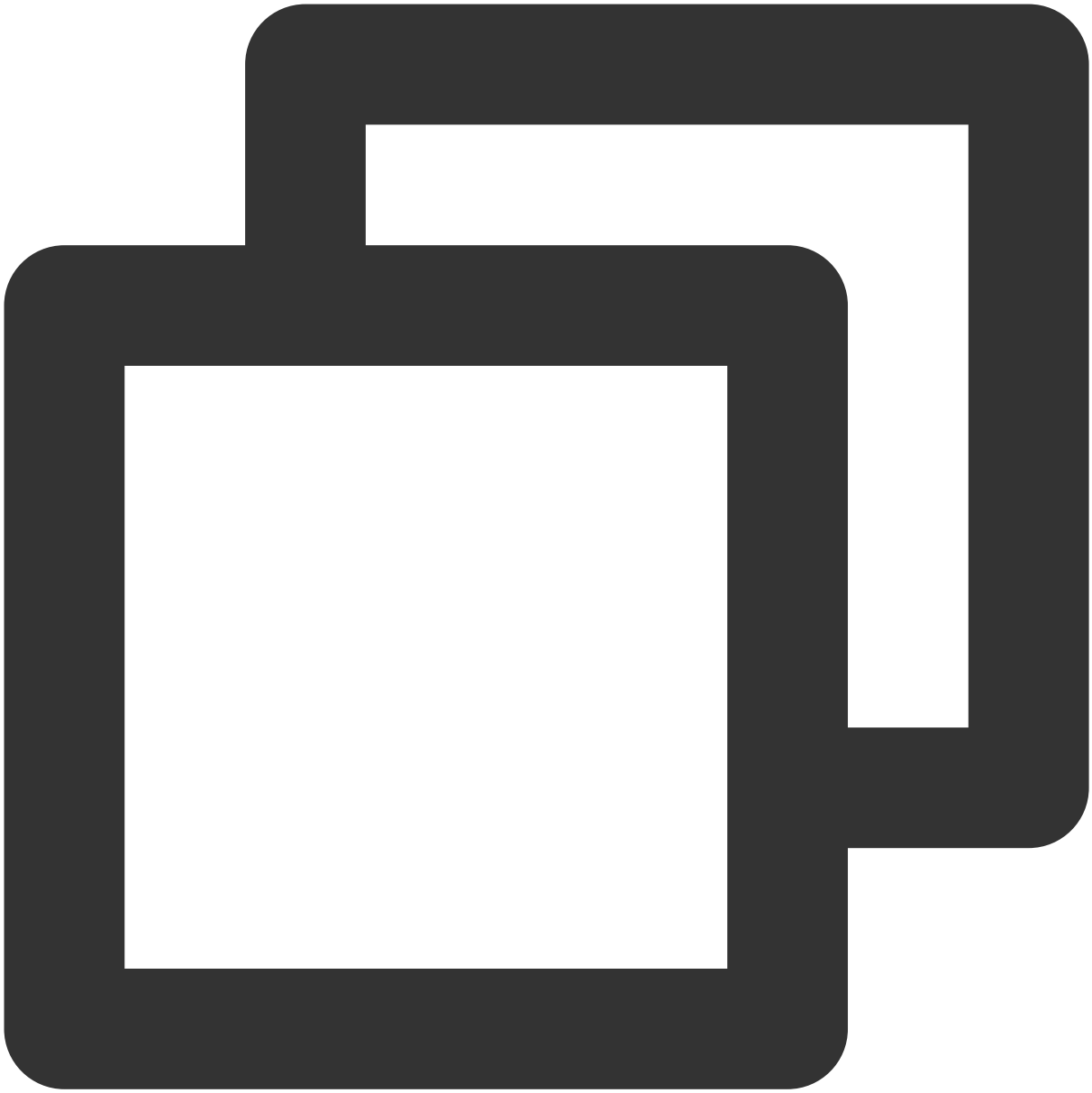
之后也可以通过调用接口 `muteRemoteAudio("denny", false)` 来解除对他的静音。



```
// Unmute user with id denny
[self.trtcCloud muteRemoteAudio:@"denny" mute:YES];
```

步骤9. 退出房间

您可以调用接口 `exitRoom` 即可退出当前的房间，SDK 会在退房结束后通过 `onExitRoom(int reason)` 回调事件通知您。



```
// Exit current room
[self.trtcCloud exitRoom];

// Listen for the onExitRoom callback to find out why you checked out
- (void)onExitRoom:(NSInteger)reason {
    if (reason == 0) {
        NSLog(@"Exit current room by calling the 'exitRoom' api of sdk ...");
    } else if (reason == 1) {
        NSLog(@"Kicked out of the current room by server through the restful api...
    } else if (reason == 2) {
        NSLog(@"Current room is dissolved by server through the restful api...");
```

```
        }
    }
```

常见问题

您可以在 API 参考 查看所有函数列表及其描述。
如果您的接入和使用中遇到问题，请参见 常见问题。

联系我们

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com。

Windows C++

最近更新时间：2024-07-18 15:23:04

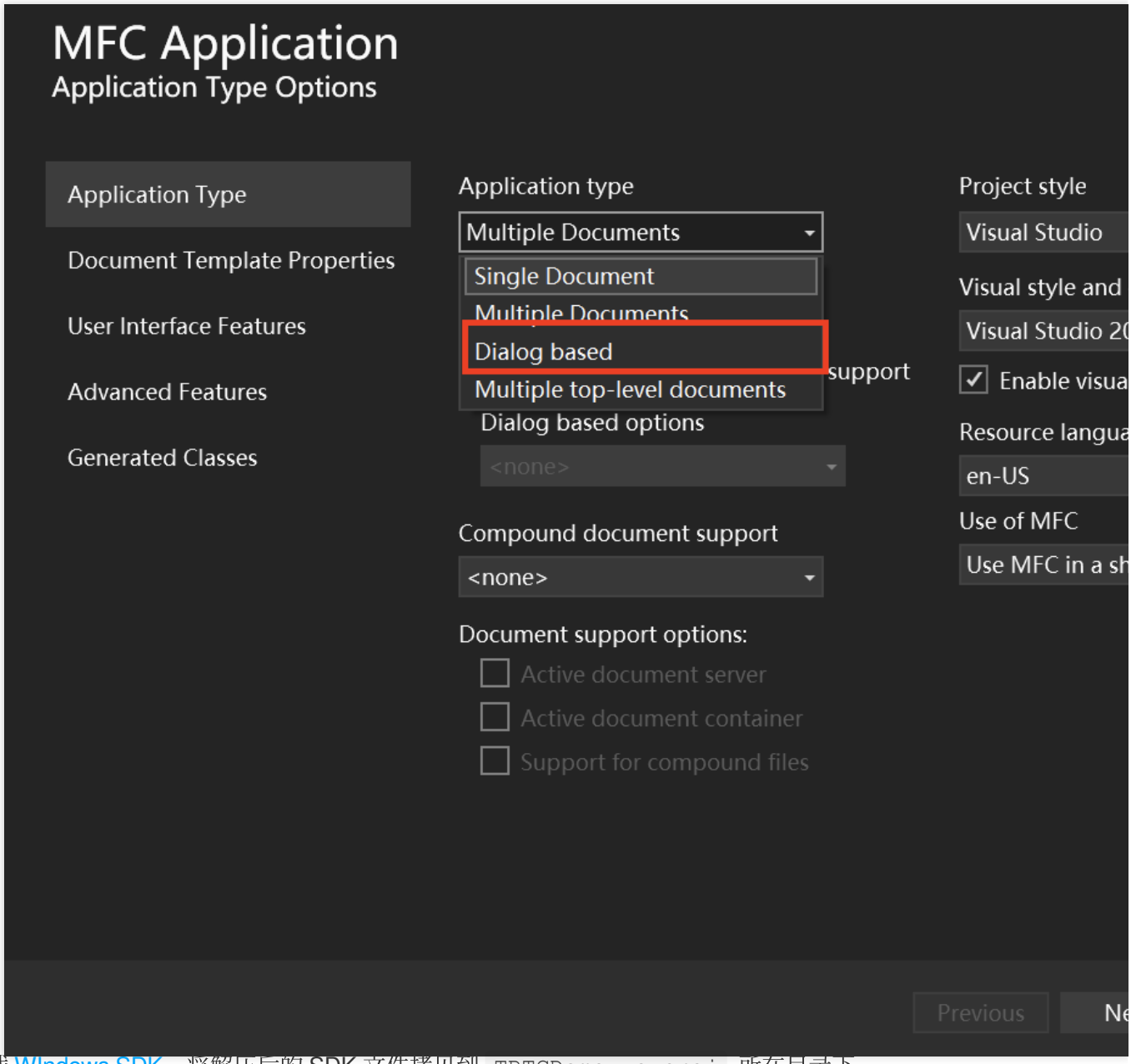
本文将介绍如何快速完成 Windows RTC Engine 的接入，实现一个基本的音视频通话。

环境准备

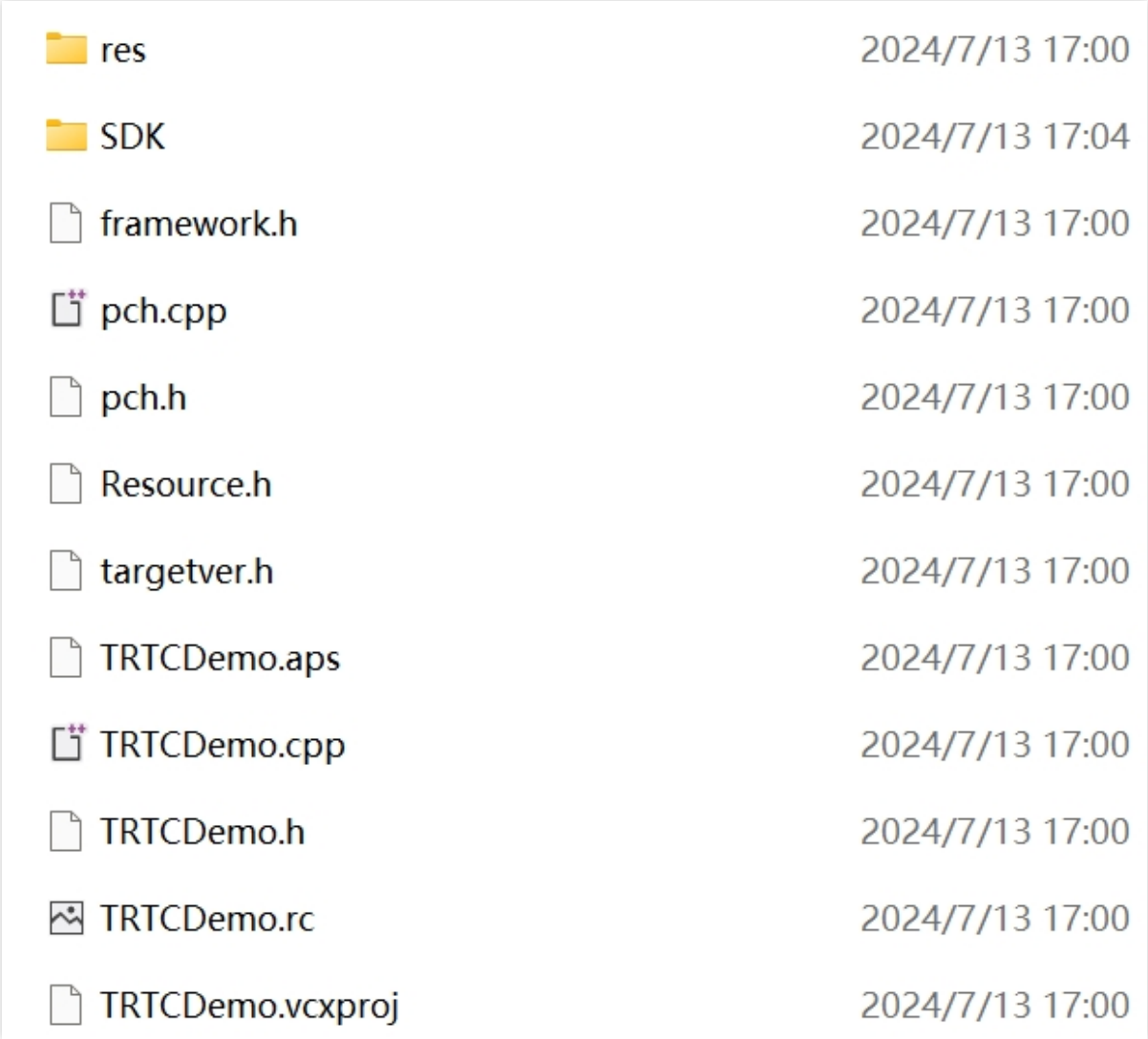
操作系统：Windows 7及以上版本。
开发环境：Visual Studio 2010及以上版本，推荐使用 Visual Studio 2015。

步骤1. 导入 SDK

1. 打开 Visual Studio，新建一个名为 TRTCDemo 的 MFC 应用程序。在向导的 **MFC Application** 页面，**Application type** 选择 **Dialog based**，其他向导配置选择默认即可，单击 `Finish` 。



2. 下载 WIndows SDK，将解压后的 SDK 文件拷贝到 `TRTCDemo.vcxproj` 所在目录下。

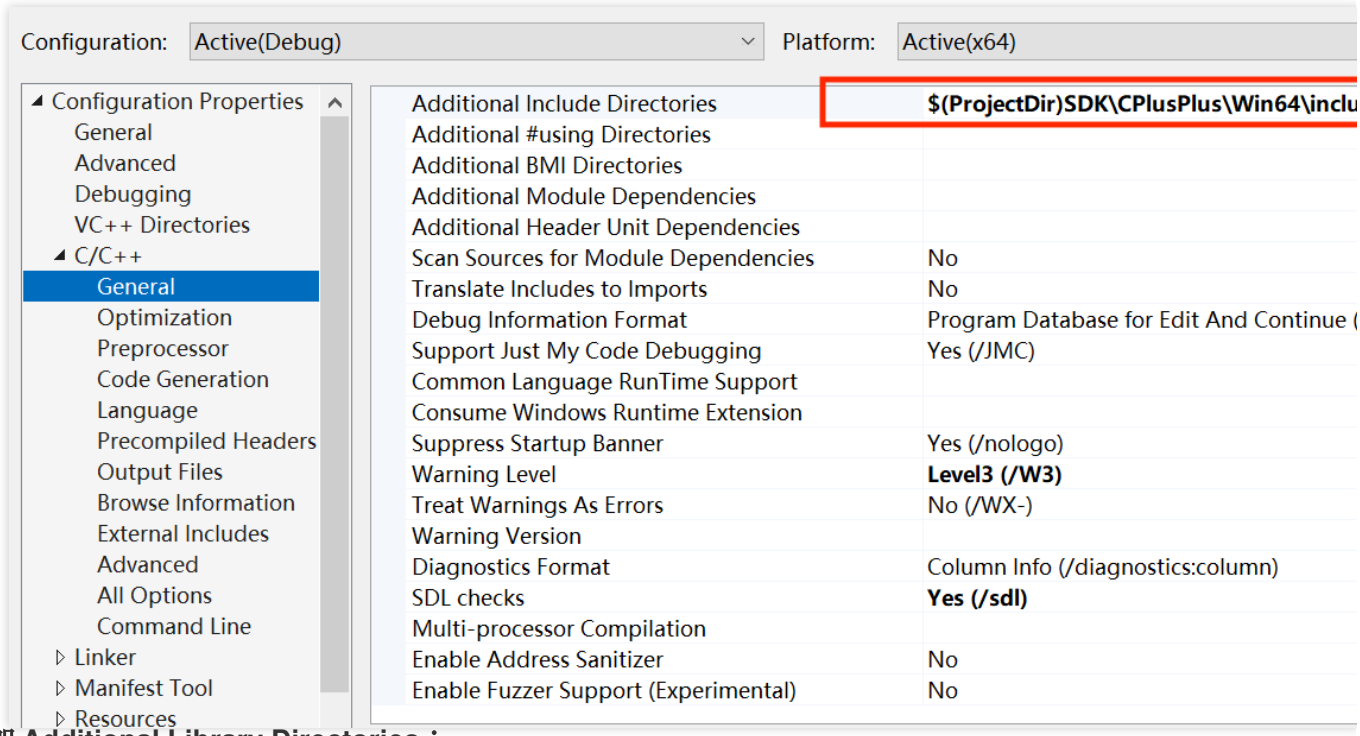


步骤2. 配置项目

打开 **TRTCDemo.sln** 属性页，在 **Solution Explorer** > **TRTCDemo 工程的右键菜单** > **Properties**，进行以下配置：

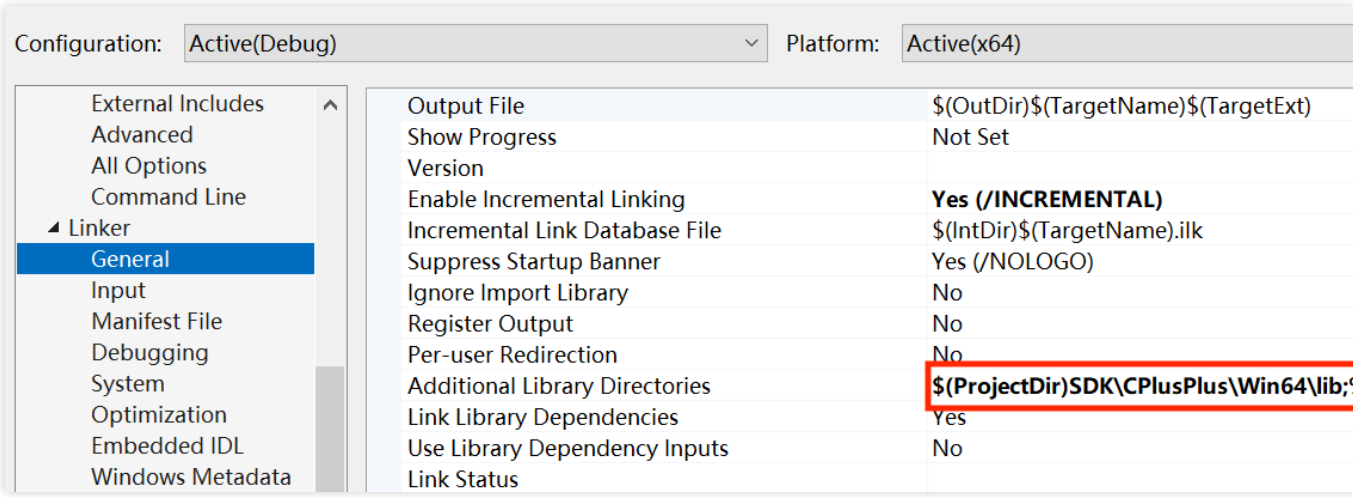
1. **添加 Additional Include Directories：**

在 **C/C++** > **General** > **Additional Include Directories**，添加 SDK 头文件目录：`$(ProjectDir)SDK\\CPlusPlus\\Win64\\include` 和 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include\\TRTC`。



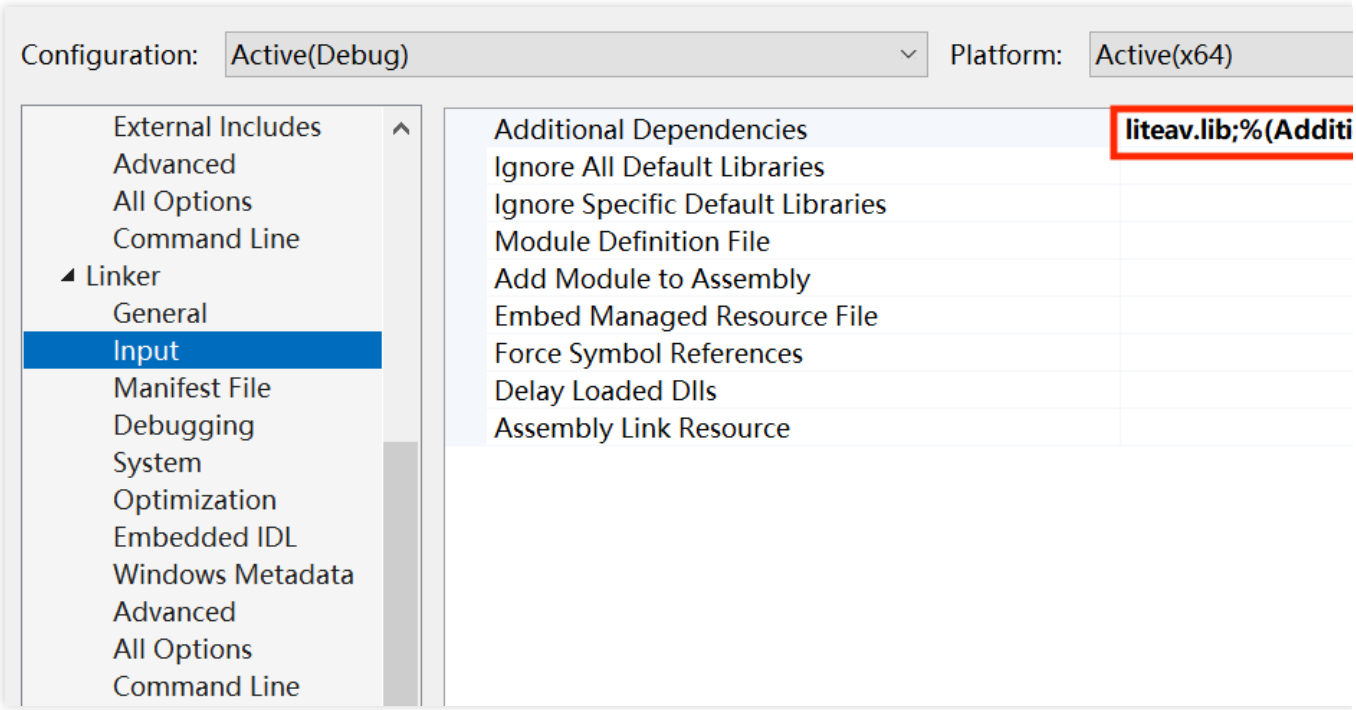
2. **添加 Additional Library Directories：**

在 **Linker** > **General** > **Additional Library Directories**，添加 SDK 库目录 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\lib` 。



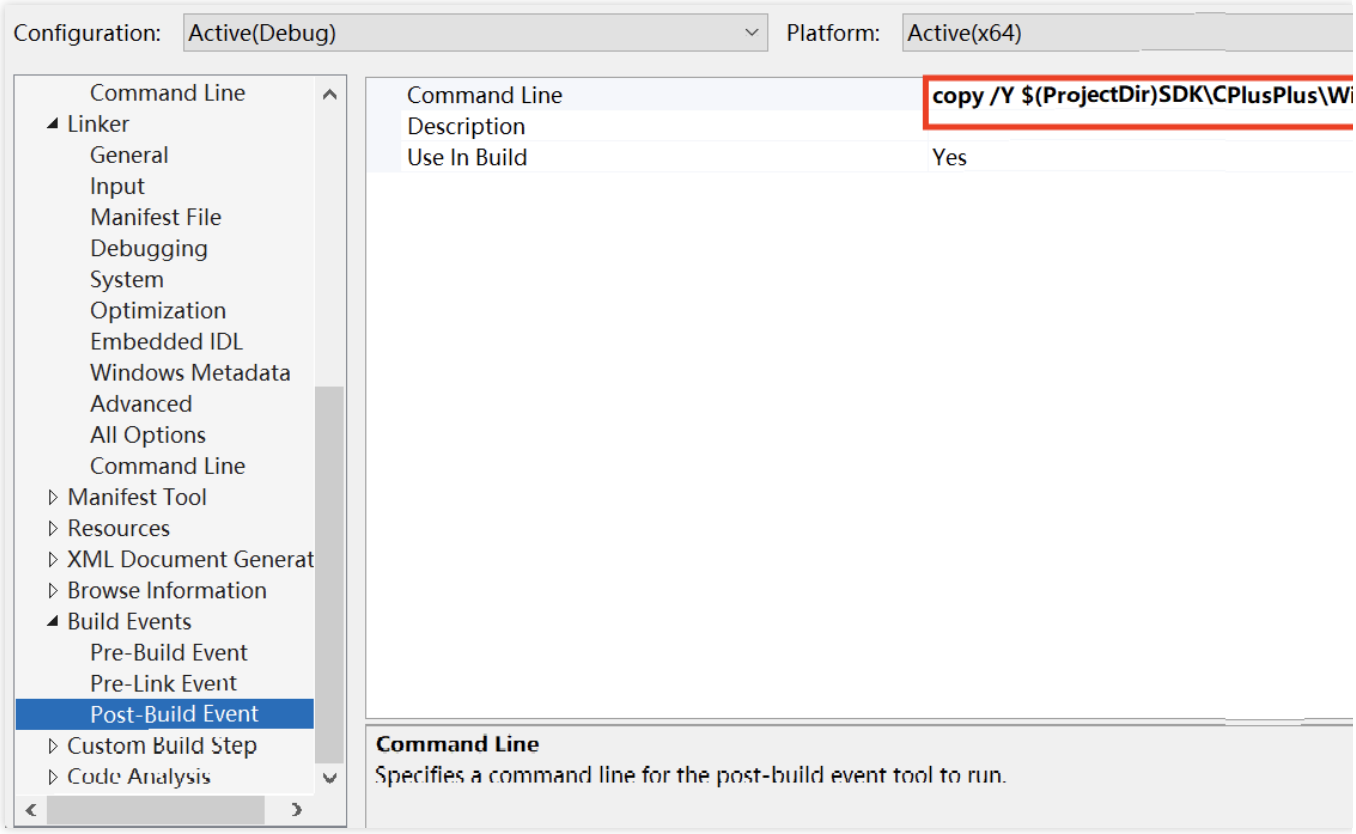
3. **添加 Additional Dependencies：**

在 **Linker** > **Input** > **Additional Dependencies**，添加 SDK 库文件 `liteav.lib` 。



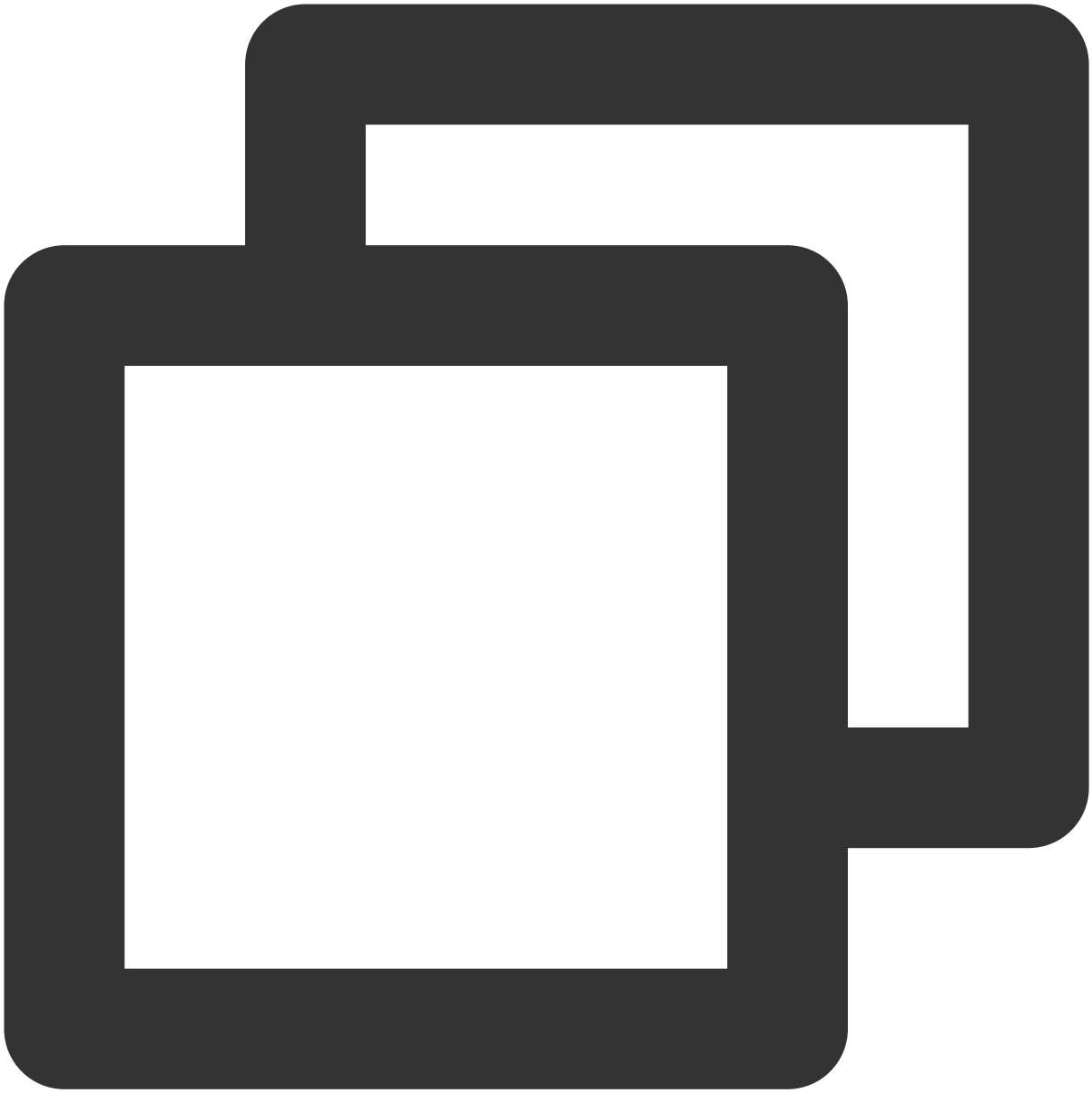
4. **添加 Command line：**

在 **Build Events** > **Post-Build Event** > **Command line**，添加 `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win64\\lib\\*.dll  $(OutDir)` 。



5. **打印 SDK 版本号：**

在 **TRTCDemoDlg.cpp** 文件顶部引入头文件：

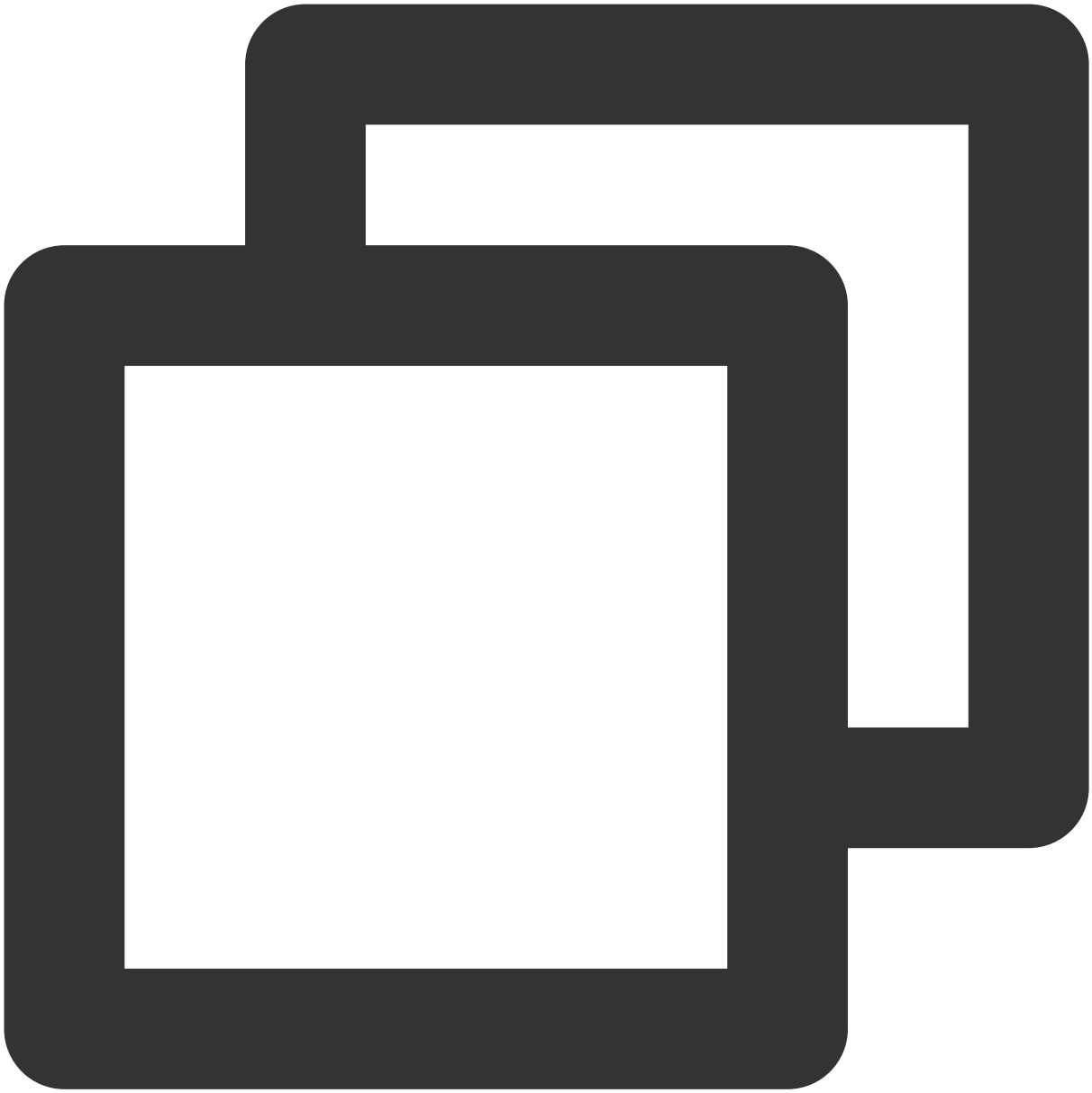


```
#include "ITRTCCloud.h"
```

注意：

在已有头文件的后面引用 `"ITRTCCloud.h"`，否则将会出现报错：`ITRTCCloud：未定义标识符`。

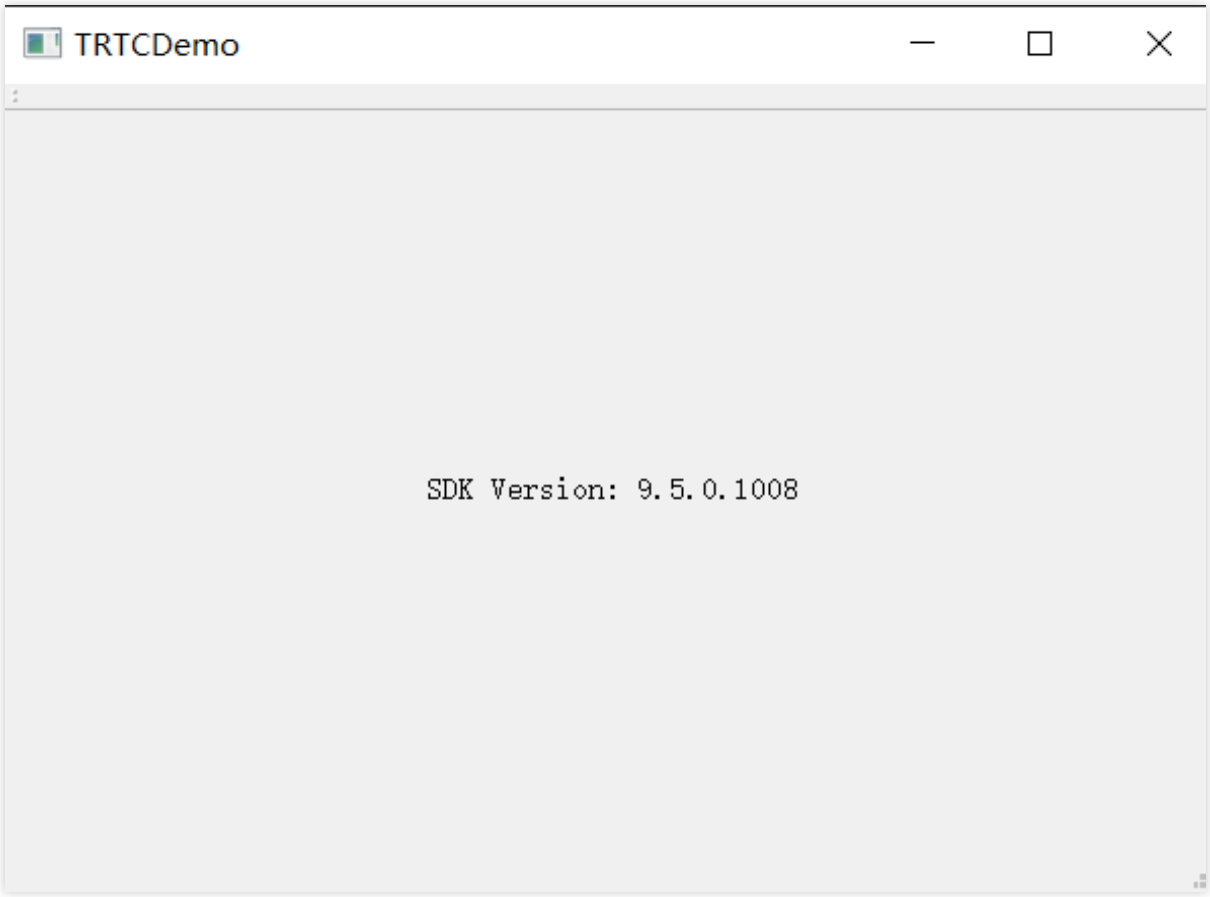
在 `CTRTCDemoDlg::OnInitDialog` 函数中，添加以下代码：



```
ITRTCCloud * pTRTCCloud = getTRTCShareInstance();
CString szText;
szText.Format(L"SDK version: %hs", pTRTCCloud->getSDKVersion());

CWnd *pStatic = GetDlgItem(IDC_STATIC);
pStatic->SetWindowTextW(szText);
```

完成以上步骤，单击**运行**可以打印 SDK 版本号。

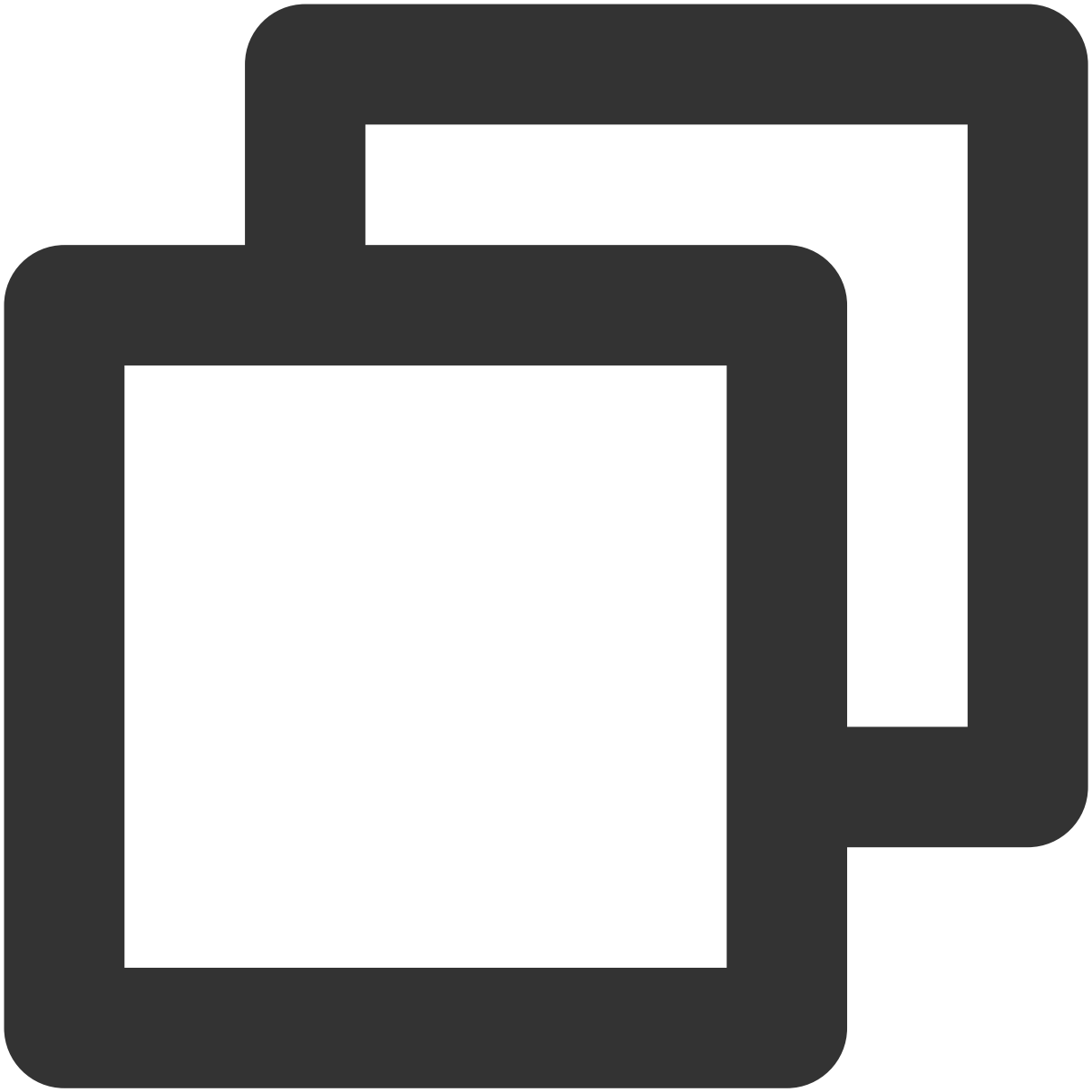


说明：

如果出现报错信息：module machine type 'x86' conflicts with target machine type 'x64'，在解决方案平台选择 'x64'。

步骤3. 创建 TRTC 实例

1. 在 **TRTCDemo.h** 文件中引用头文件 `"ITRTCCloud.h"`，将 **CTRTCDemo** 类公有继承于 **CWinApp** 和 **ITRTCCloudCallback**，并声明回调函数和成员变量。



```
#include "ITRTCCloud.h"

class CTRTCDemoApp : public CWinApp, public ITRTCCloudCallback
{
public:
        CTRTCDemoApp();

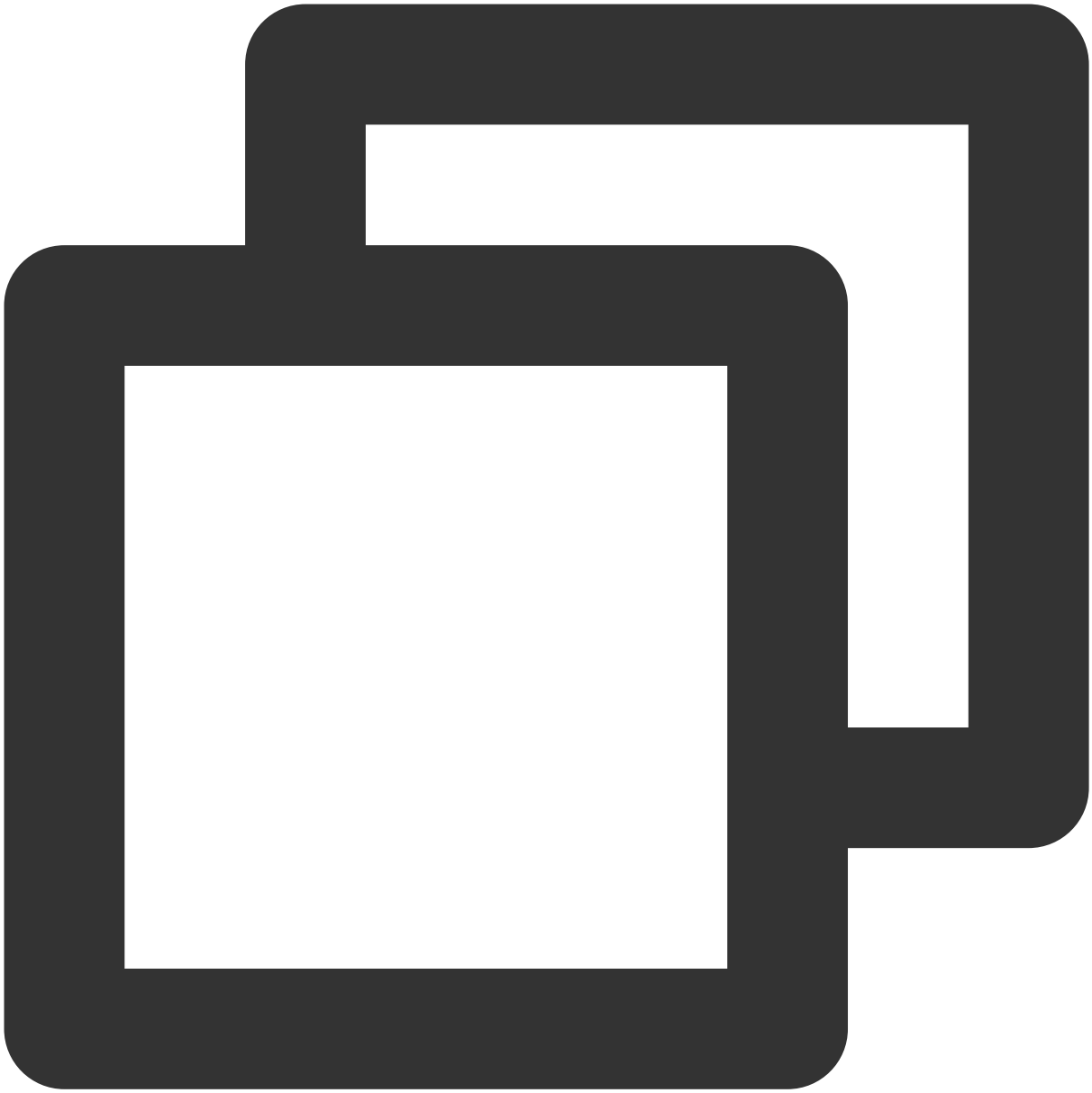
// Overwrite
public:
        virtual BOOL InitInstance();
        virtual void onError(TXLiteAVError errCode, const char* errMsg, void* extra
```

```
        virtual void onWarning(TXLiteAVWarning warningCode, const char* warningMsg,
        virtual void onEnterRoom(int result) override;
        virtual void onExitRoom(int reason) override;
// Emplement

        DECLARE_MESSAGE_MAP()

private:
        ITRTCCloud* trtc_cloud_; // Declare the member variable trtc cloud
};
```

2. 在 **TRTCDemo.cpp** 文件的 `CTRTCDemo::InitInstance()` 方法中调用初始化接口创建 TRTC 的对象实例。

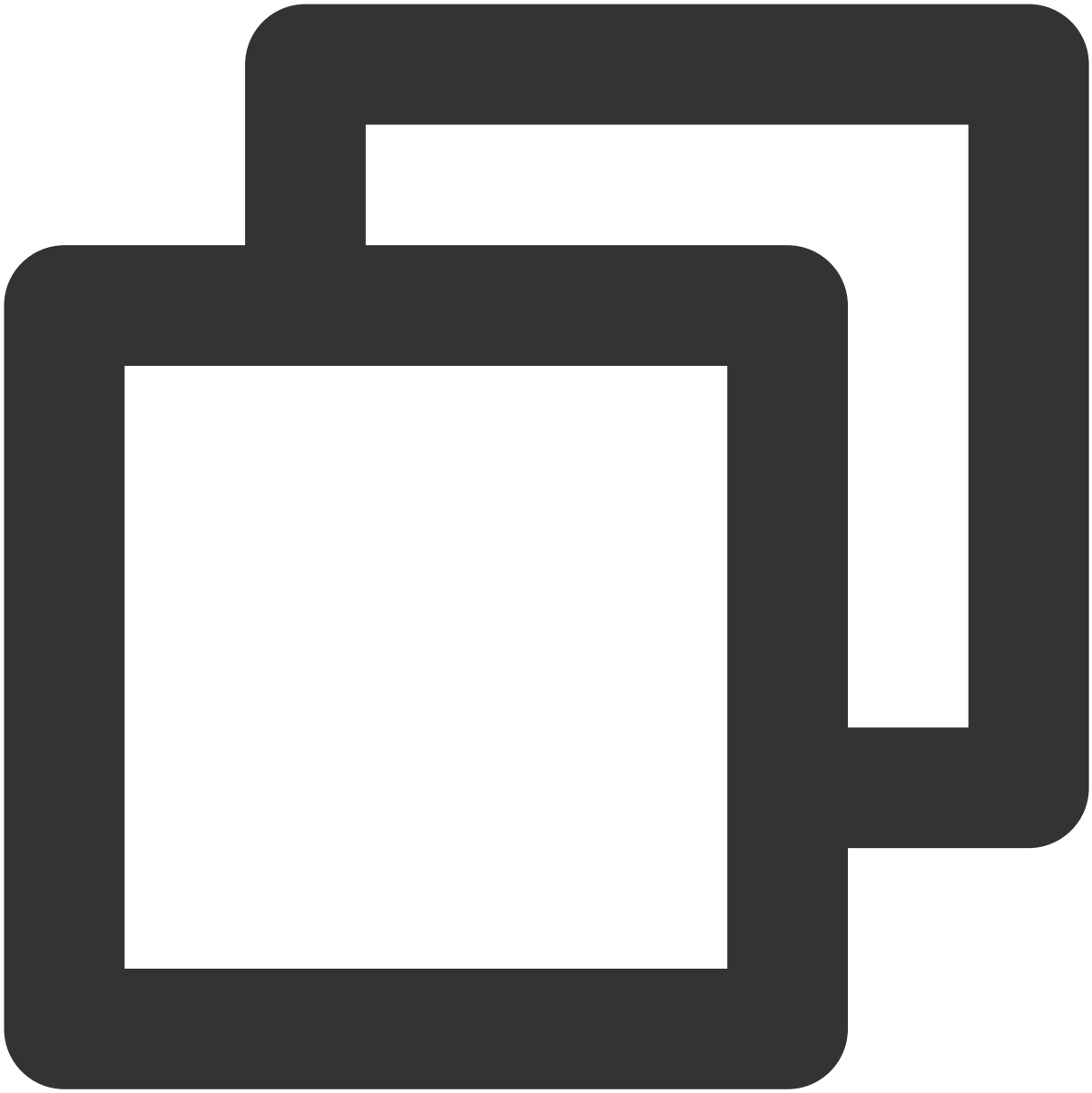


```
// Create trtc instance(singleton)  and set up event listeners
trtc_cloud_ = getTRTCShareInstance();
trtc_cloud_->addCallback(this);
```

说明：

将初始化接口代码添加在 `SetRegistryKey(_T("Local AppWizard-Generated Applications"))` 方法后。

3. 实现声明的回调方法。



```
// Listen for SDK events, and print logs for error messages such as "Current applic
// Listen to the "onError" event, and print logs for errors such as "Camera is not
void CTRTCDemoApp::onError(TXLiteAVError errCode, const char* errMsg, void* extraIn
        if (errCode == ERR_CAMERA_NOT_AUTHORIZED) {
                printf("Current application is not authorized to use the camera");
        }
}

// Listen for SDK events, and print logs for warning messages
// Listen to the "onWarning" event, and print logs for errors such as "WARNING_VIDE
void CTRTCDemoApp::onWarning(TXLiteAVWarning warningCode, const char* warningMsg, v
```

```
        if (warningCode == WARNING_VIDEO_RENDER_FAIL) {
                printf("WARNING_VIDEO_RENDER_FAIL");
        }

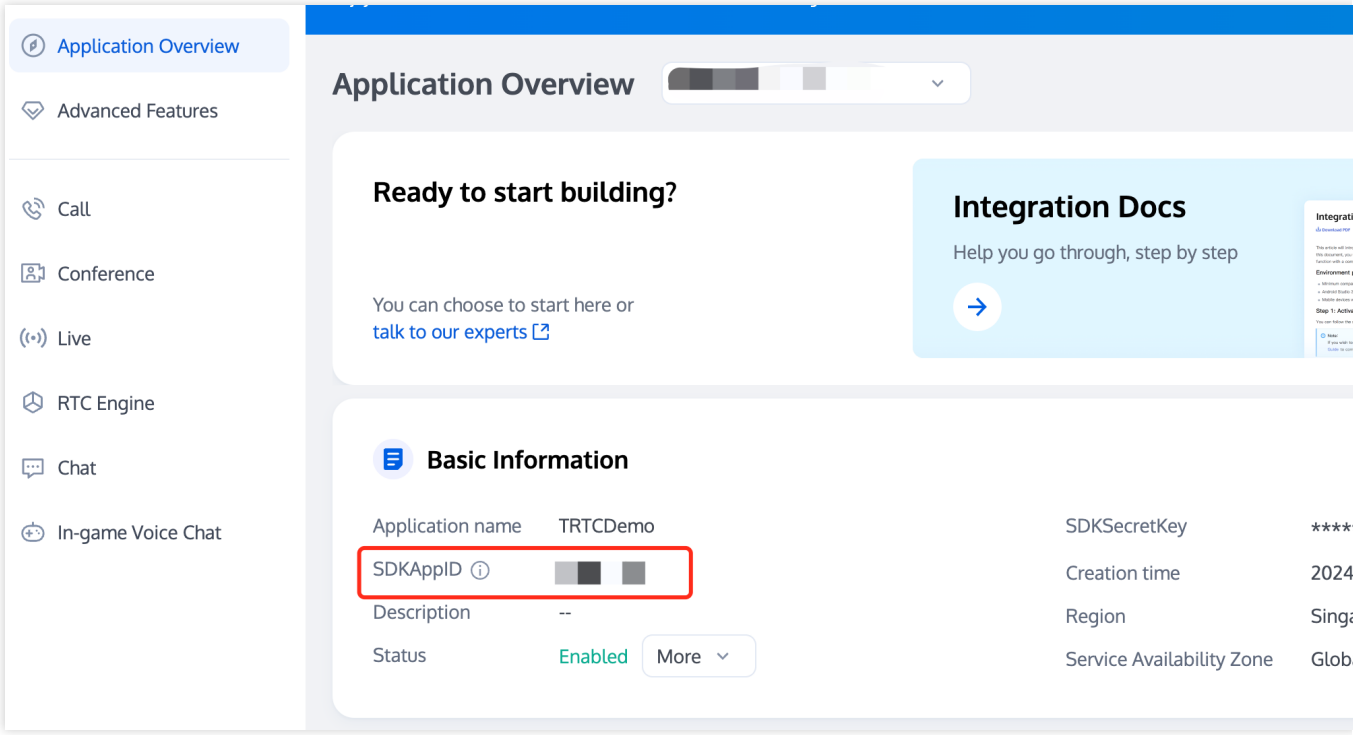
}

// Listen for the `onEnterRoom` event of the SDK to get the room entry result
// override to the onEnterRoom event of the SDK and learn whether the room is succe
void CTRTCDemoApp::onEnterRoom(int result) {
        if (result > 0) {
                printf("Enter room succeed");
        }
        else {
                printf("Enter room failed");
        }
}

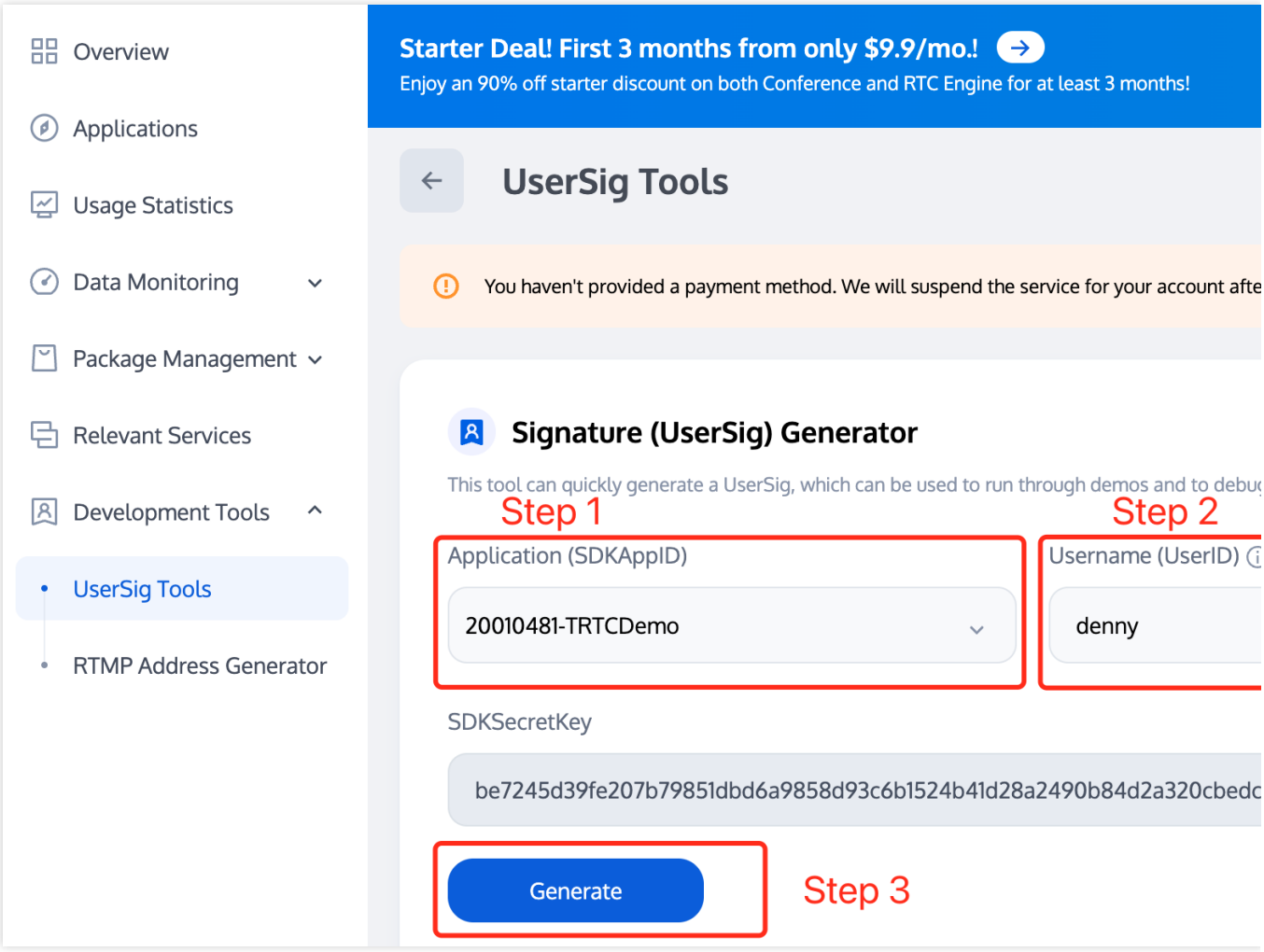
// Listen for the `onExitRoom` event of the SDK to get the room exit result
// override to the onExitRoom event of the SDK and learn whether the room is succes
void CTRTCDemoApp::onExitRoom(int reason) {
    if (reason == 0) {
        printf("Exit current room by calling the 'exitRoom' api of sdk ...");
    } else if (reason == 1) {
        printf("Kicked out of the current room by server through the restful api...
    } else if (reason == 2) {
        printf("Current room is dissolved by server through the restful api...");
    }
}
```

步骤4. 进入房间

1. 在Tencent RTC 控制台中单击 `Create Application` ，在 **Application Overview** 中获取 **SDKAppID** 。

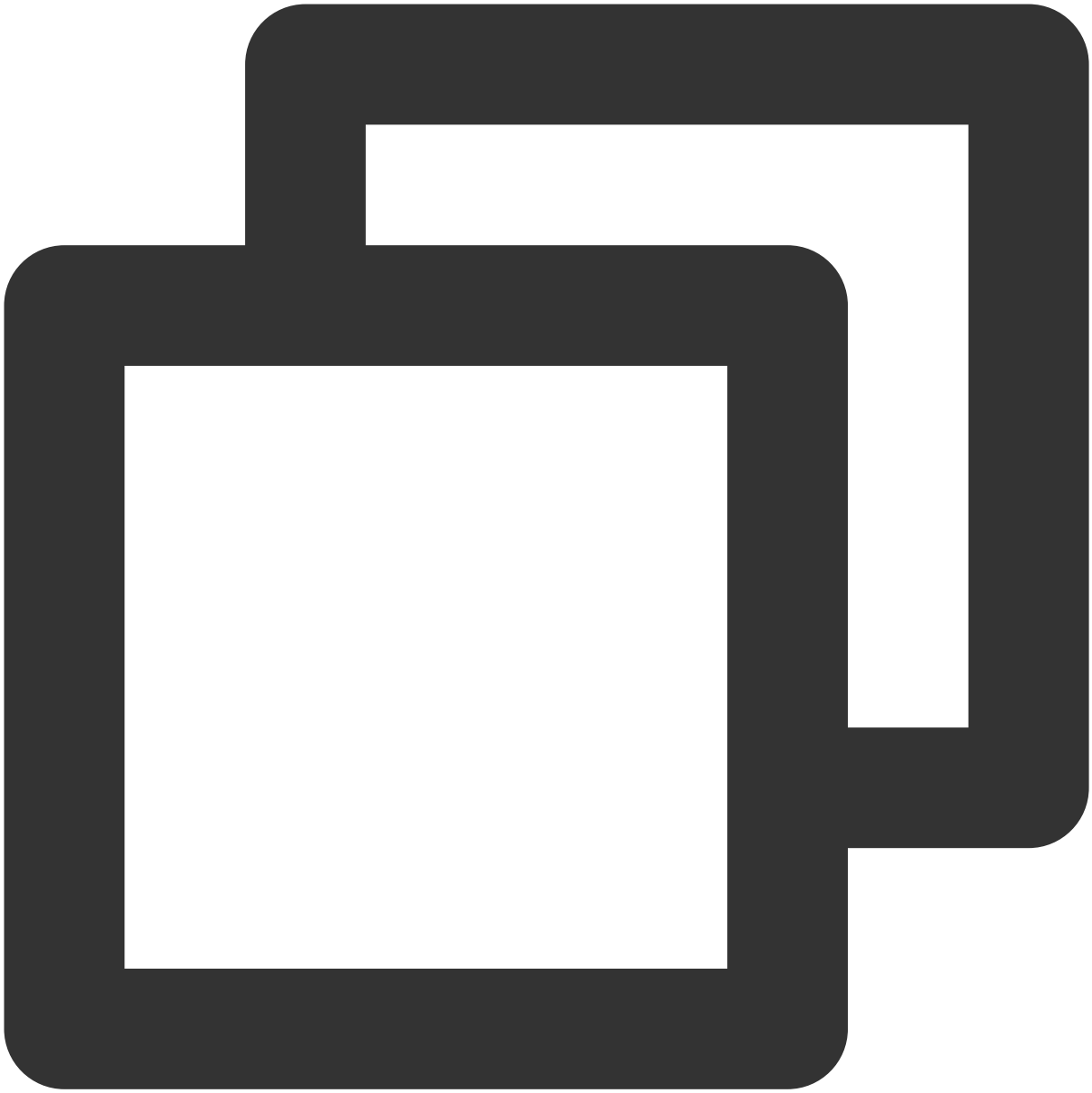


2. 在UserSig Tools中下来选择 **SDKAppID**，并输入您自己的**用户名（UserID）**，单击 `Generate` 即可获取您自己的 **UserSig** 。



3. 在 `CTRTCDemo::InitInstance()` 方法中设置进房参数 **TRTCParams** 后，调用 `enterRoom` 接口函数即可进入房间。

主播角色：



```
BOOL CTRTCDemo::InitInstance()
{
    // ...other codes

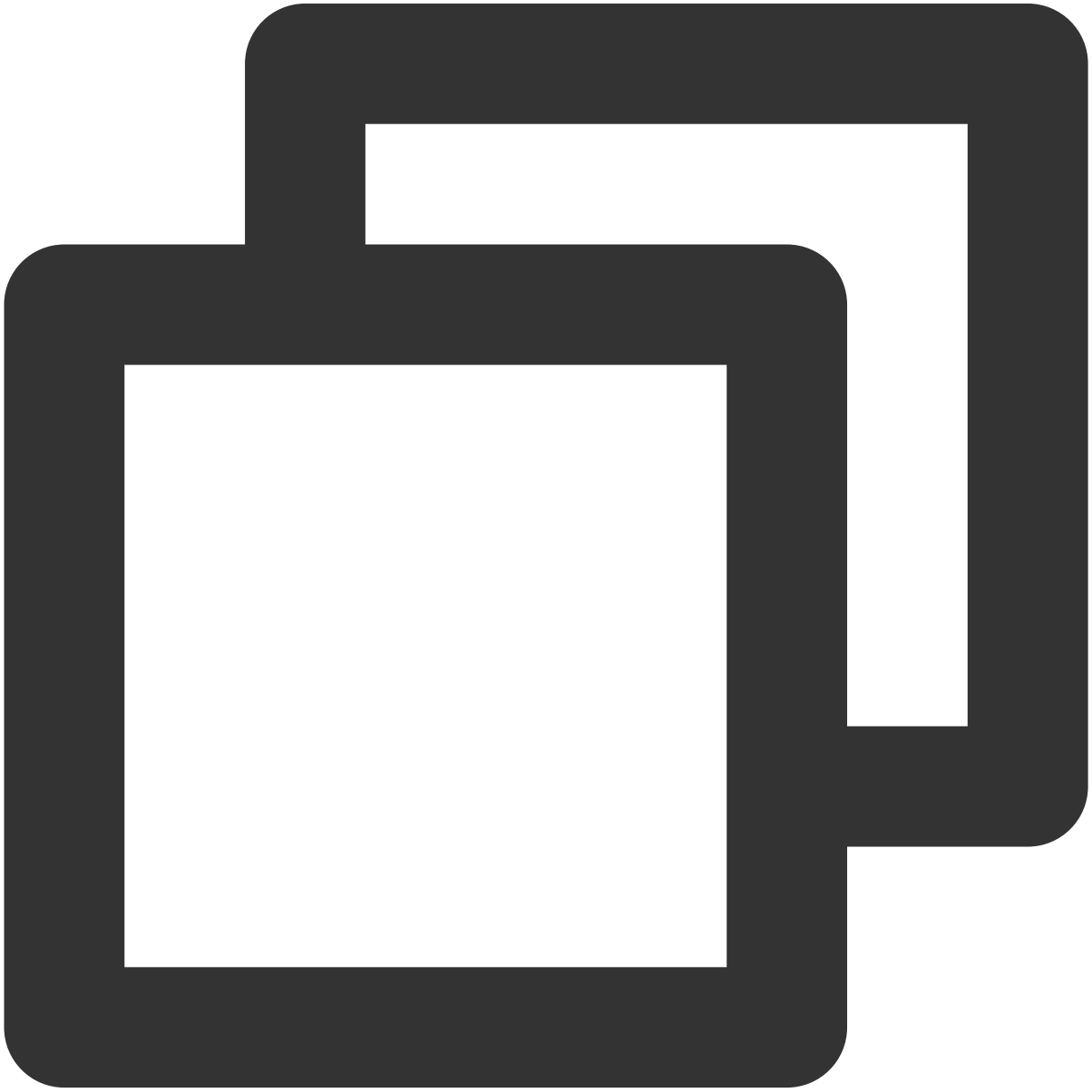
    // Assemble TRTC room entry parameters. Replace the field values in `TRTCParams
    // Replace each field in TRTCParams with your own parameters
    liteav::TRTCParams trtcParams;
    trtcParams.sdkAppId = 1400000123;  // Replace with your own SDKAppID
    trtcParams.userId = "denny";       // Replace with your own user ID
    trtcParams.roomId = 123321;        // Replace with your own room number
    trtcParams.userSig = "xxx";        // Replace with your own userSig
```

```
    trtcParams.role = liteav::TRTCRoleAnchor;

    // If your scenario is live streaming, set the application scenario to `TRTC_AP
    // If your application scenario is a group video call, use "TRTC_APP_SCENE_LIVE
    trtc_cloud_->enterRoom(trtcParams, liteav::TRTCAppSceneLIVE);

    // ...other codes
}
```

观众角色：



```
BOOL CTRTCDemo::InitInstance()
```

```
{
    // ...other codes

    // Assemble TRTC room entry parameters. Replace the field values in `TRTCParams
    // Replace each field in TRTCParams with your own parameters
    liteav::TRTCParams trtcParams;
    trtcParams.sdkAppId = 1400000123;  // Replace with your own SDKAppID
    trtcParams.userId = "denny";       // Replace with your own user ID
    trtcParams.roomId = 123321;        // Replace with your own room number
    trtcParams.userSig = "xxx";        // Replace with your own userSig
    trtcParams.role = liteav::TRTCRoleAudience;

    // If your scenario is live streaming, set the application scenario to `TRTC_AP
    // If your application scenario is a group video call, use "TRTC_APP_SCENE_LIVE
    trtc_cloud_->enterRoom(trtcParams, liteav::TRTCAppSceneLIVE);

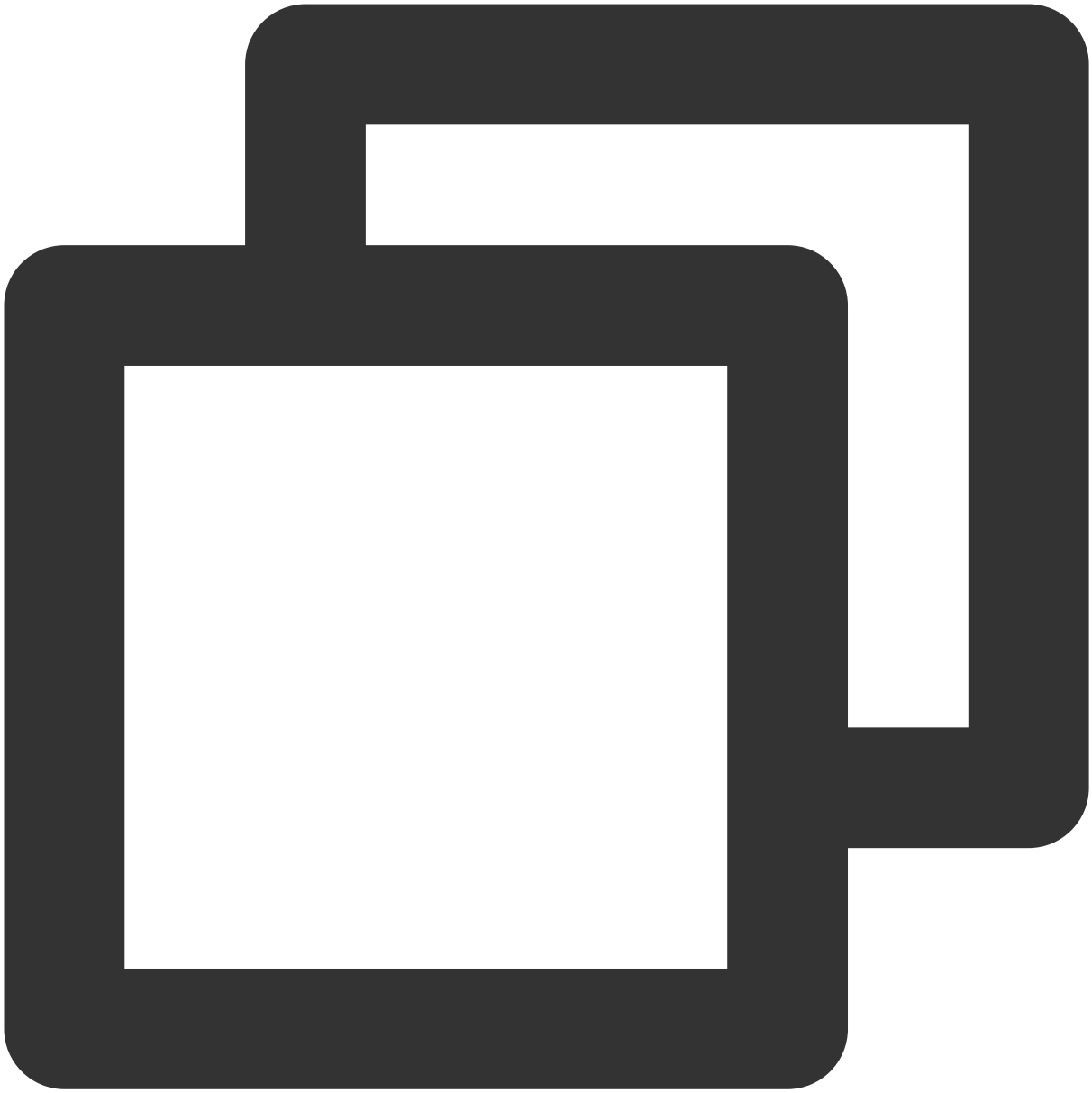
    // ...other codes
}
```

注意：

如果以**观众角色**进入房间，**sdkAppId** 和 **roomId** 需要与主播端相同，而**userId** 和 **userSig** 需要替换为自己的值。

步骤5. 打开摄像头

1. 在资源文件 **IDD_TRTCDEMO_DIALOG** 中，单击左侧边框的 **Toolbox**，拖动 **Picture Control** 到对话框中。
2. 在 **Picture Control** 控件的右键菜单中选择 **Properties**，ID 选择 **AFX_IDC_PICTURE**。
3. 在 `CTRTCDemo::onEnterRoom()` 方法中调用接口 `setLocalRenderParams` 来设置本地预览的渲染参数，然后调用接口 `startLocalPreview` 打开摄像头预览，如以下代码：



```
void CTRTCDemo::onEnterRoom(int result) {
        if (result > 0) {
                // Set the preview mode of the local video image: Enable horizontal
                liteav::TRTCRenderParams render_params;
                render_params.mirrorType = liteav::TRTCVideoMirrorType_Enable;
                render_params.fillMode = TRTCVideoFillMode_Fill;
                trtc_cloud_->setLocalRenderParams(render_params);

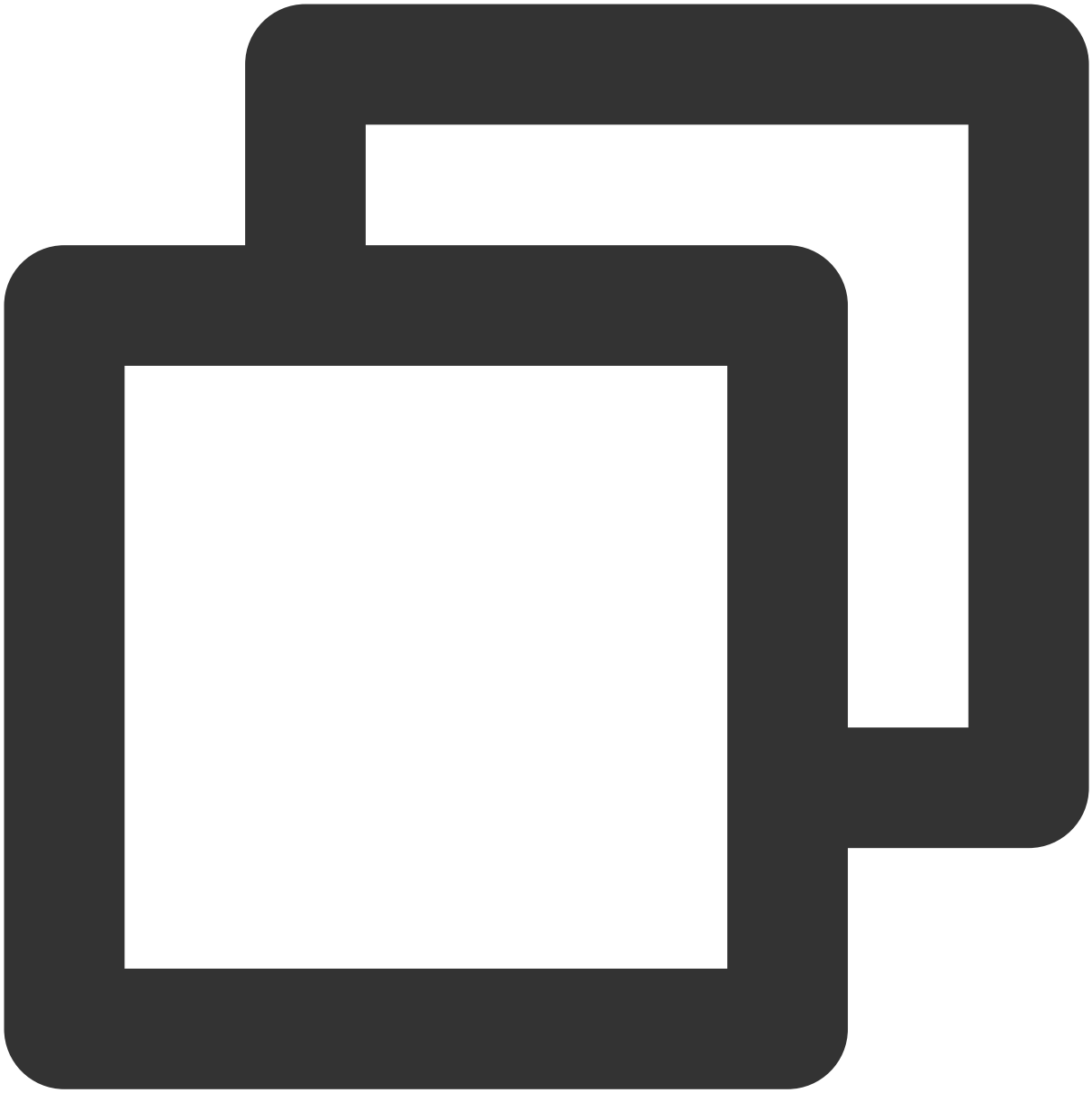
        // Start a preview of the local camera
                CWnd* pLocalVideoView = m_pMainWnd->GetDlgItem(AFX_IDC_PICTURE);
                auto local_view = (liteav::TXView)(pLocalVideoView->GetSafeHwnd());
```

```
                trtc_cloud_->startLocalPreview(local_view);

                printf("Enter room succeed");
        }
        else {
                printf("Enter room failed");
        }
}
```

步骤6. 打开麦克风

您可以调用 `startLocalAudio` 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式，建议根据您的需求**选择以下其中一个适合您项目的模式**。



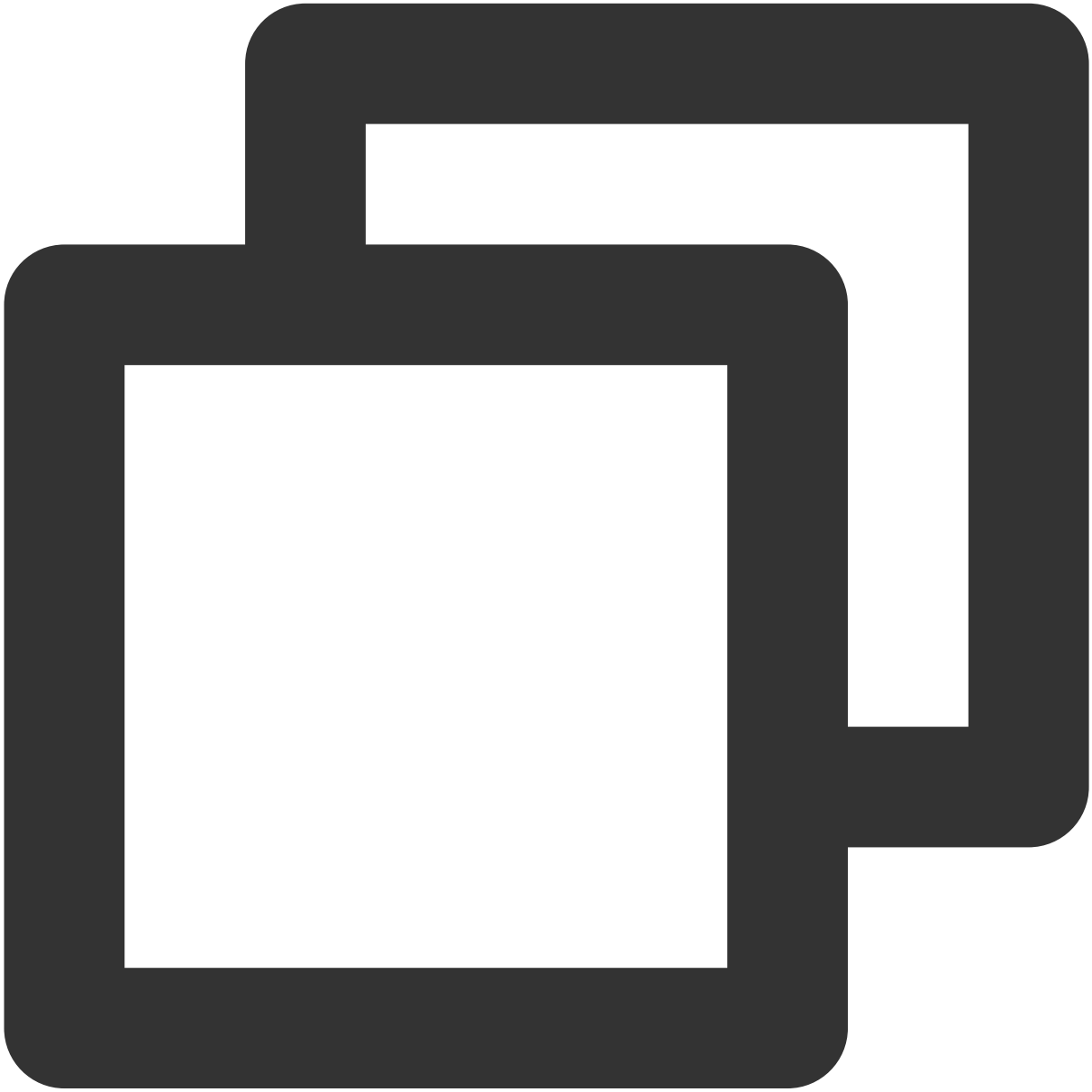
```
// Enable microphone acquisition and set the current scene to: Voice mode
// For high noise suppression capability, strong and weak network resistance
trtc_cloud_->startLocalAudio(TRTCAudioQualitySpeech);

// Enable microphone acquisition, and set the current scene to: Music mode
// For high fidelity acquisition, low sound quality loss, recommended to use with p
trtc_cloud_->startLocalAudio(TRTCAudioQualityMusic);
```

版权所有：腾讯云计算（北京）有限责任公司

步骤7. 播放/停止视频流

1. 创建一个新项目，按照**步骤1-4**以**观众角色**进入 **denny** 的房间。
2. 在资源文件 **IDD_TRTCDEMO_DIALOG** 中添加 **Picture Control** 控件，并将 **ID** 选择为 **AFX_IDC_PICTURE**。
3. 在 `CTRTCDemo::onEnterRoom()` 方法中调用接口 `startRemoteView` 来播放远端用户的视频画面。

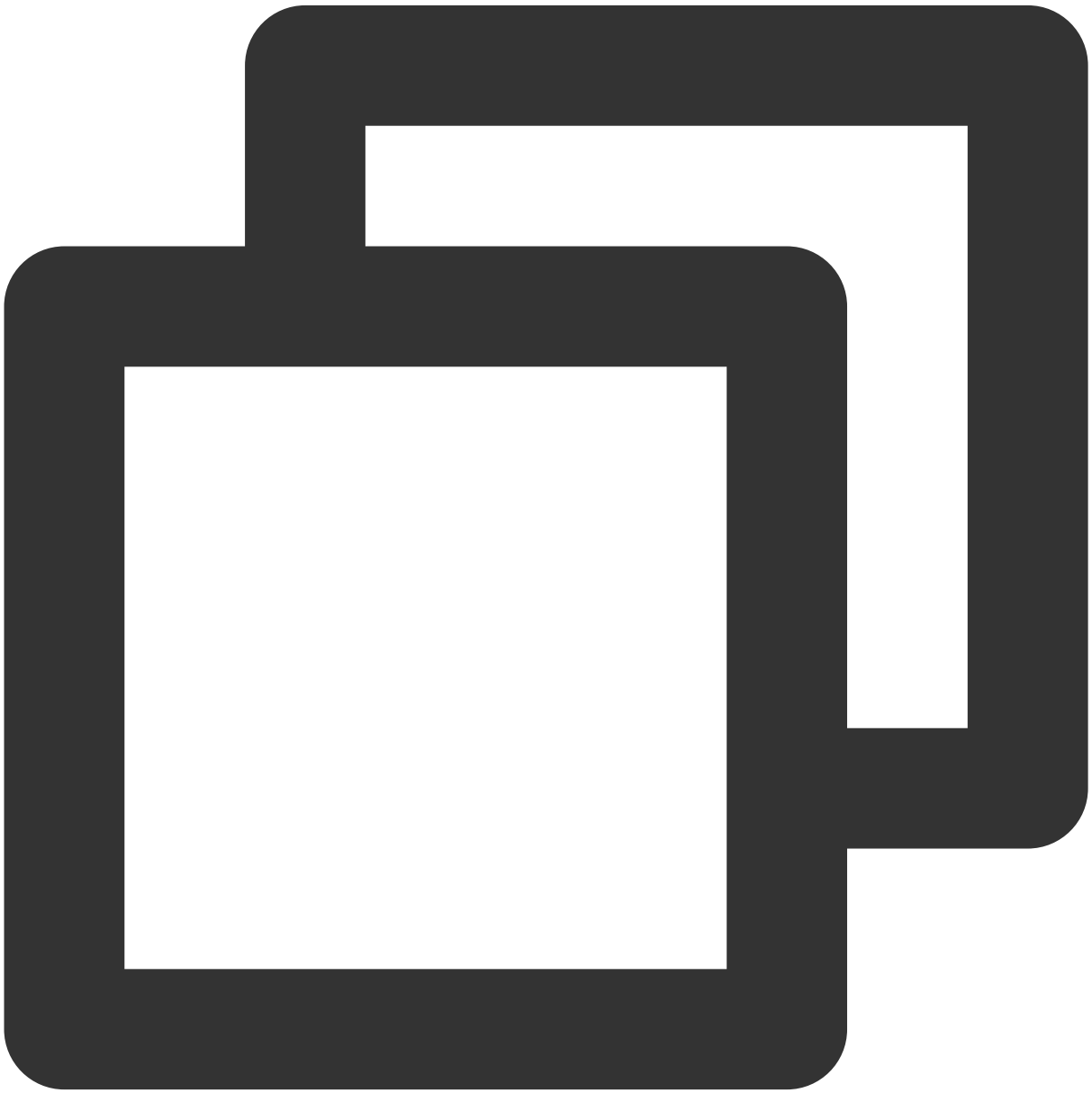


```
if (result > 0) {
    // Start a preview of the local camera
    CWnd* pVideoView = m_pMainWnd->GetDlgItem(AFX_IDC_PICTURE);
    auto video_view = (liteav::TXView)(pVideoView->GetSafeHwnd());
```

```
    // Play denny's camera footage
    trtc_cloud_->startRemoteView("denny", liteav::TRTCVideoStreamTypeBig, video_vie

    printf("Enter room succeed");
}
```

之后，您可以调用 `stopRemoteView` 接口停止某个远端用户的视频，也可以通过接口 `stopAllRemoteView` 停止播放所有远端用户的视频。



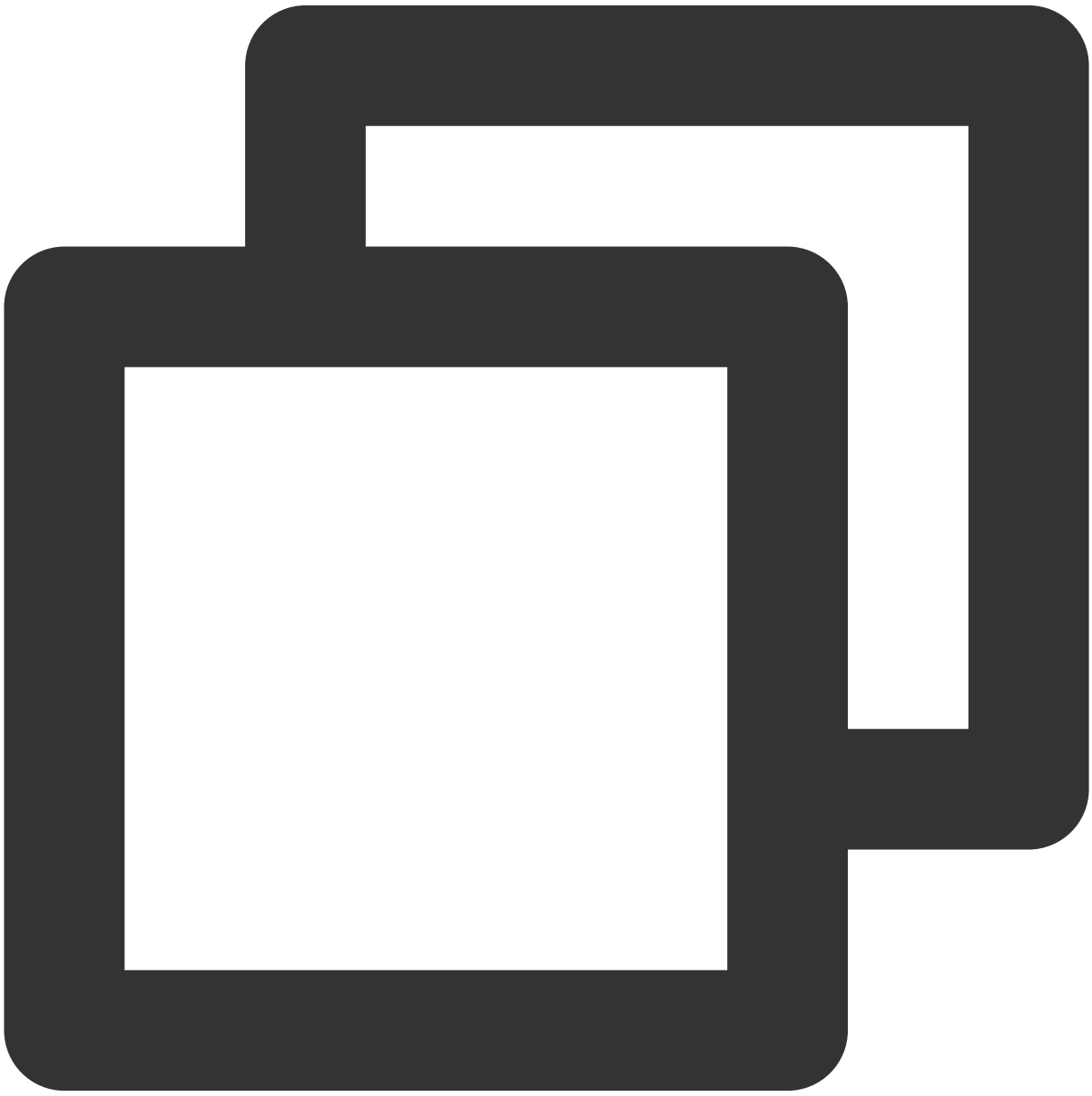
```
// Stop denny's camera footage
```

```
trtc_cloud_->stopRemoteView("denny", liteav::TRTCVideoStreamTypeBig);

// Stop all camera footages
trtc_cloud_->stopAllRemoteView();
```

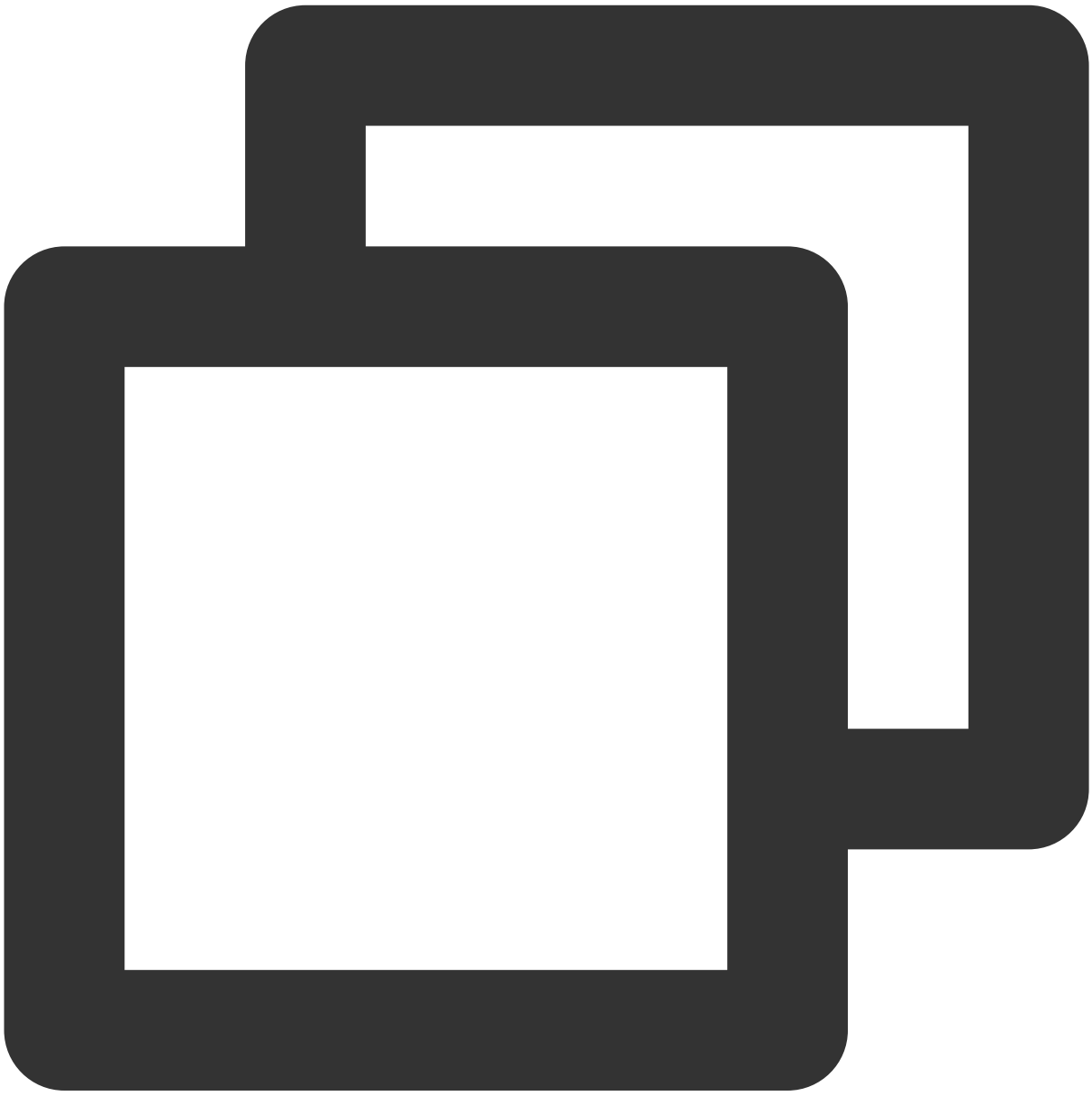
步骤8. 播放/停止音频流

您可以通过调用接口 `muteRemoteAudio("denny", true)` 来静音远端用户 **denny** 的声音。



```
// Mute user with id denny
trtc_cloud_->muteRemoteAudio("denny", true);
```

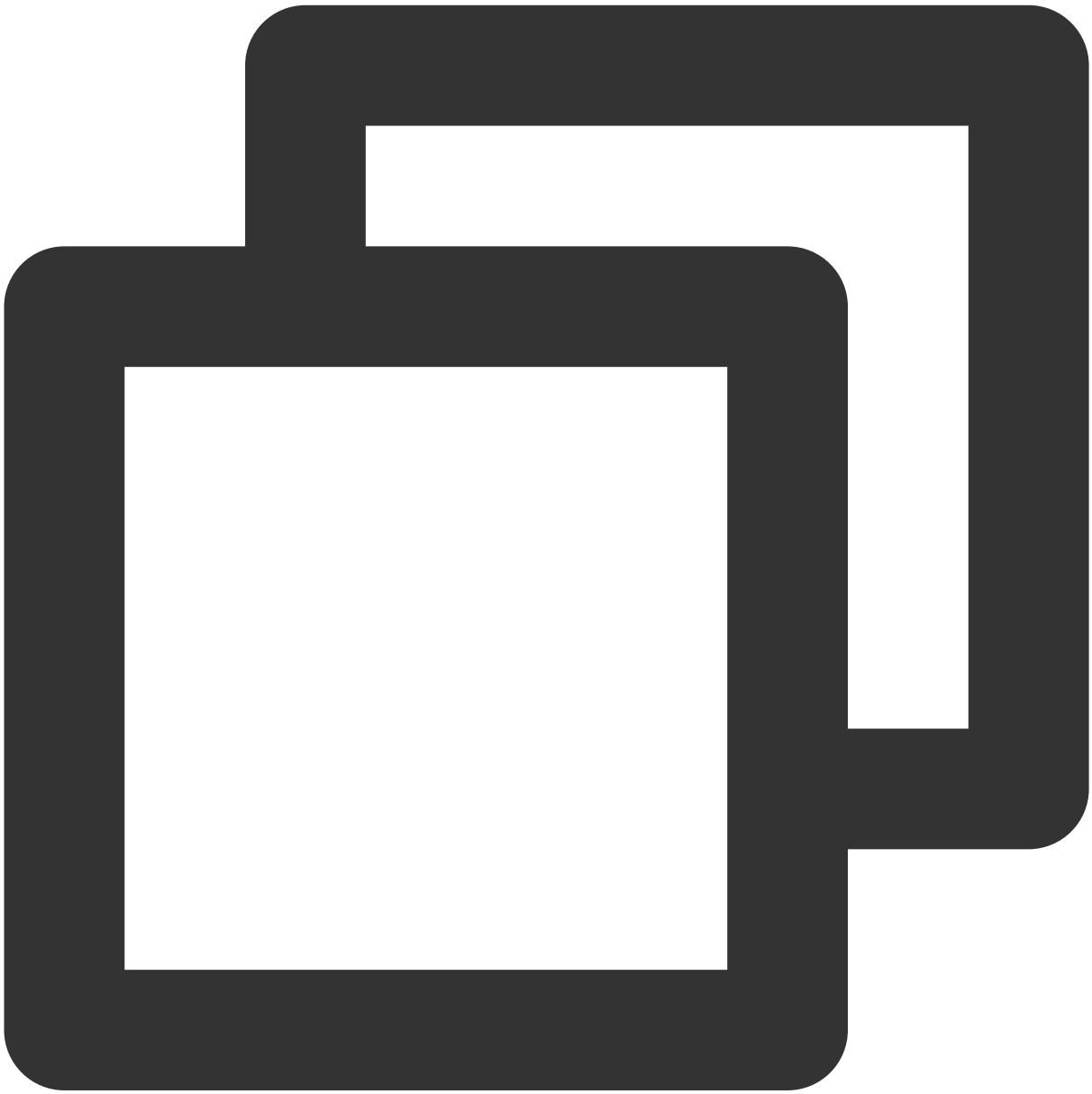
之后，您可以通过调用接口 `muteRemoteAudio("denny", false)` 来解除对他的静音。



```
// Unmute user with id denny
trtc_cloud_->muteRemoteAudio("denny", false);
```

步骤9. 退出房间

您可以调用接口 `exitRoom` 即可退出当前的房间，SDK 会在退房结束后通过 `onExitRoom(int reason)` 回调事件通知您。



```
// Exit current room
trtc_cloud_->exitRoom();
```

常见问题

您可以在API 参考查看所有函数列表及其描述。
如果您的接入和使用中遇到问题，请参见 常见问题。

联系我们

如果有任何需要或者反馈，您可以联系：info_rtc@tencent.com。

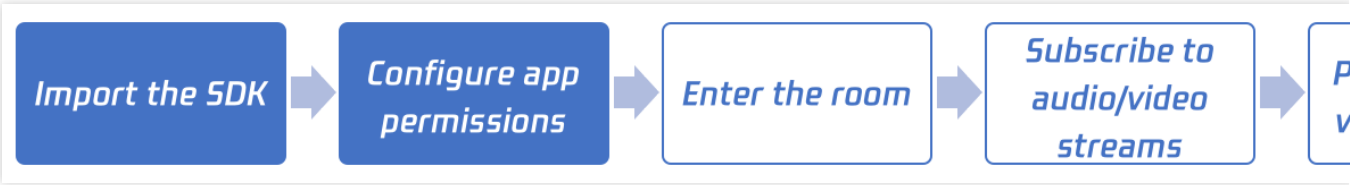
版权所有：腾讯云计算（北京）有限责任公司

01. 导入 SDK 到项目中

iOS

最近更新时间：2024-06-07 11:20:49

本文介绍如何将 SDK 导入到您的项目中：



开发环境要求

Xcode 9.0+。
iOS 9.0 以上的 iPhone 或者 iPad 真机。
项目已配置有效的开发者签名。

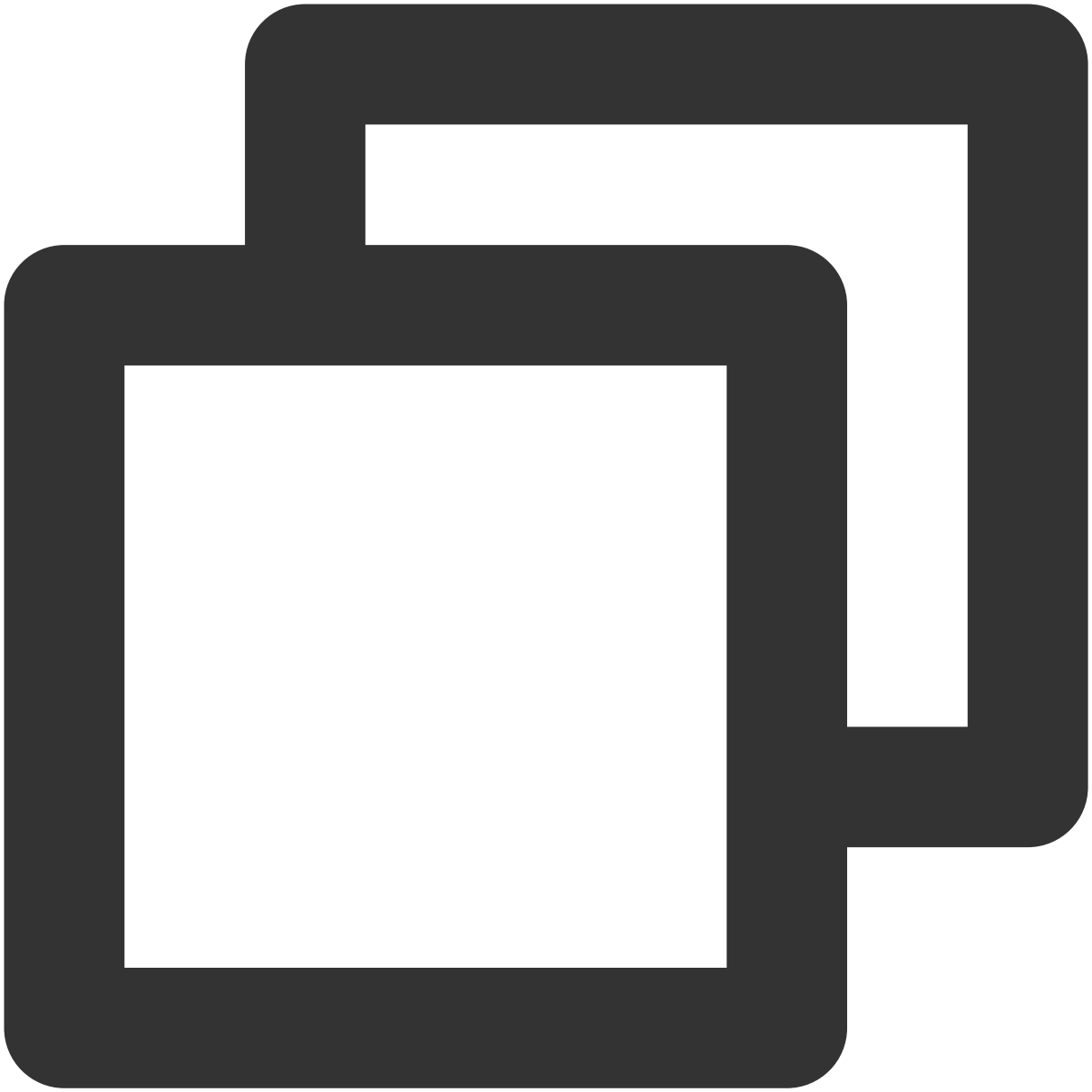
第一步：导入 SDK

您可以选择使用 CocoaPods 方案，或者先将 SDK 下载到本地，再将其手动导入到您当前的项目中。

方案一：使用 CocoaPods

1. 安装 CocoaPods

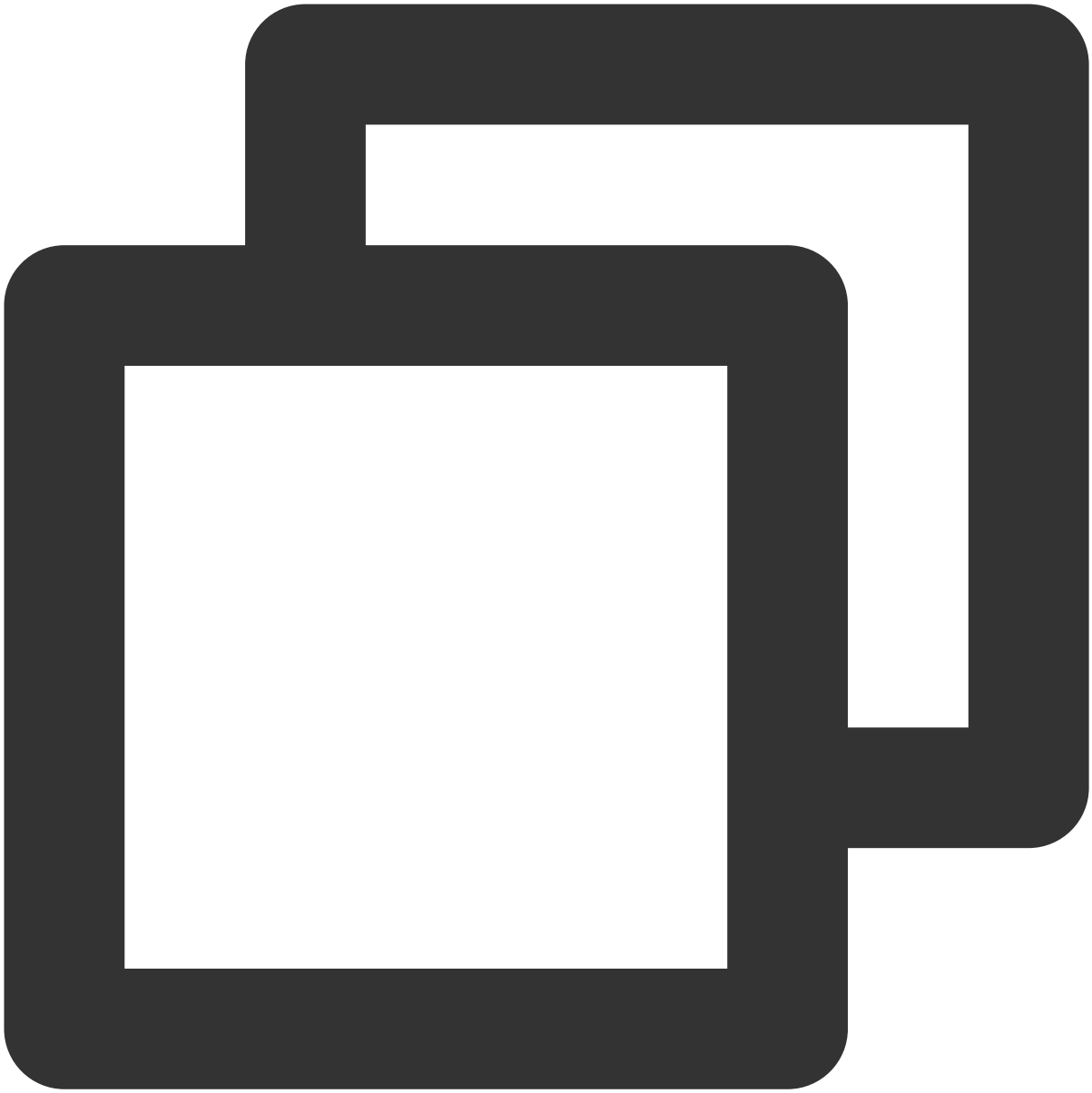
在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：



```
sudo gem install cocoapods
```

2. 创建 Podfile 文件

进入项目所在路径，输入以下命令行之后项目路径下会出现一个 Podfile 文件。



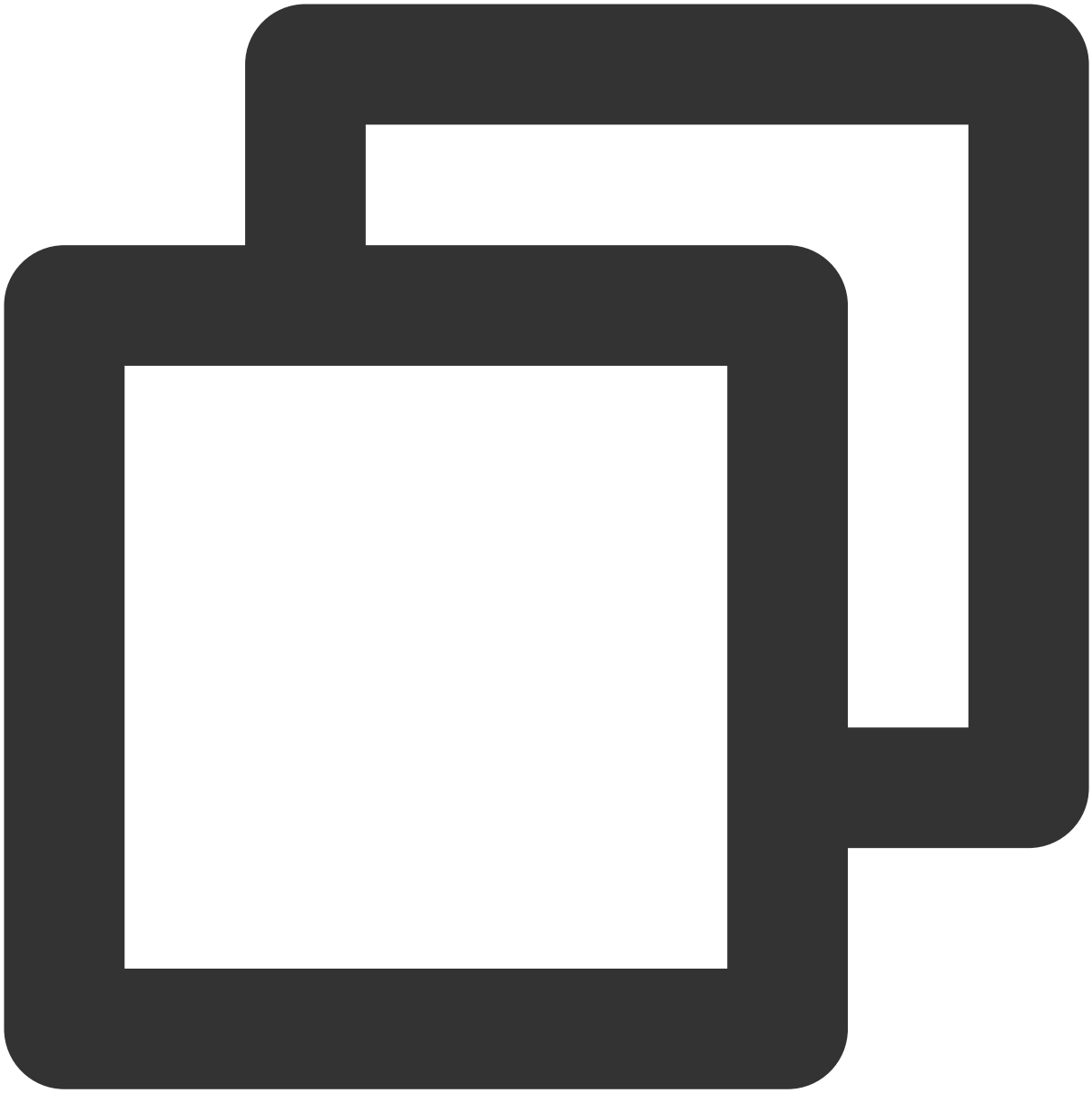
```
pod init
```

3. **编辑 Podfile 文件**

根据您的项目需要选择合适的版本，并编辑 Podfile 文件：

选项一：TRTC 精简版

安装包体积增量最小，但仅支持实时音视频（TRTC）和 直播播放器（TXLivePlayer）两项功能。如选择此版本，请按如下方式编辑 Podfile 文件：

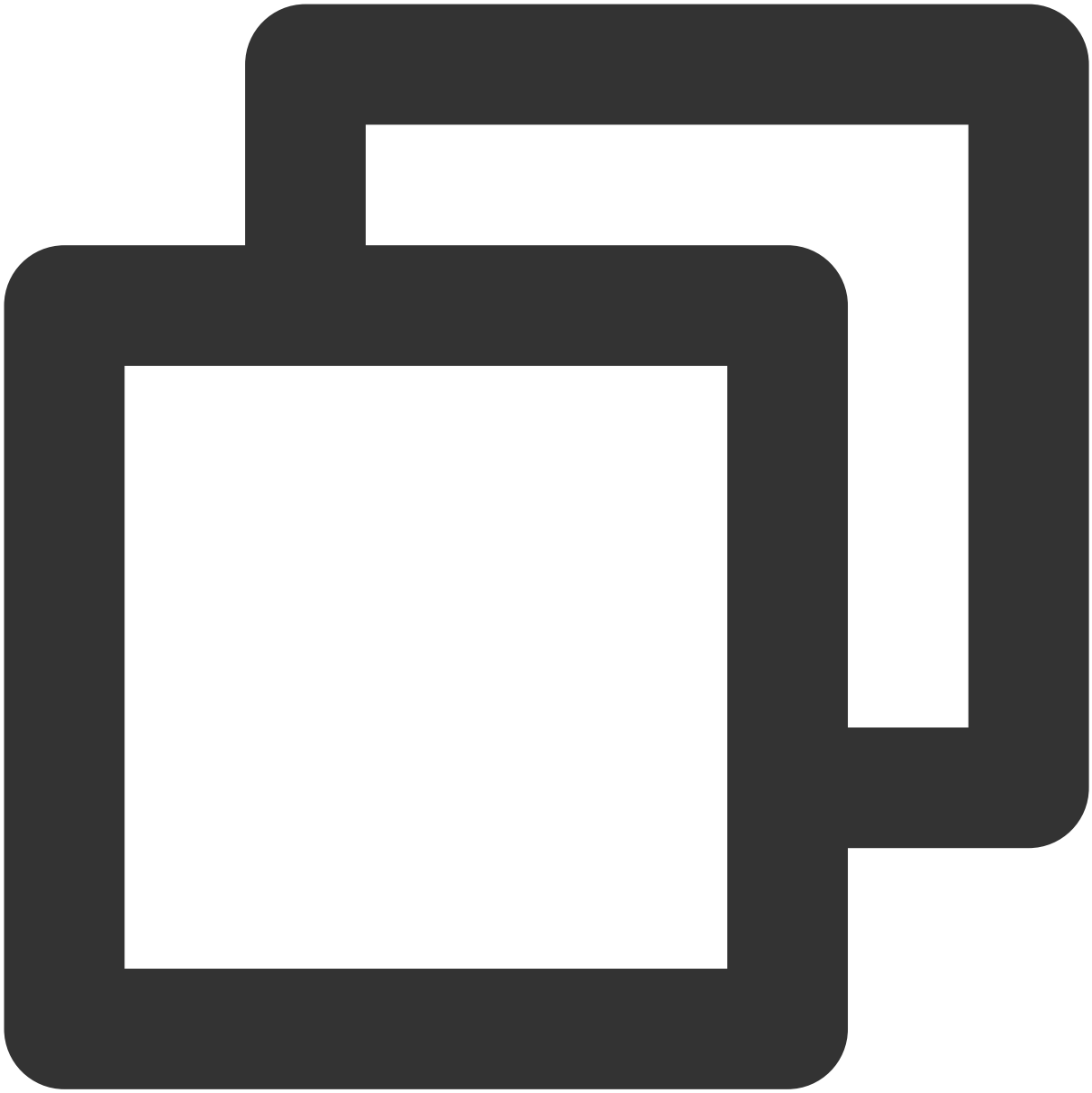


```
platform :ios, '8.0'

 target 'App' do
 pod 'TXLiteAVSDK_TRTC', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsdks
 end
```

选项二：Professional 专业版

包含实时音视频（TRTC）、直播播放器（TXLivePlayer）、RTMP 推流（TXLivePusher）、点播播放器（TXVodPlayer）和短视频录制和编辑（UGSV）等众多功能。如选择此版本，请按如下方式编辑 Podfile 文件：

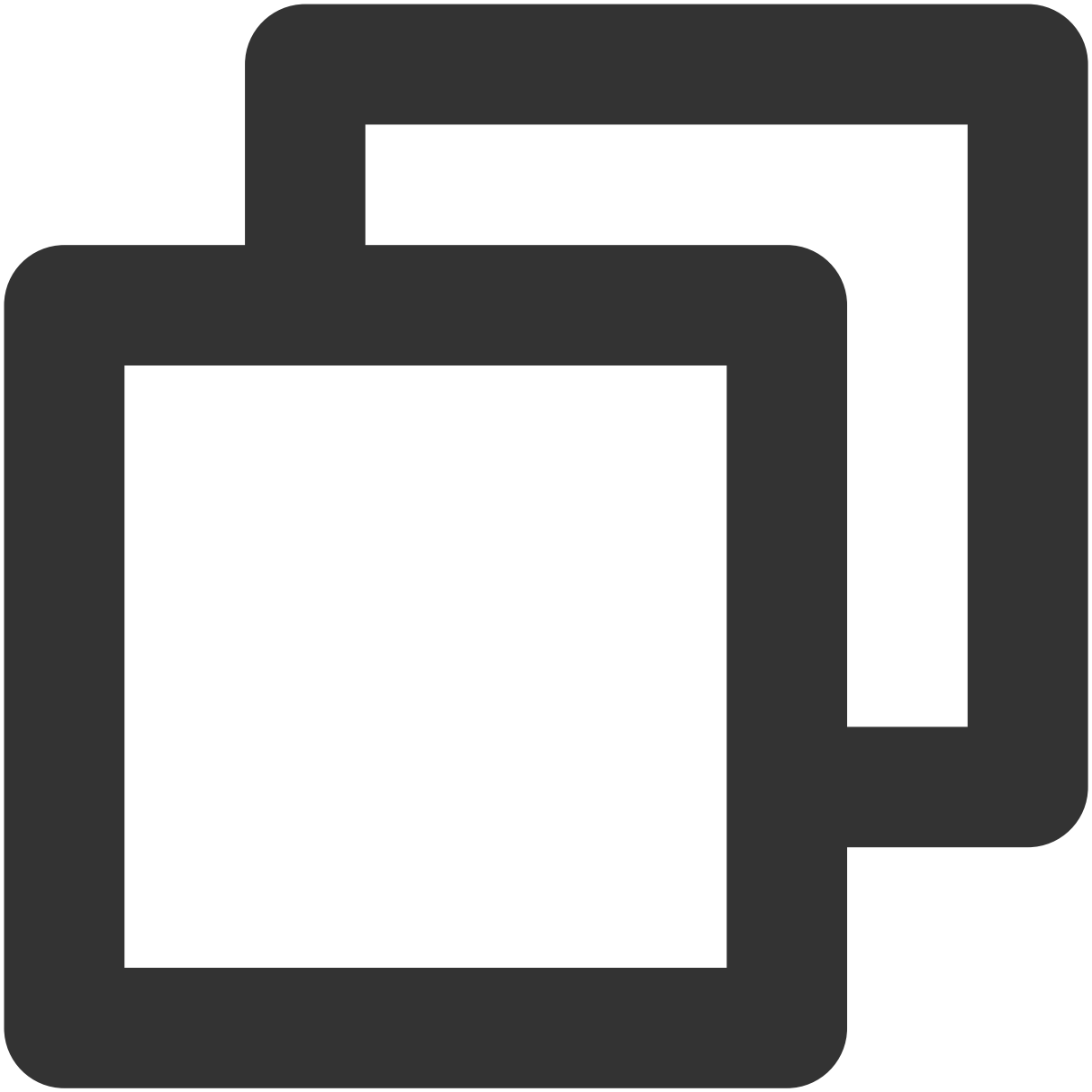


```
platform :ios, '8.0'

 target 'App' do
 pod 'TXLiteAVSDK_Professional', :podspec => 'https://liteav.sdk.qcloud.com/pod/li
 end
```

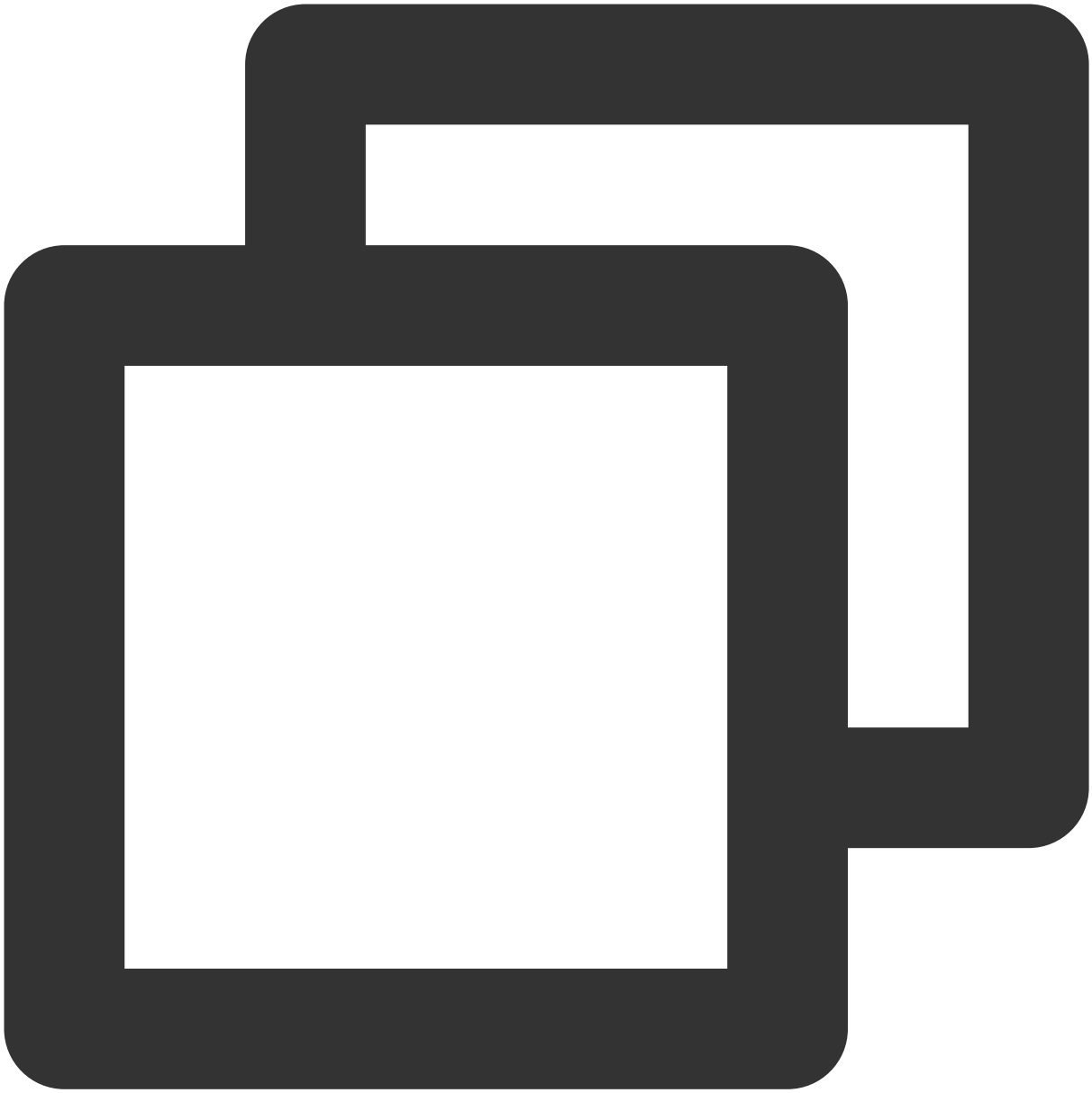
4. **更新并安装 SDK。**

在终端窗口中输入如下命令以更新本地库文件，并安装 SDK：



```
pod install
```

或使用以下命令更新本地库版本：

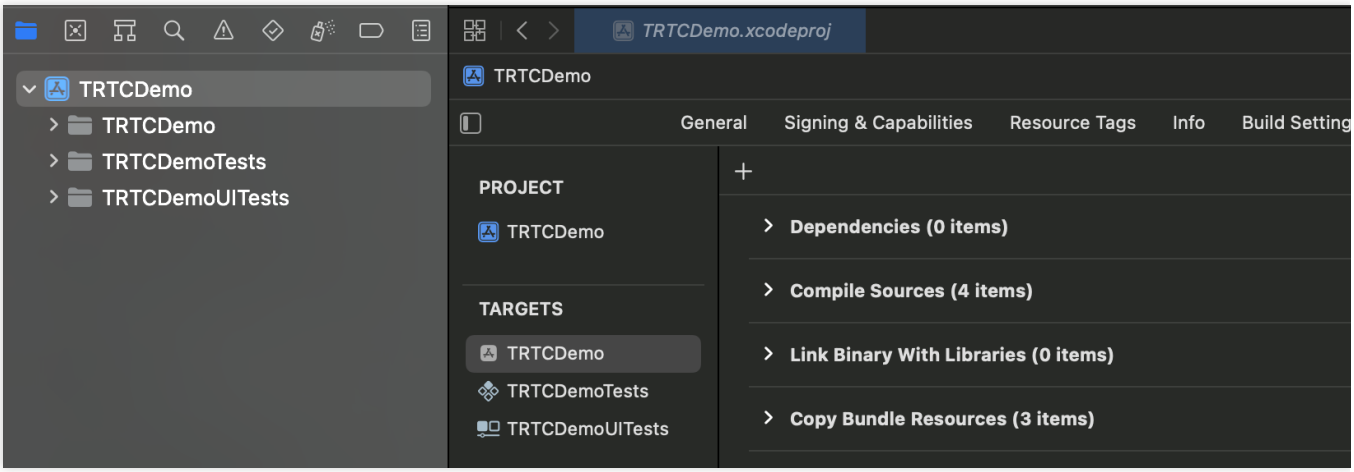


```
pod update
```

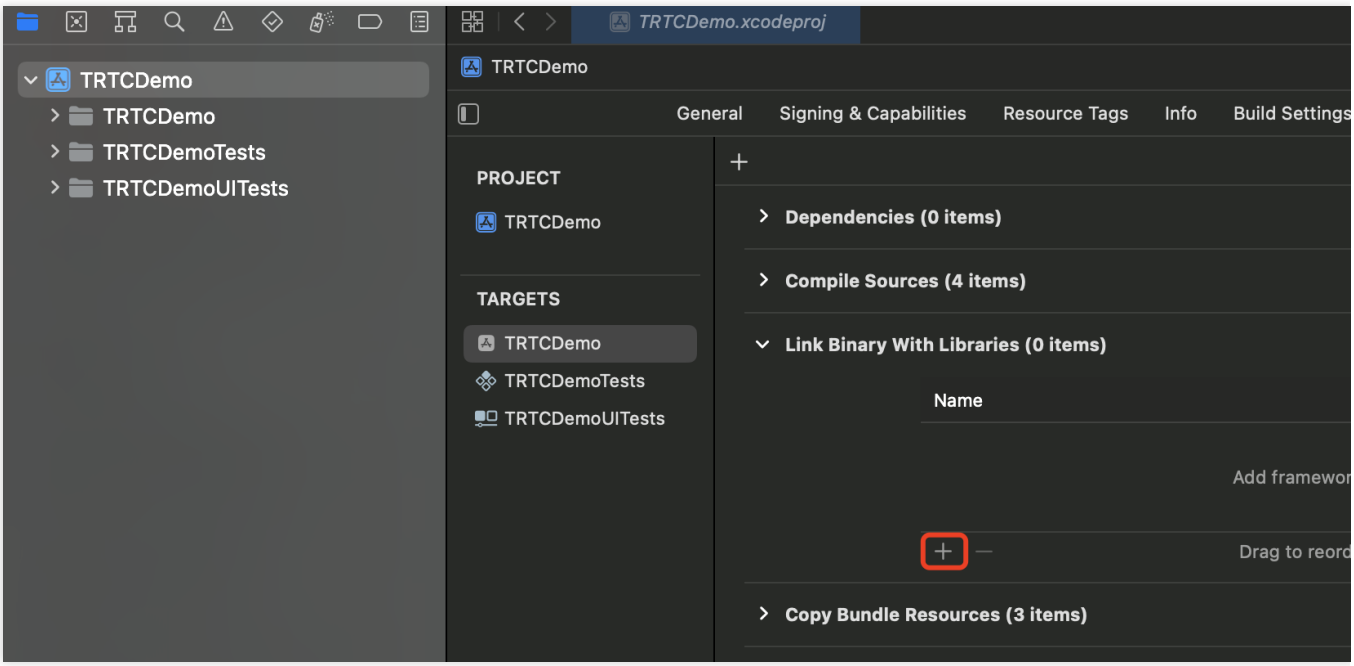
pod 命令执行完后，会生成集成了 SDK 的 .xcworkspace 后缀的工程文件，双击打开即可。

方案二：下载 SDK 并手动导入

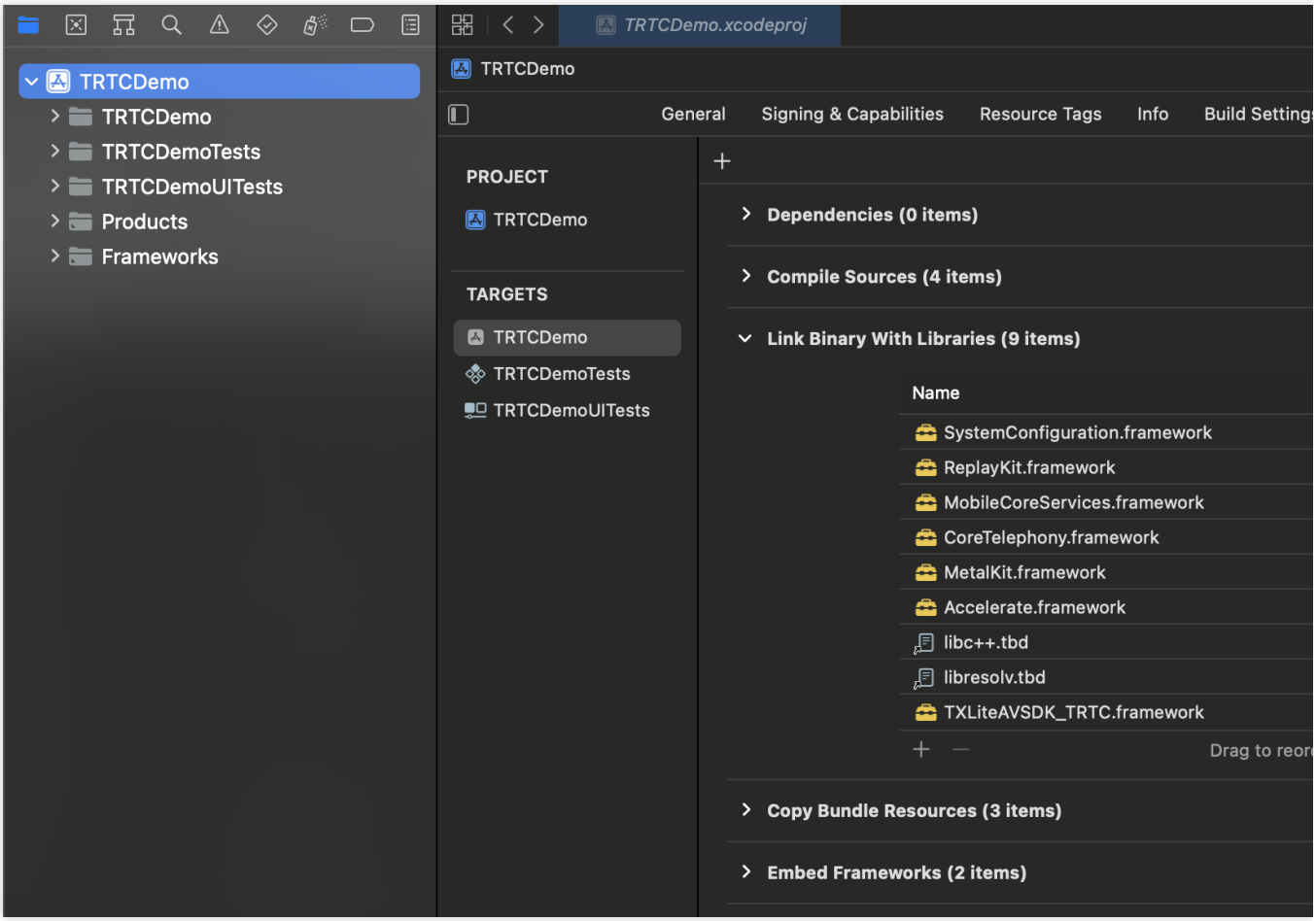
1. 下载 SDK 并解压到本地。
2. 打开您的 Xcode 工程项目，选择要运行的 target , 选中 **Build Phases** 项。



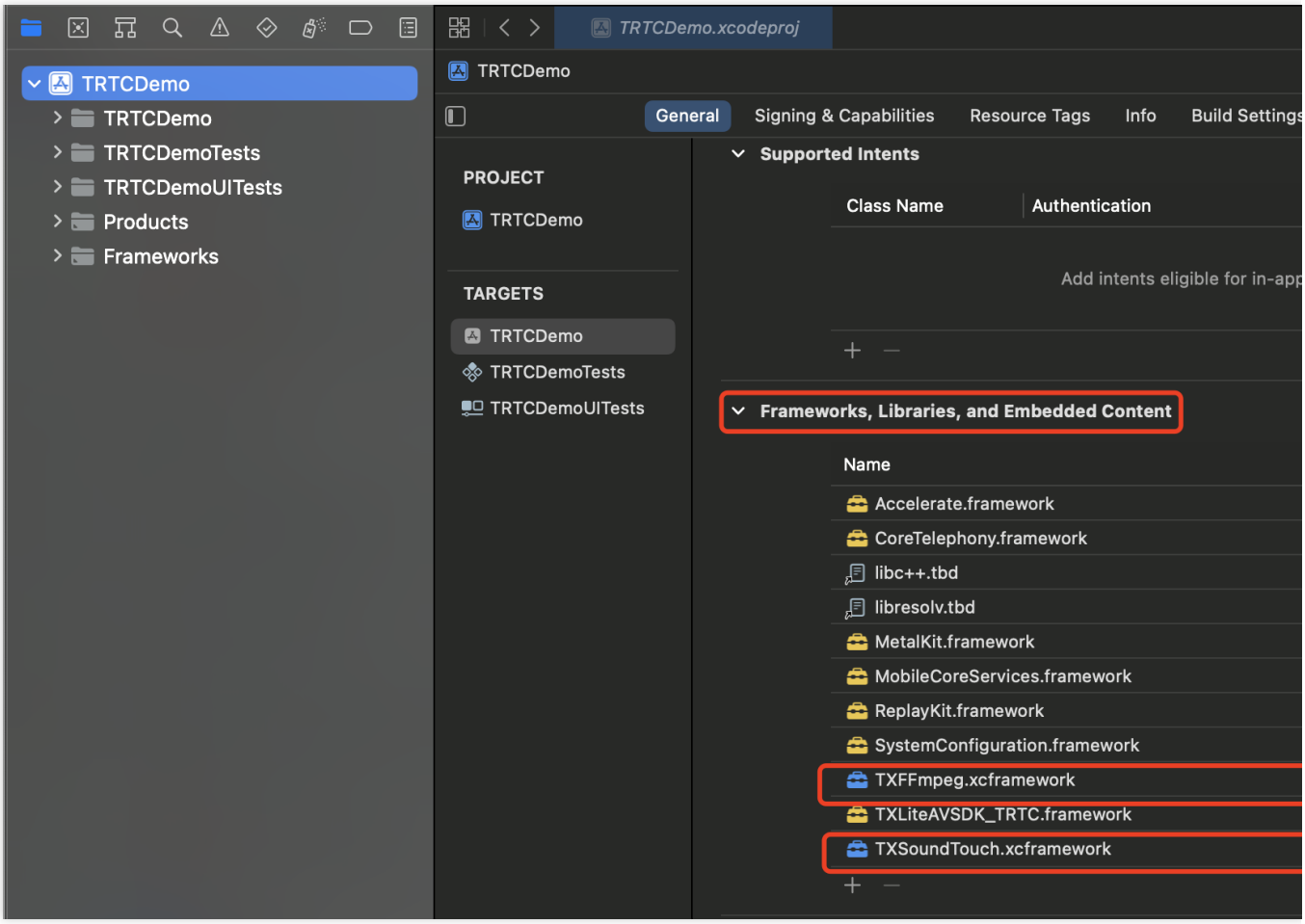
3. 单击 **Link Binary with Libraries** 项展开，单击底下的“+”号图标去添加依赖库。



4. 依次添加下载的 `TXLiteAVSDK_TRTC.Framework`（或者 `TXLiteAVSDK_Professional.Framework`）、`TXFFmpeg.xcframework`、`TXSoundTouch.xcframework`、及其所需依赖库 `GLKit.framework`、`AssetsLibrary.framework`、`SystemConfiguration.framework`、`libsqlite3.0.tbd`、`CoreTelephony.framework`、`AVFoundation.framework`、`OpenGLES.framework`、`Accelerate.framework`、`MetalKit.framework`、`libresolv.tbd`、`MobileCoreServices.framework`、`libc++.tbd`、`CoreMedia.framework`。



5. 单击 **General**，选择 **Frameworks,Libraries,and Embedded Content**，检查 TXLiteAVSDK_TRTC.framework 所需要动态库 **TXFFmpeg.xcframework**、**TXSoundTouch.xcframework**是否已经添加，是否正确选择选择 **Embed & Sign**，如果没有单击底下的 **+** 号图标依次添加。

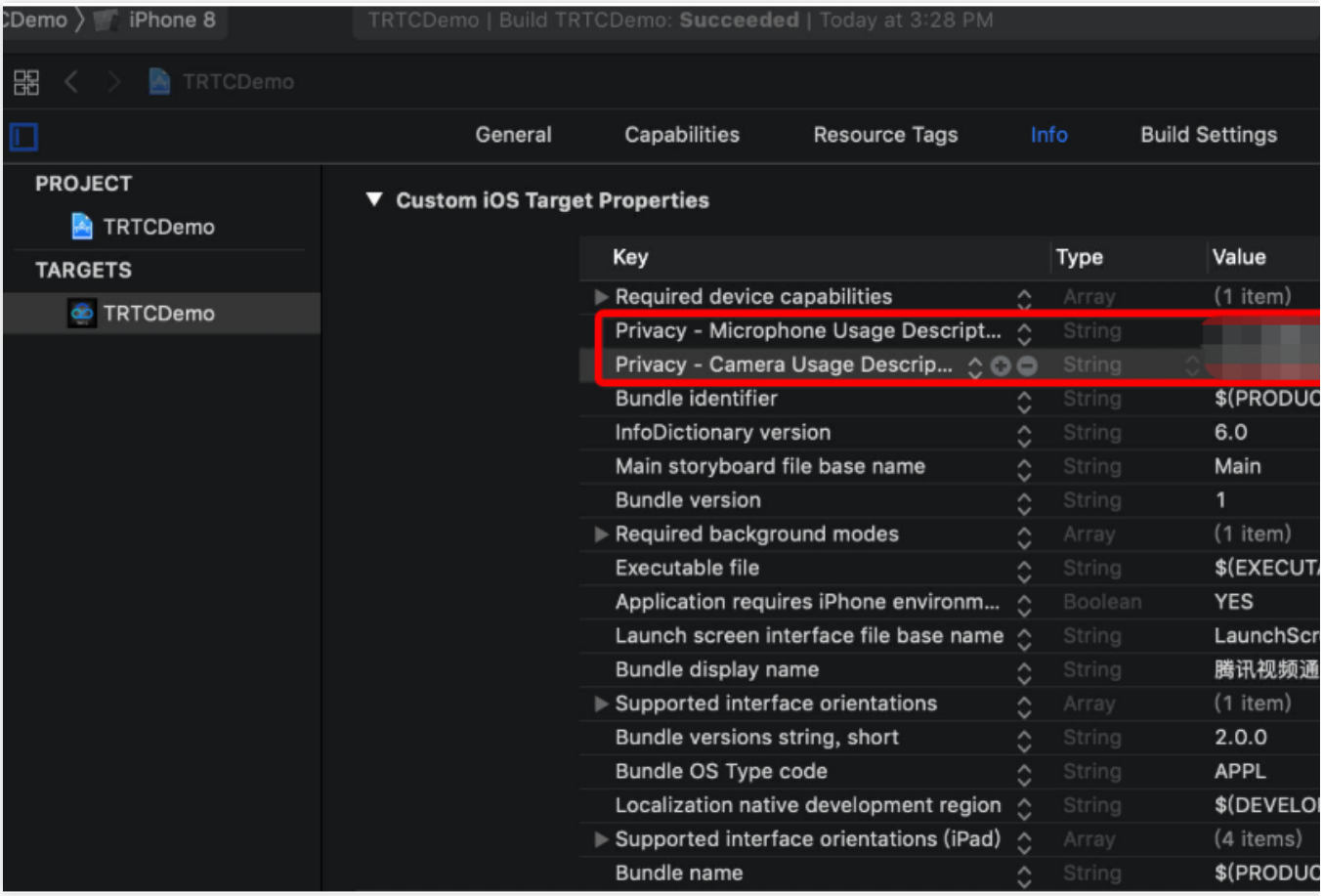


第二步：配置 App 权限

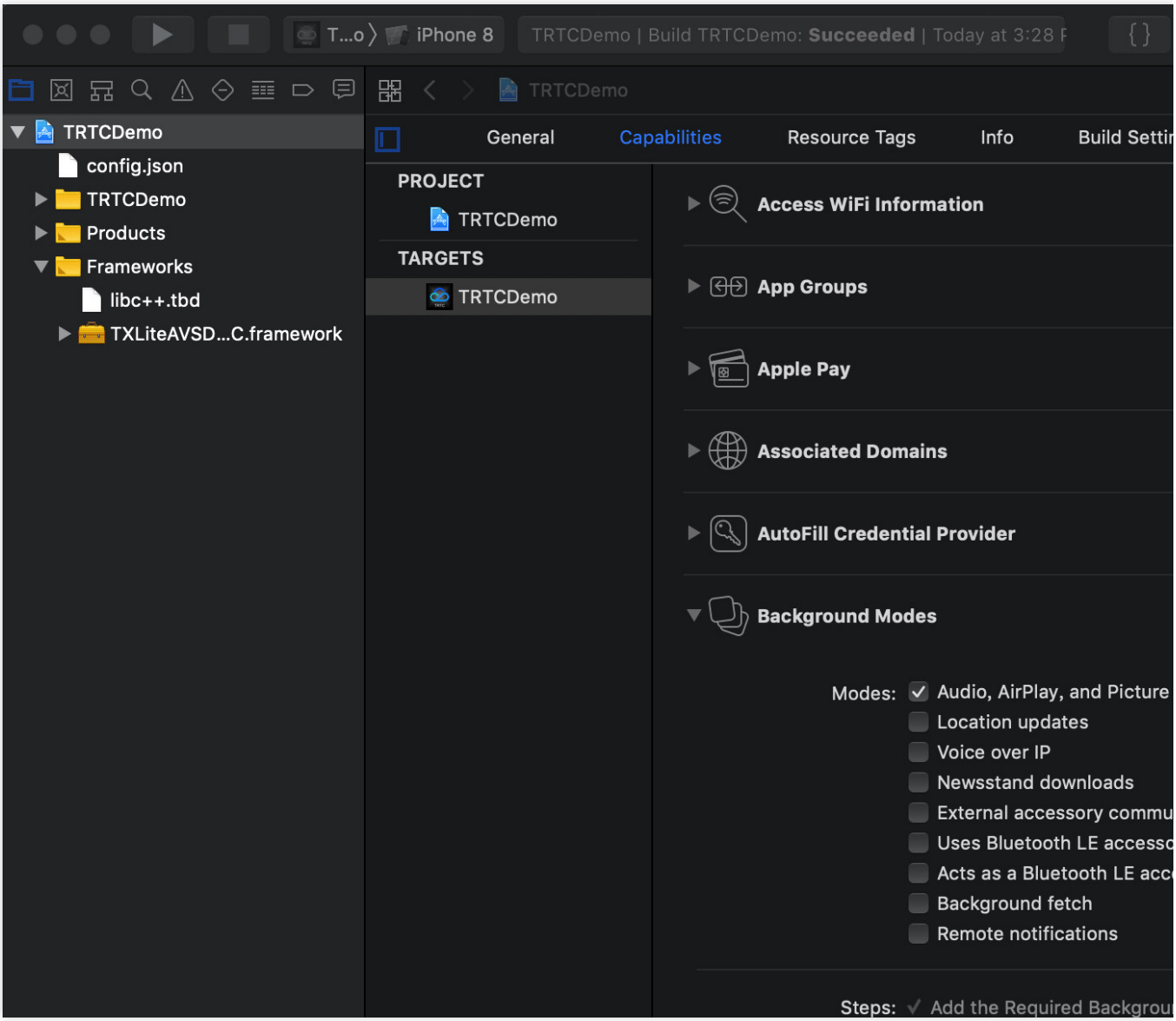
1. 如需使用 SDK 提供的音视频功能，需要给 App 授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

Privacy - Microphone Usage Description，并填入麦克风使用目的提示语。

Privacy - Camera Usage Description，并填入摄像头使用目的提示语。



2. 如需 App 进入后台仍然运行相关功能，可在 XCode 中选中当前工程项目，并在 **Capabilities** 下将设置项 **Background Modes** 设定为 **ON**，并勾选 **Audio，AirPlay and Picture in Picture** ，如下图所示：



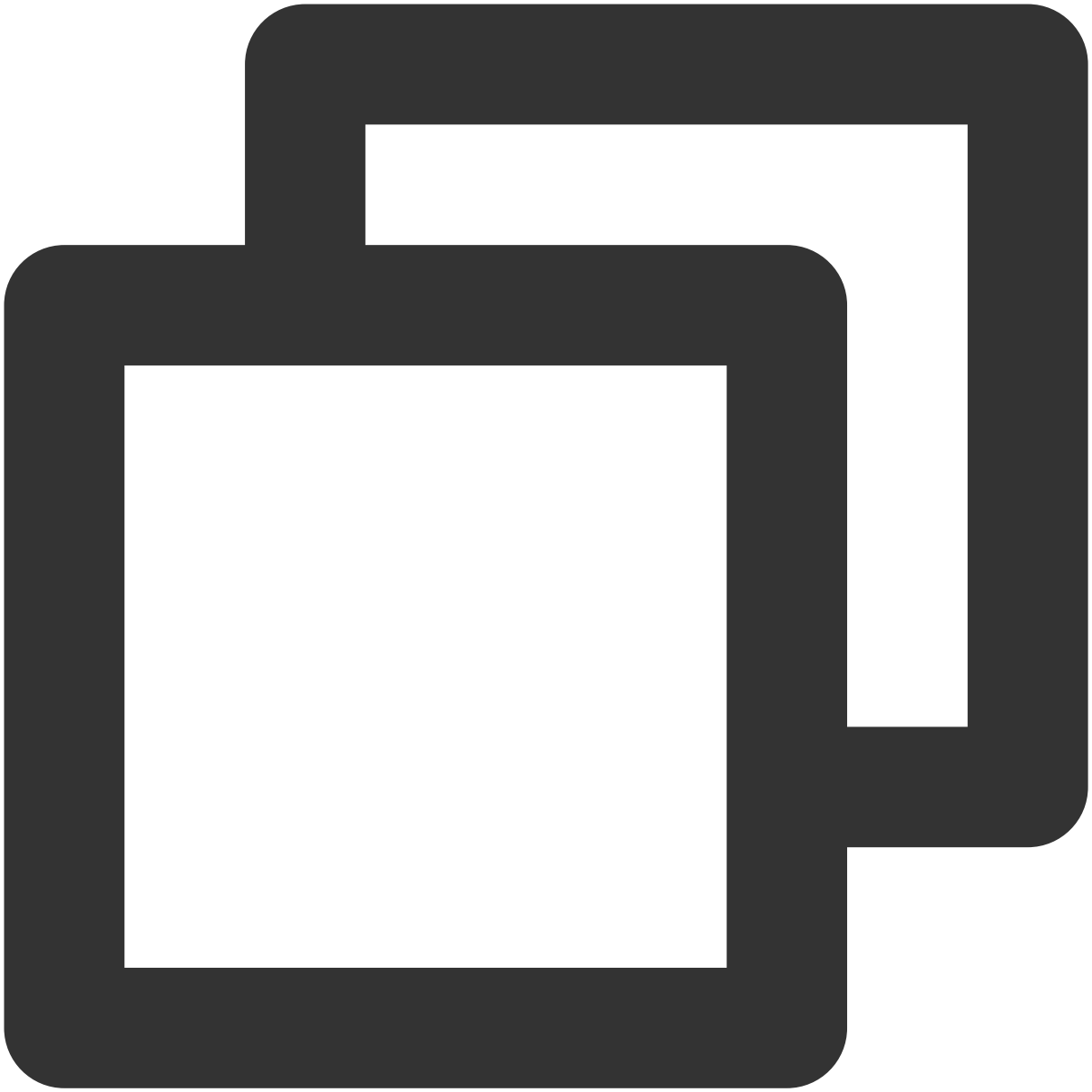
第三步：在项目中引用 SDK

在完成了第一步的导入和第二步的设备权限授权后，就可以在项目中引用 SDK 中提供的接口 API 了。

通过 Objective-C 或 Swift 接口引用

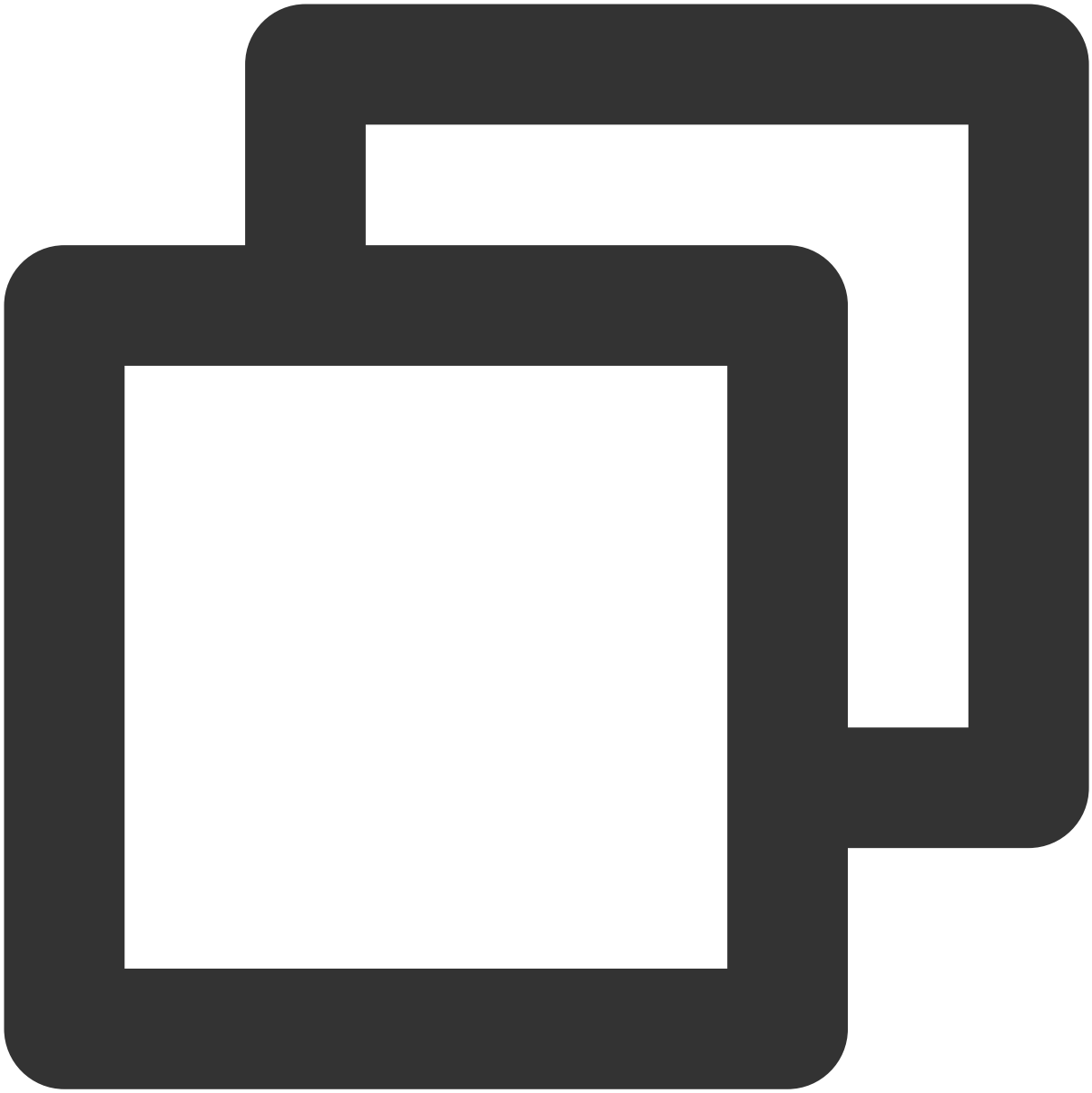
在 Objective-C 或 Swift 代码中使用 SDK 有两种方式：

模块引用：在项目需要使用 SDK API 的文件里，添加模块引用。



```
@import TXLiteAVSDK_TRTC;
```

头文件引用：在项目需要使用 SDK API 的文件里，引入具体的头文件。



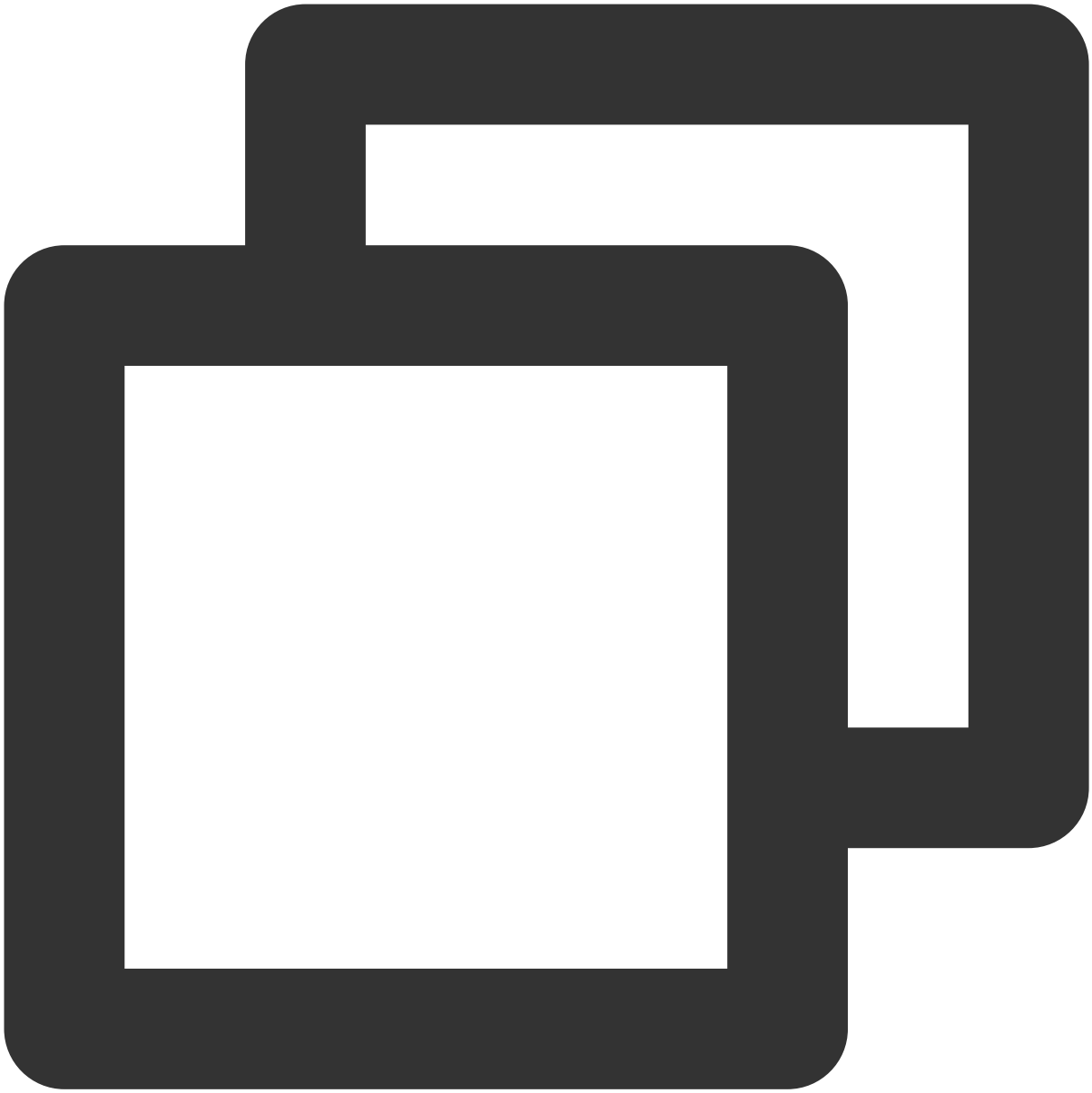
```
#import "TXLiteAVSDK_TRTC/TRTCCloud.h"
```

说明：

对于 Objective-C 接口的使用方式，请参见 iOS&Mac API 概览。

通过 C++ 接口引用(可选)

如果您的项目是通过 QT 或者 Electron 这样的跨平台框架引入 SDK，请引用 `TXLiteAVSDK_TRTC.framework/Headers/cpp_interface` 目录下的头文件：



```
#include "TXLiteAVSDK_TRTC/cpp_interface/ITRTCCloud.h"
```

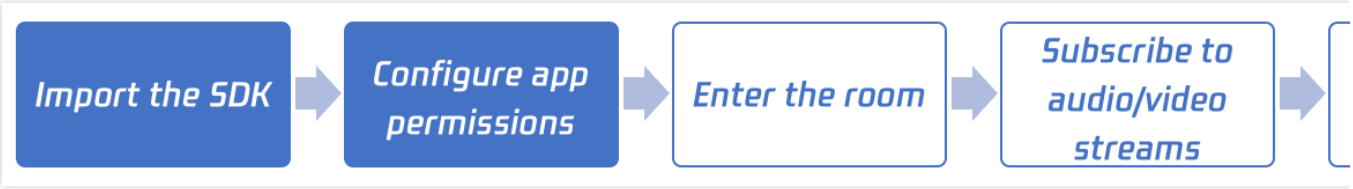
说明：

对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

Android

最近更新时间：2024-06-07 11:20:49

本文介绍如何将 SDK 导入到您的项目中：



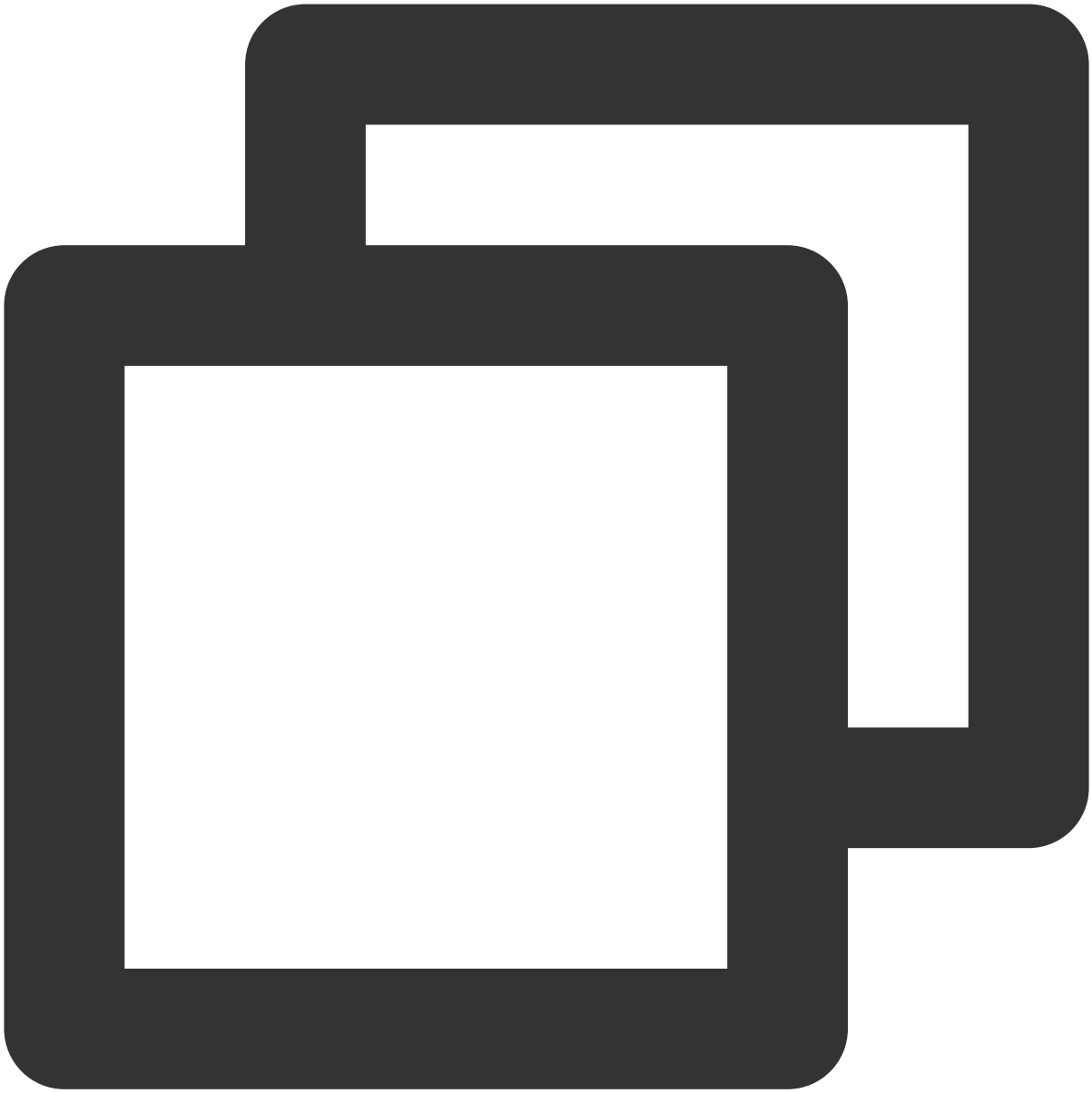
开发环境要求

Android Studio 3.5+。
Android 4.1（SDK API 16）及以上系统。

第一步：导入SDK

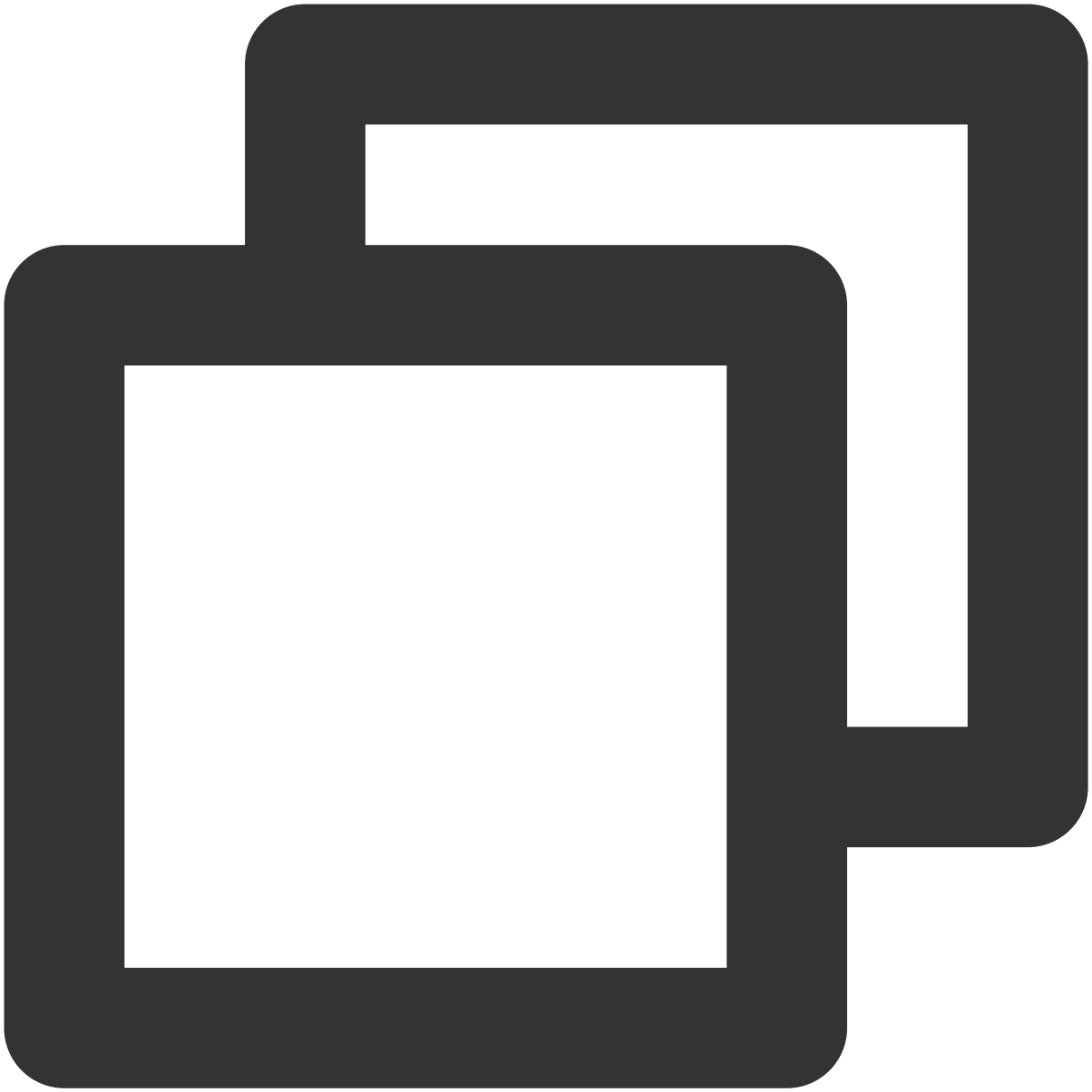
方案一：自动加载（aar）

TRTC SDK 已经发布到 mavenCentral 库，您可以通过配置 gradle 自动下载更新。
1. 在 dependencies 中添加 TRTCSDK 的依赖。



```
dependencies {
     implementation 'com.tencent.liteav:LiteAVSDK_TRTC:latest.release'
}
```

2. 在 defaultConfig 中，指定 App 使用的 CPU 架构。



```
defaultConfig {
     ndk {
             abiFilters "armeabi-v7a", "arm64-v8a"
     }
}
```

说明：

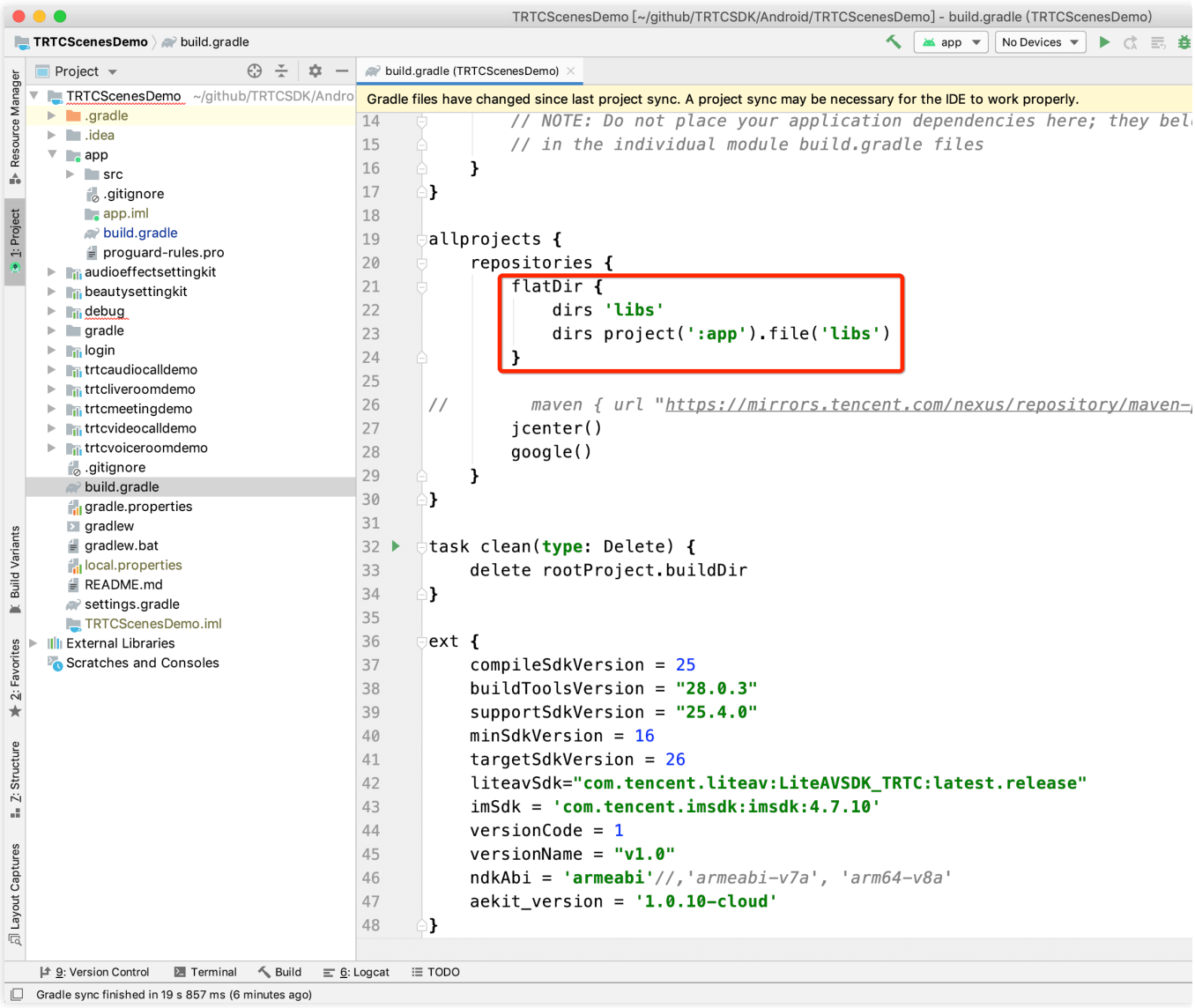
目前 TRTC SDK 支持 armeabi-v7a 和 arm64-v8a。

3. 单击

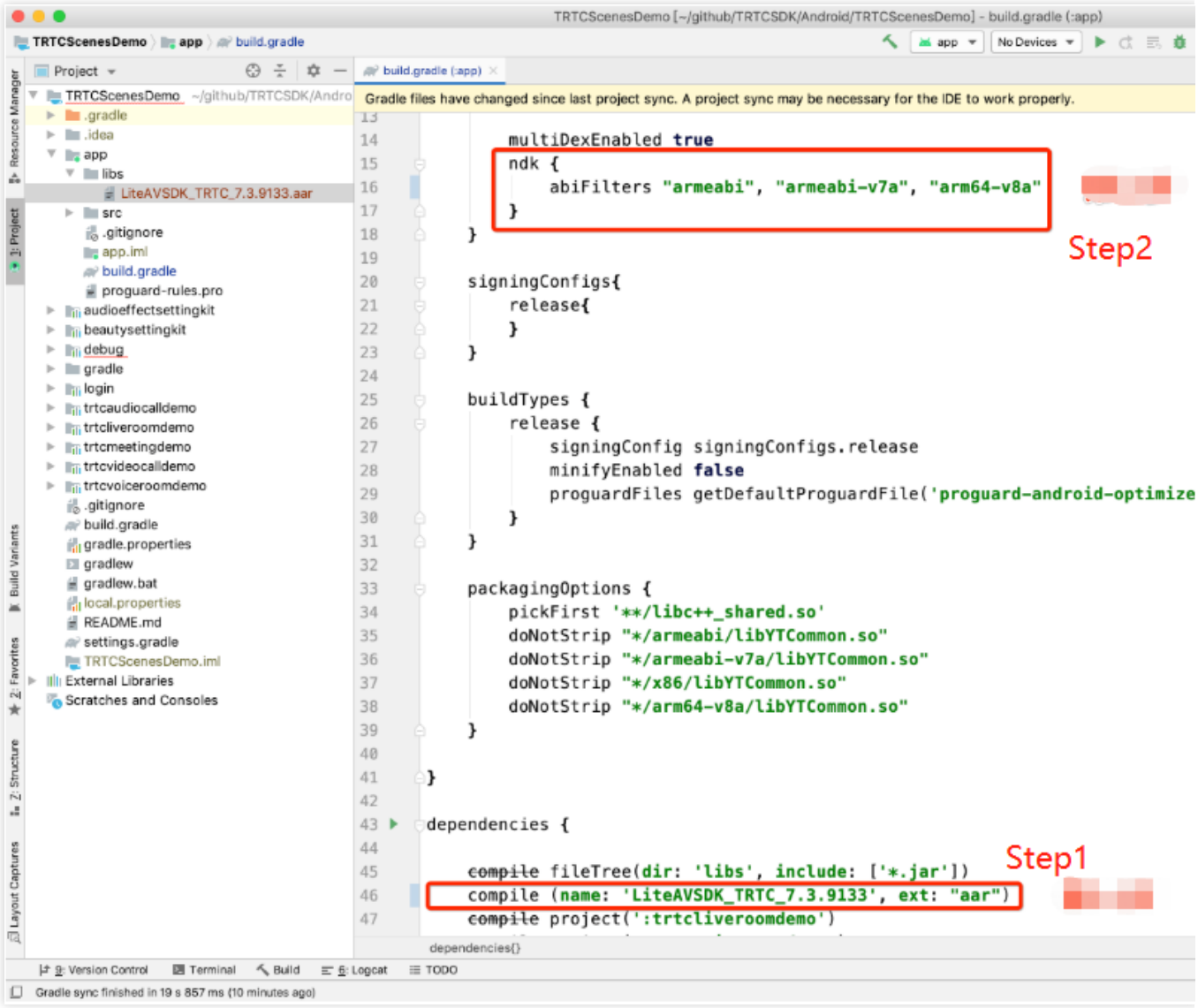
Sync Now，自动下载 SDK 并集成到工程里。

方案二：下载 SDK 并手动导入

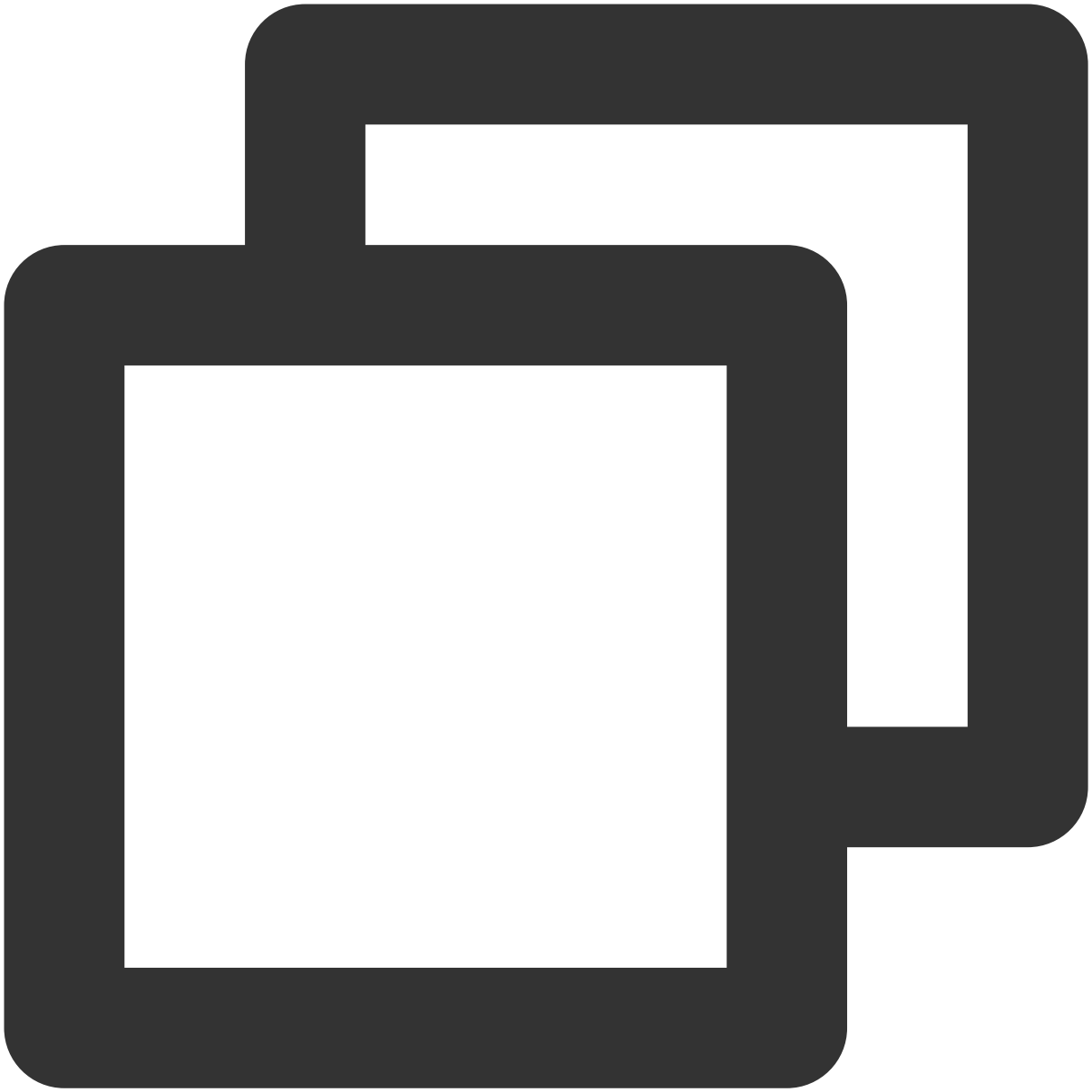
1. 下载 SDK 并解压到本地。
2. 将解压的 aar 文件拷贝到工程的 **app/libs** 目录下。
3. 在工程根目录下的 build.gradle 中，添加 **flatDir**，指定本地仓库路径。



4. 在 app/build.gradle 中，添加引用 aar 包的代码。



5. 在 app/build.gradle的defaultConfig 中，指定 App 使用的 CPU 架构。



```
defaultConfig {
     ndk {
             abiFilters "armeabi", "armeabi-v7a", "arm64-v8a"
     }
}
```

说明：

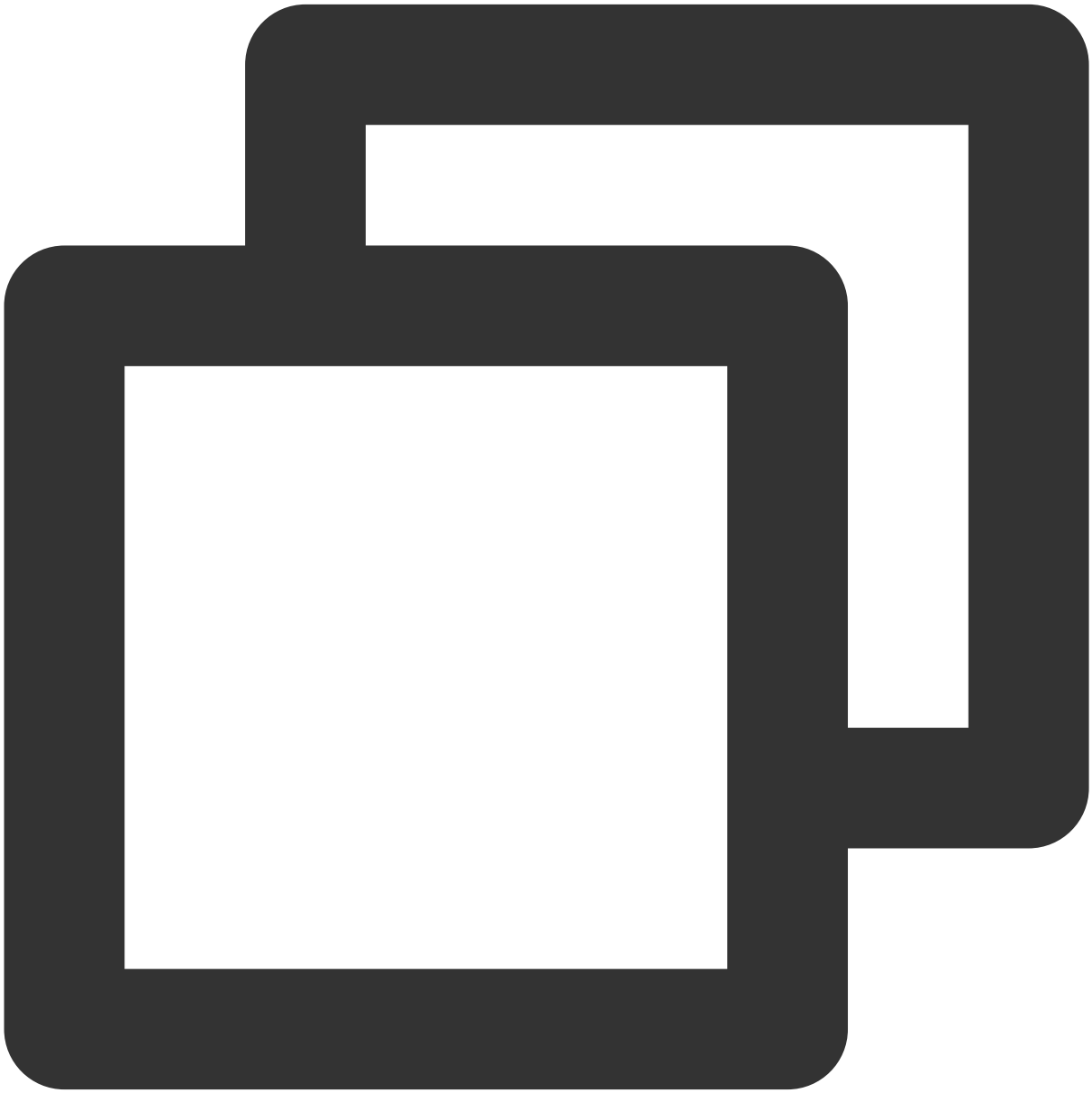
目前 TRTC SDK 支持 armeabi ， armeabi-v7a 和 arm64-v8a。

6. 单击

Sync Now，完成 TRTC SDK 的集成工作。

第二步：配置 App 权限

在 AndroidManifest.xml 中配置 App 的权限，TRTC SDK 需要以下权限：



```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

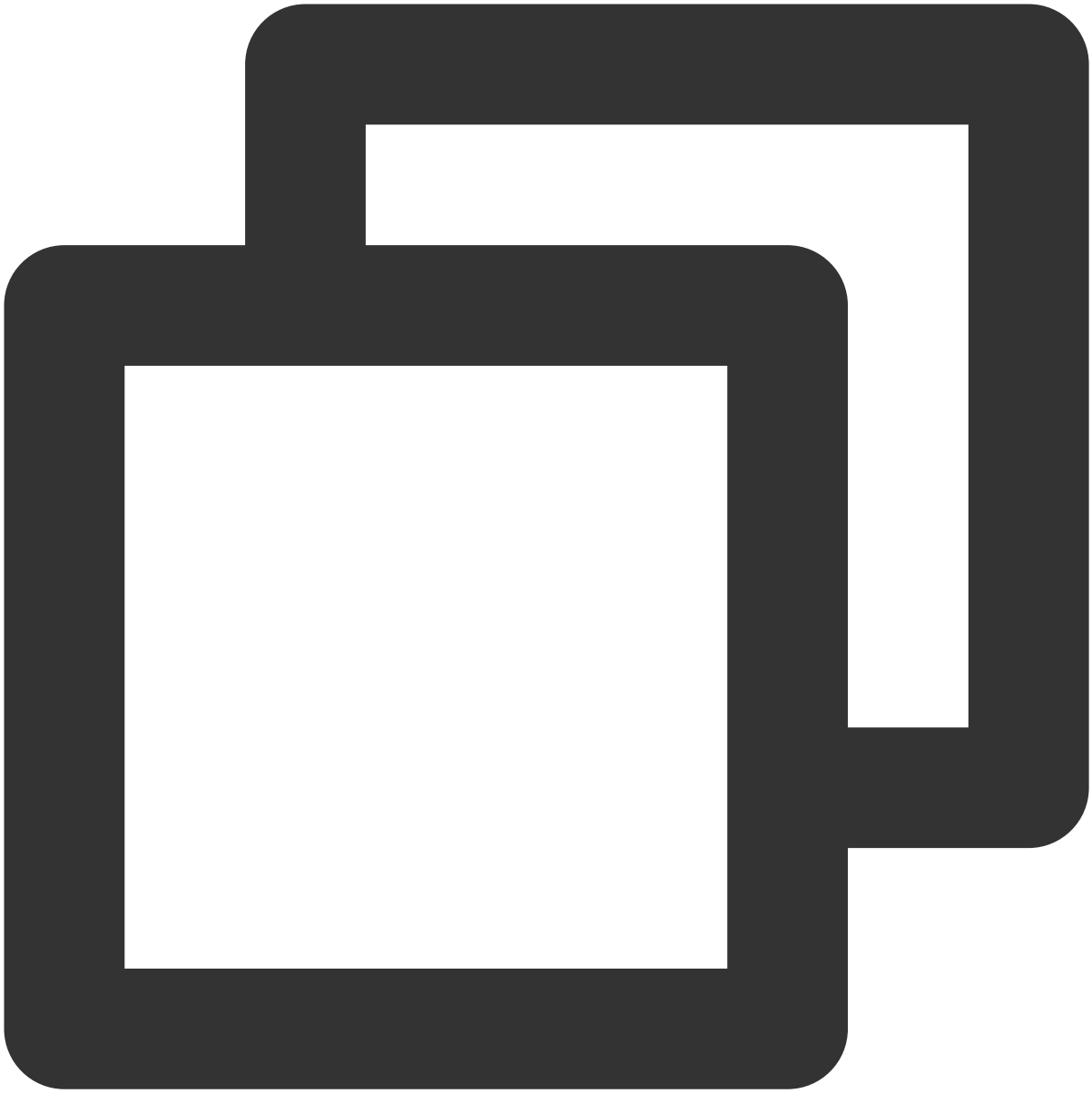
```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

注意：

请勿设置 `android:hardwareAccelerated="false"`，关闭硬件加速之后，会导致对方的视频流无法渲染。

第三步：设置混淆规则

在 proguard-rules.pro 文件，将 TRTC SDK 相关类加入不混淆名单：



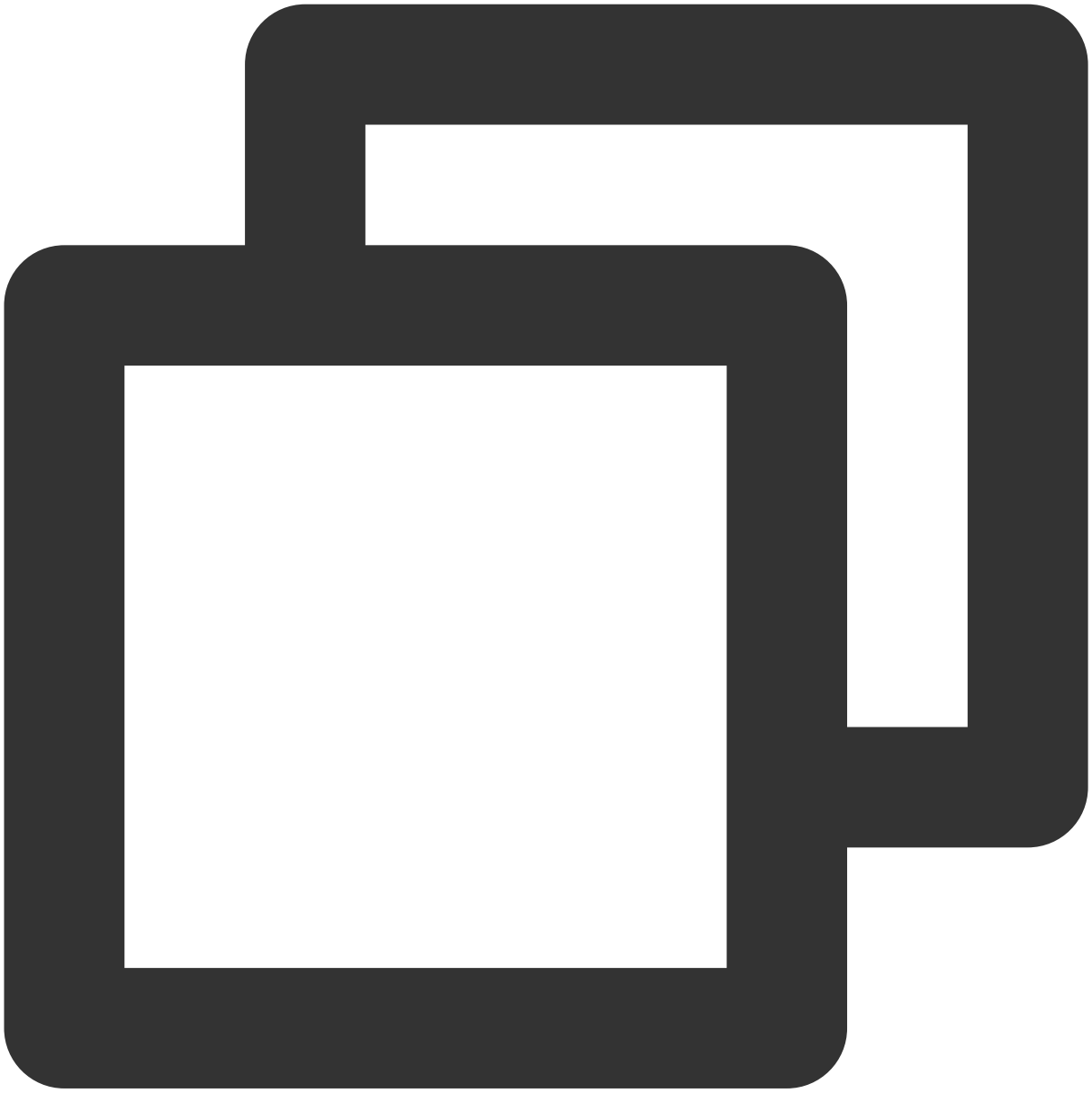
```
-keep class com.tencent.** { *; }
```

通过 C++ 接口使用 SDK（可选）

如果您更倾向于使用 C++ 接口，而不是 Java 进行开发，可以执行此步骤；如果您仅使用 Java 语言来调用 TRTC SDK，请忽略此步。

1. 首先需要根据上文的指引，通过导入 jar 和 so 库的方式集成 TRTC SDK。

2. 拷贝头文件：将 SDK 中的 C++ 头文件拷贝到项目中（路径为：`SDK/LiteAVSDK_TRTC_xxx/libs/include`），并在 CMakeLists.txt 中配置 include 文件夹路径及 so 库的动态链接。



```
cmake_minimum_required(VERSION 3.6)

# 配置 C++ 接口头文件路径
include_directories(
     ${CMAKE_CURRENT_SOURCE_DIR}/include  # 拷贝自 SDK/LiteAVSDK_TRTC_xxx/libs/inclu
)
```

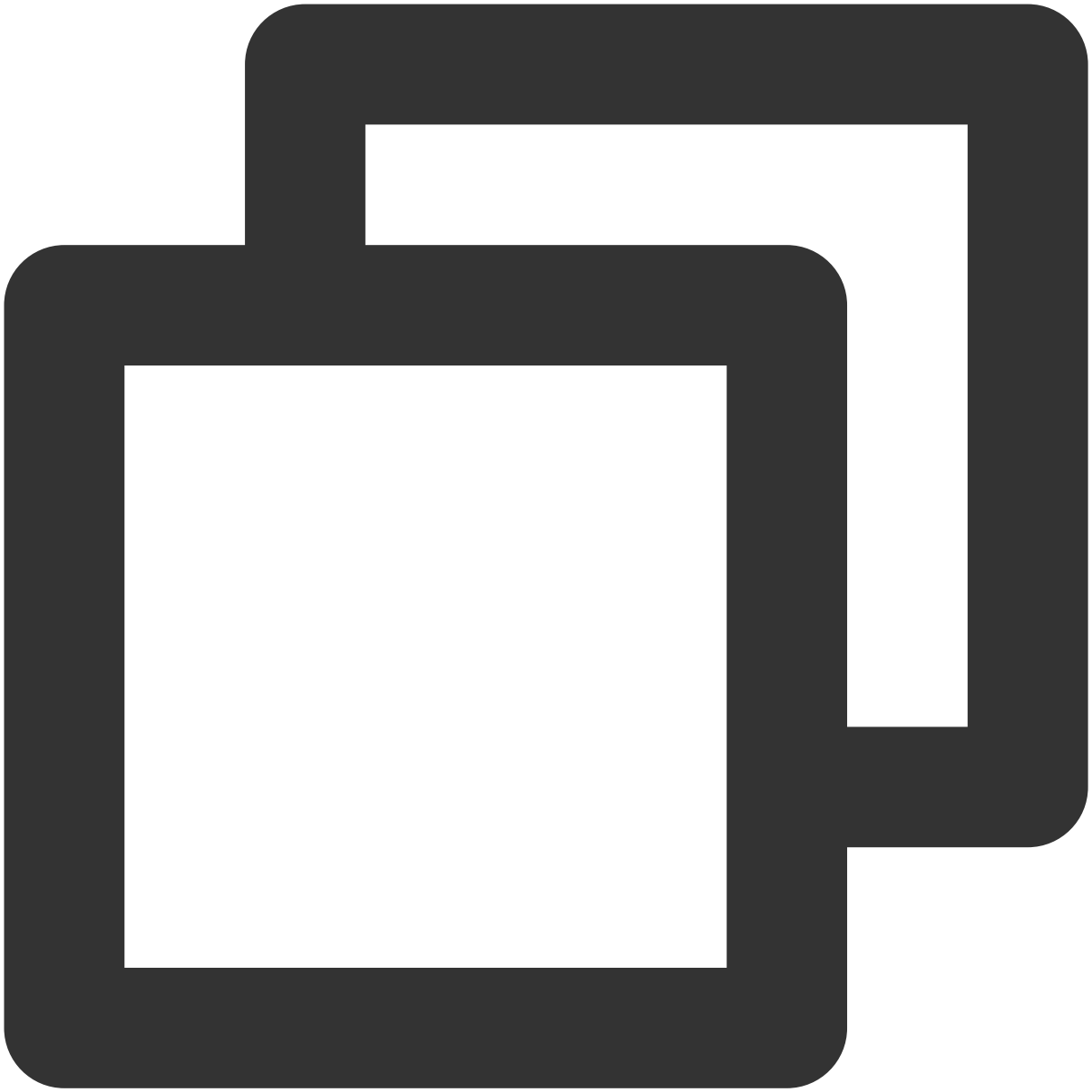
```
add_library(
     native-lib
     SHARED
     native-lib.cpp)

# 配置 libliteavsdk.so 动态库路径
add_library(libliteavsdk SHARED IMPORTED)
set_target_properties(libliteavsdk  PROPERTIES IMPORTED_LOCATION ${CMAKE_CURRENT_SO

find_library(
     log-lib
     log)

# 配置 libliteavsdk.so 动态链接
target_link_libraries(
     native-lib
     libliteavsdk
     ${log-lib})
```

3. 使用命名空间：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间



```
using namespace trtc;
```

说明：

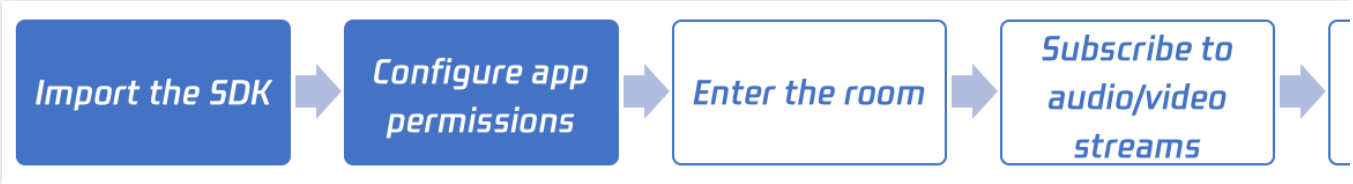
配置 Android Studio C/C++ 开发环境具体可以参考 Android Studio 官方文档：向 Android 项目添加 C 和 C++ 代码 。

目前只有 TRTC 版本的 SDK 支持 C++ 接口；对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

Mac

最近更新时间：2024-06-07 11:20:49

本文主要介绍如何快速地将腾讯云 TRTC SDK（Mac）集成到您的项目中，只要按照如下步骤进行配置，就可以完成 SDK 的集成工作。



开发环境要求

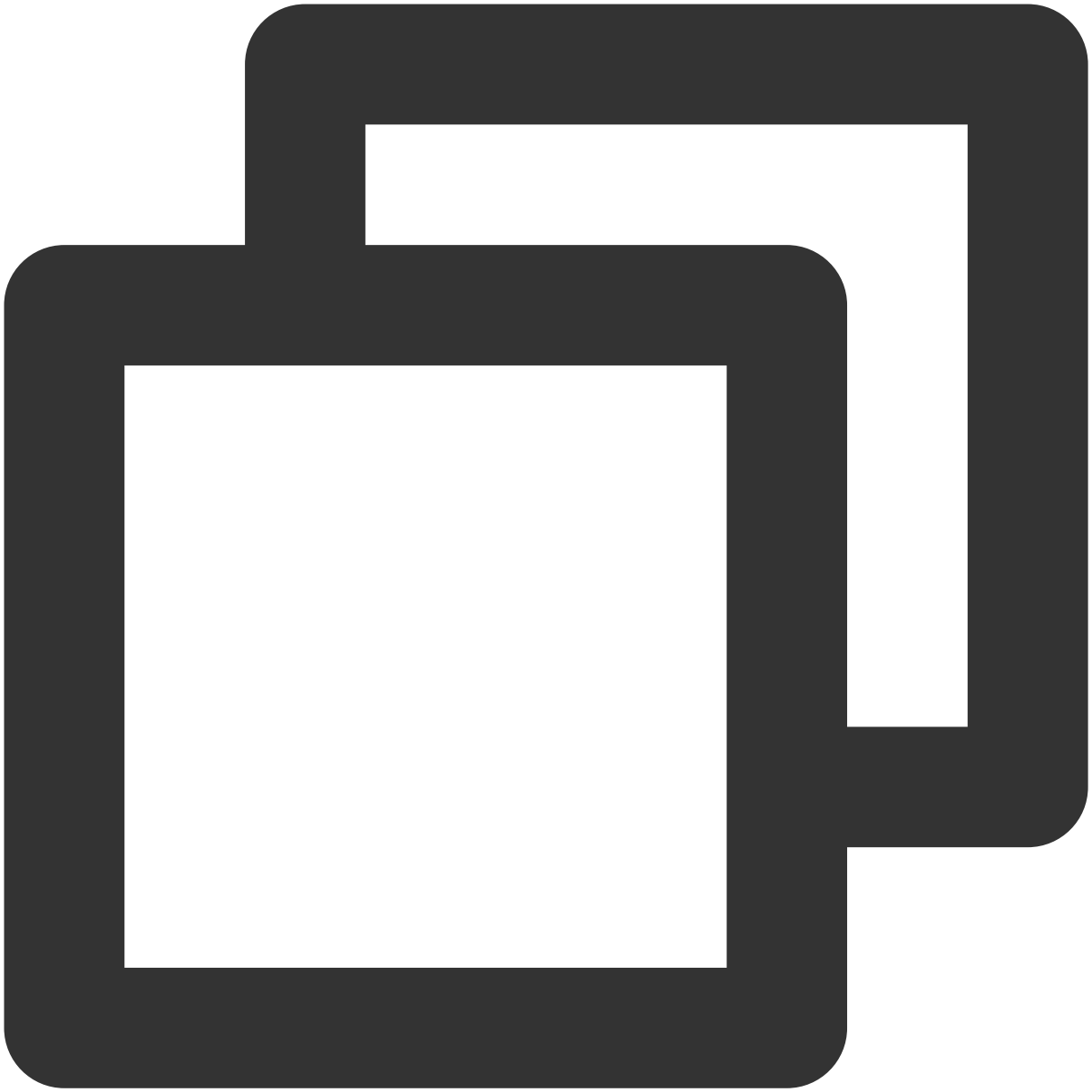
Xcode 9.0+。
OS X10.10+ 的 Mac 真机。
项目已配置有效的开发者签名。

第一步：导入 SDK

您可以选择使用 CocoaPods 自动加载的方式，或者手动先下载 SDK 再将其导入到您当前的工程项目中。

方案一：CocoaPods

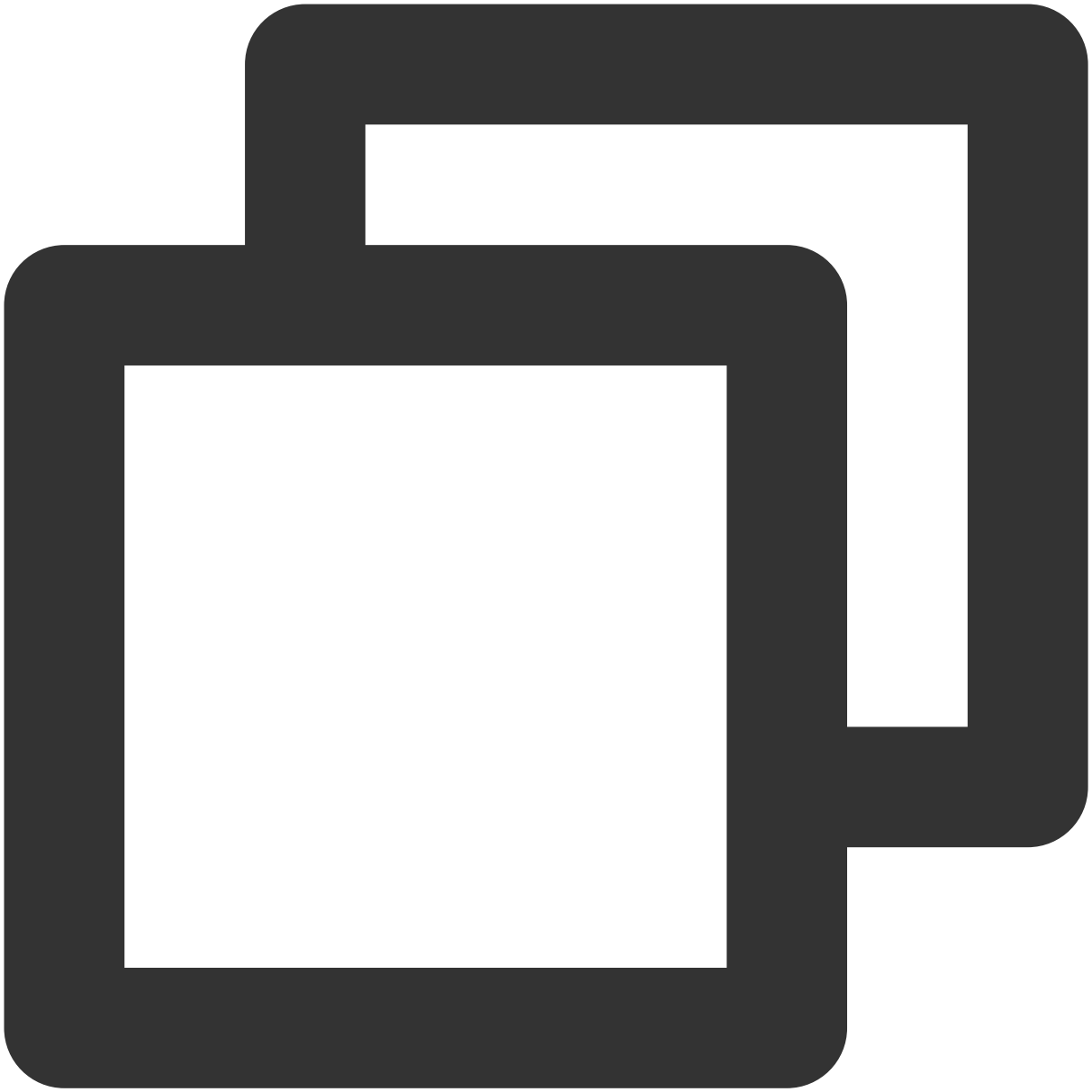
1. **安装 CocoaPods。**
在终端窗口中输入如下命令（需要提前在 Mac 中安装 Ruby 环境）：



```
sudo gem install cocoapods
```

2. **创建 Podfile 文件。**

进入项目所在路径，输入以下命令行之后项目路径下会出现一个 Podfile 文件。

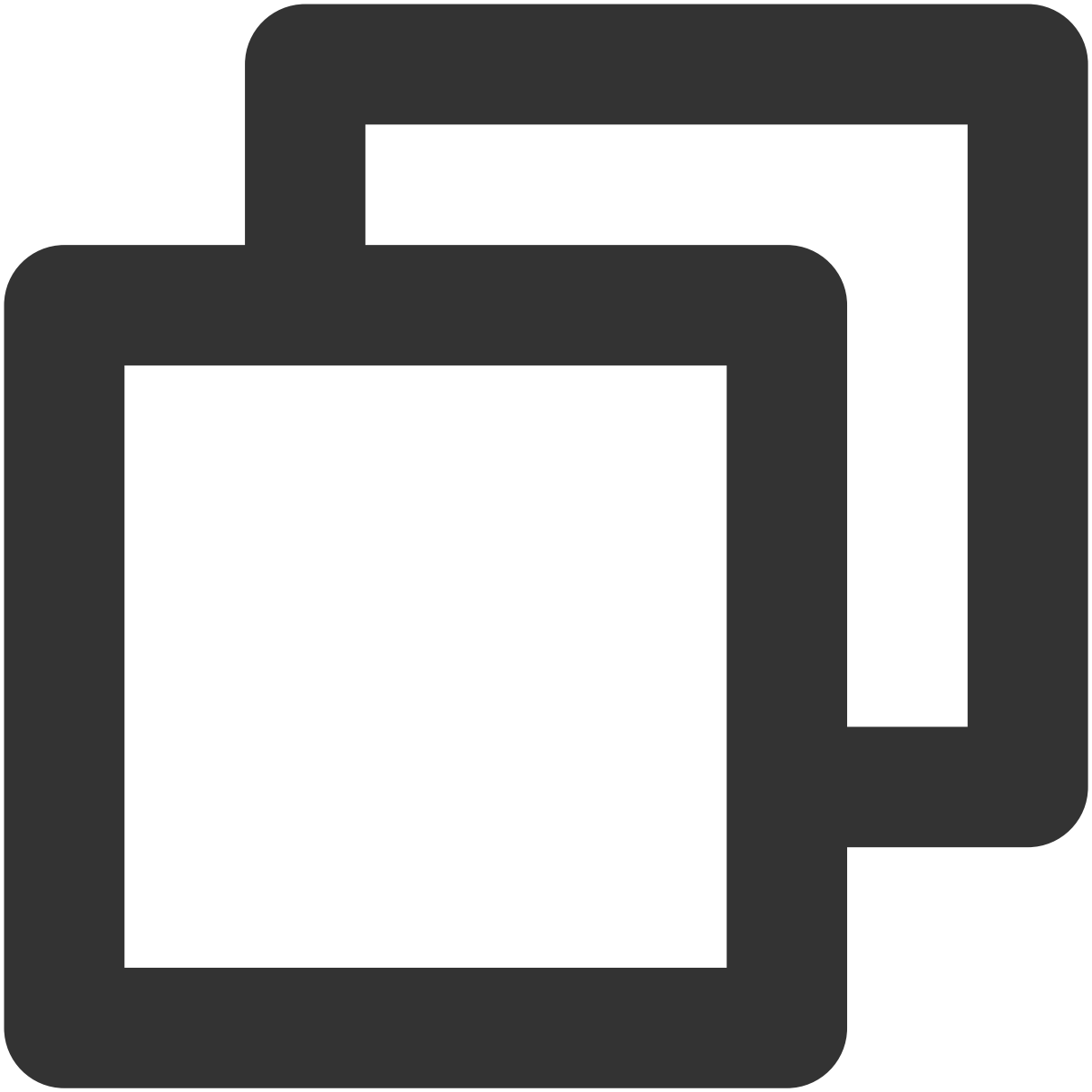


```
pod init
```

3. **编辑 Podfile 文件。**

编辑 Podfile 文件，有如下有两种设置方式：

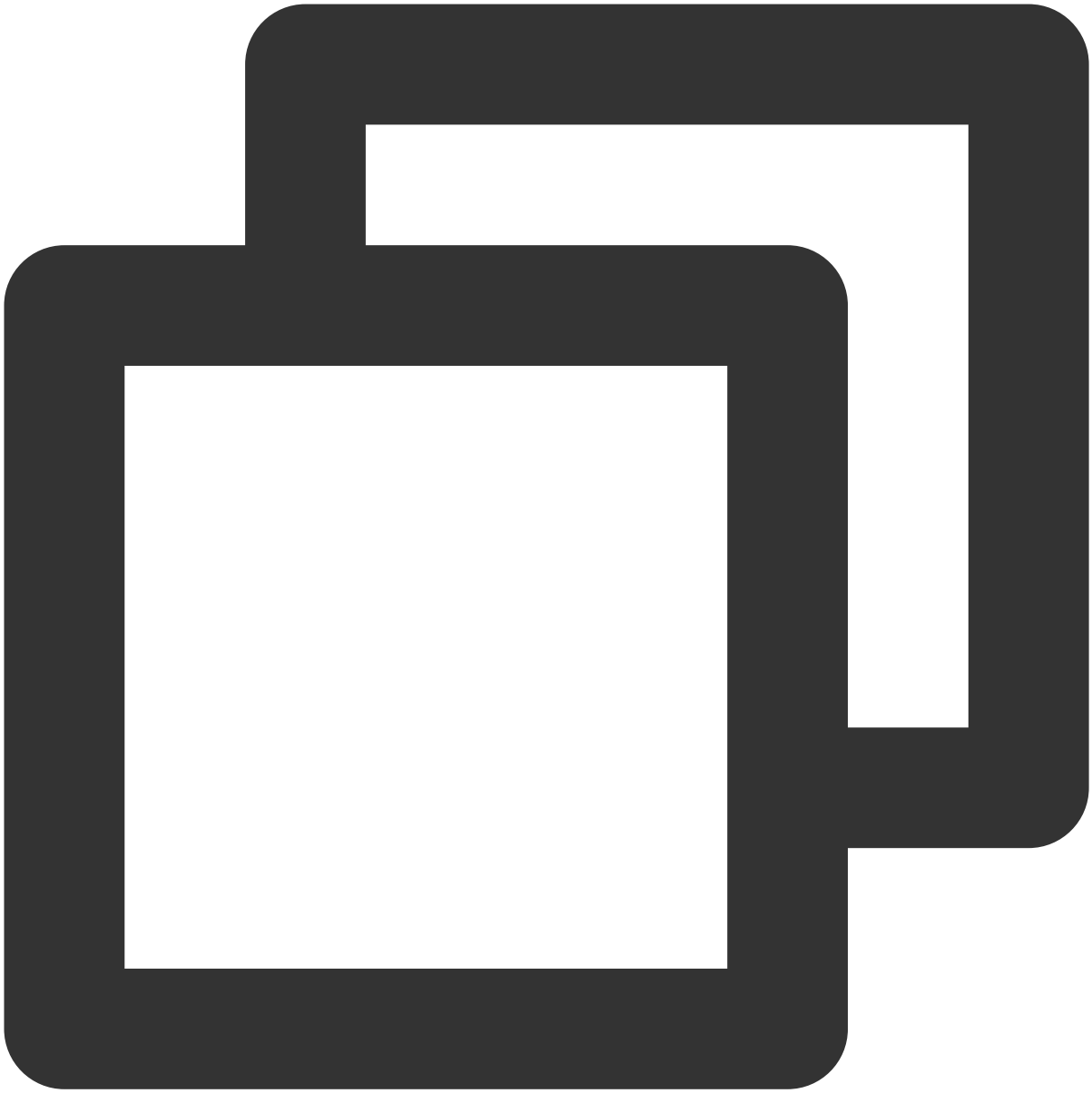
方式一：使用腾讯云 LiteAV SDK 的 pod 路径。



```
platform :osx, '10.10'

target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac', :podspec => 'https://liteav.sdk.qcloud.com/pod/liteavsd
end
```

方式二：使用 CocoaPod 官方源，支持选择版本号。

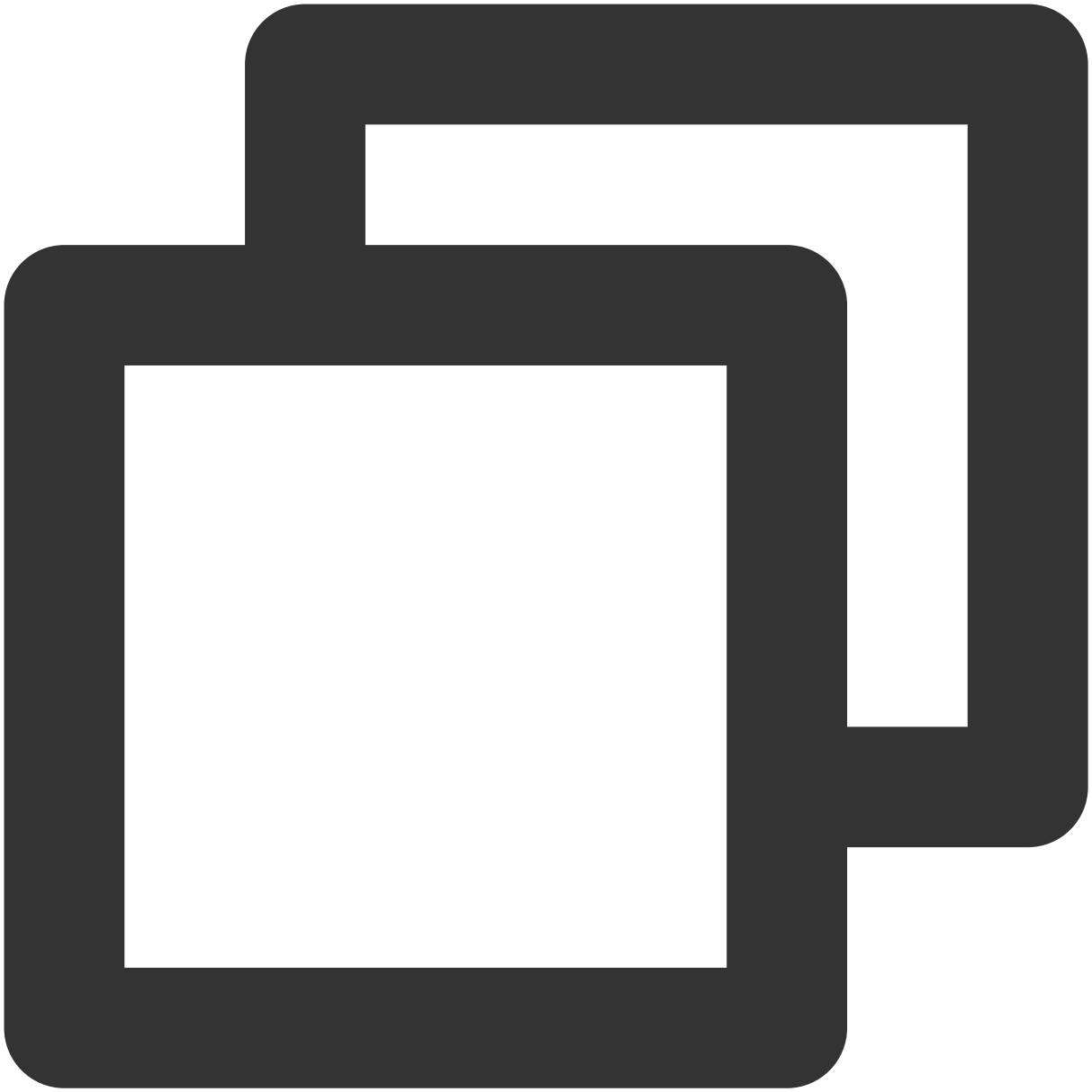


```
platform :osx, '10.10'
source 'https://github.com/CocoaPods/Specs.git'

target 'Your Target' do
pod 'TXLiteAVSDK_TRTC_Mac'
end
```

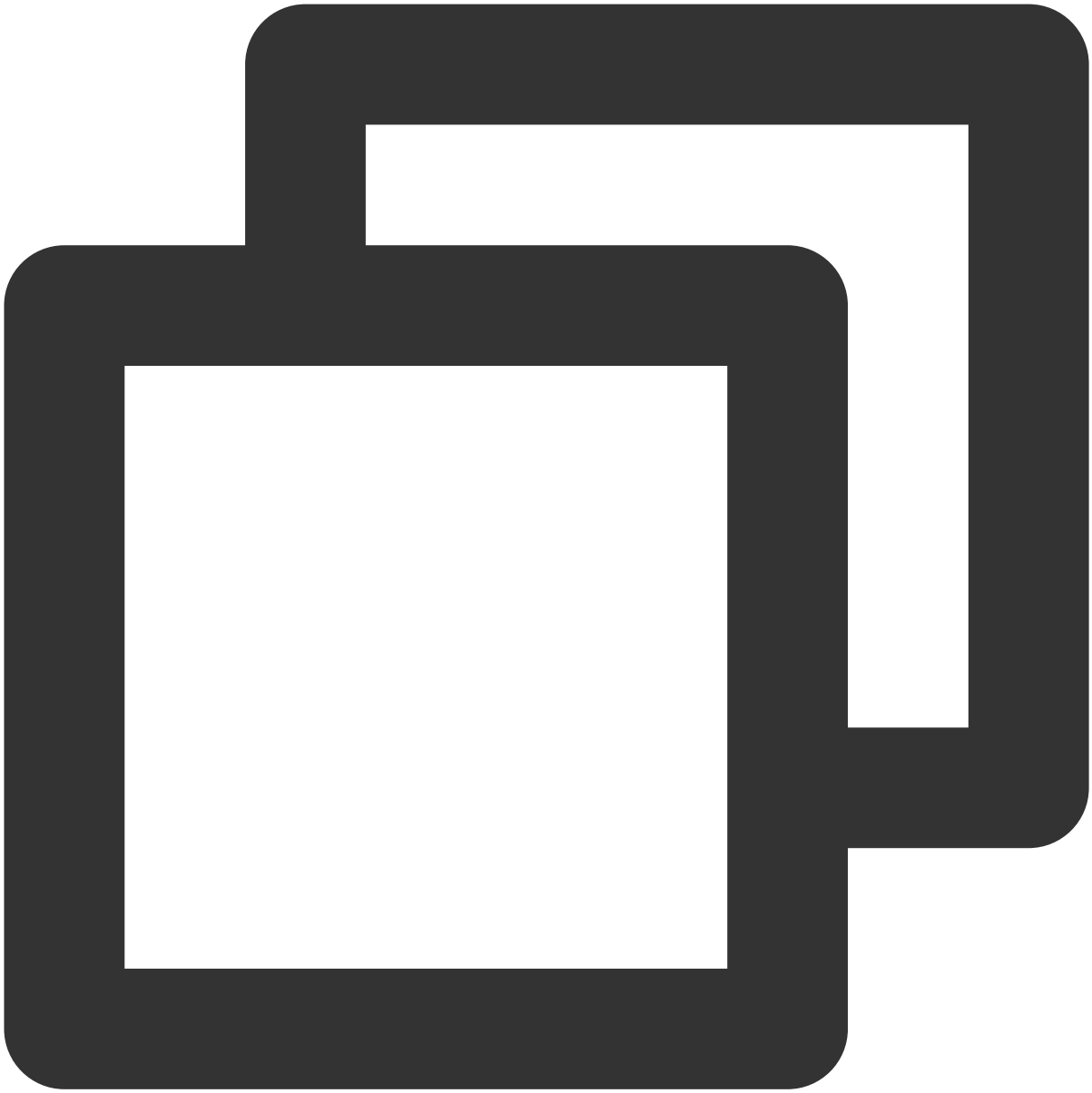
4. **安装与更新 SDK。**

在终端窗口中输入如下命令执行安装 TRTC SDK：



```
pod install
```

或使用以下命令更新本地库版本：

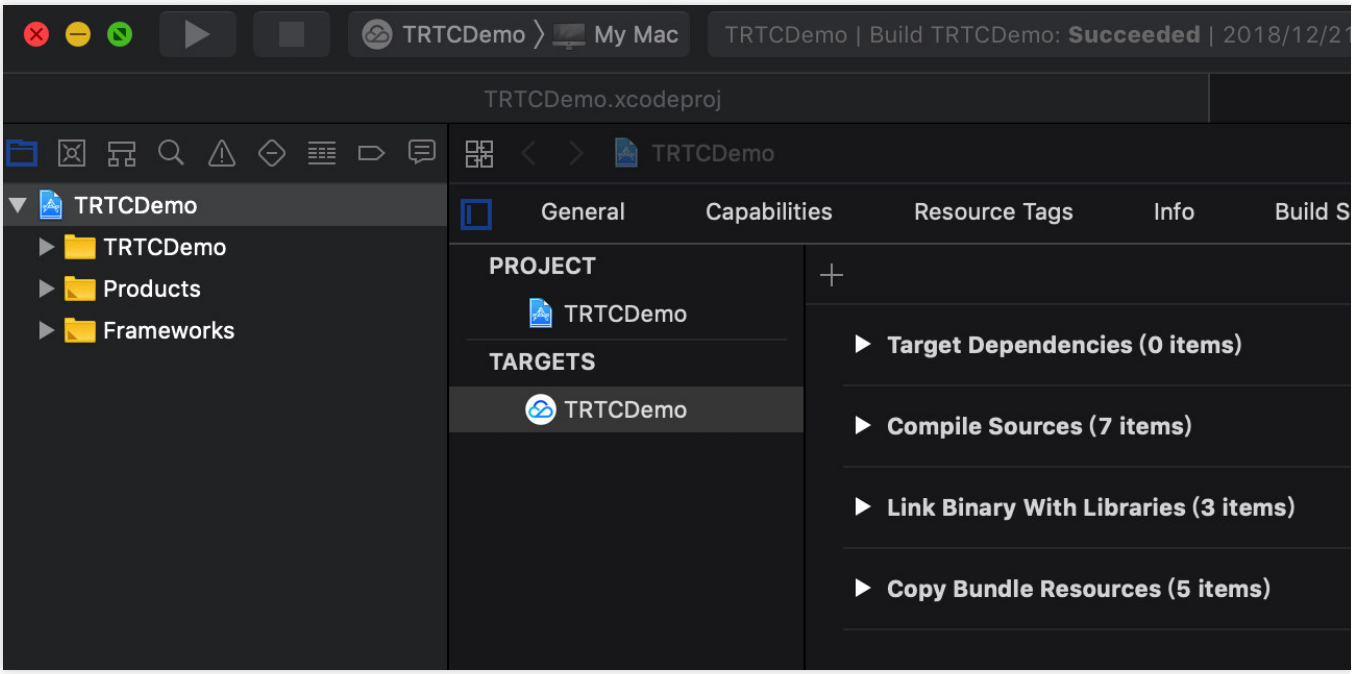


```
pod update
```

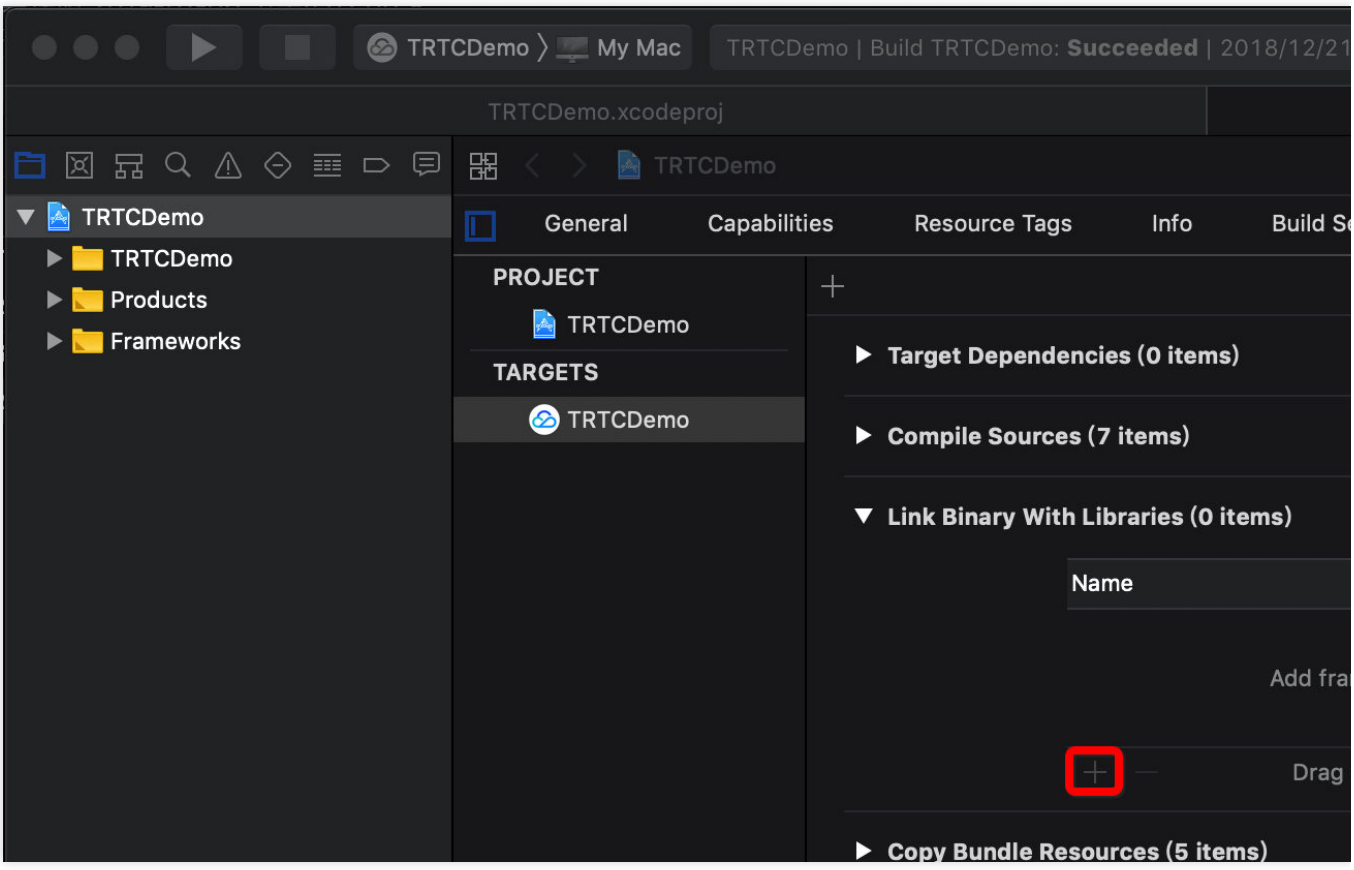
pod 命令执行完后，会生成集成了 SDK 的 `.xcworkspace` 后缀的工程文件，双击打开即可。

方案二：手动集成

1. 下载 TRTC-SDK 的 Mac 版本。
2. 打开您的 Xcode 工程项目，将第一步中下载的 framework 导入到您的工程。
3. 选择要运行的 target，选中 Build Phases 项。

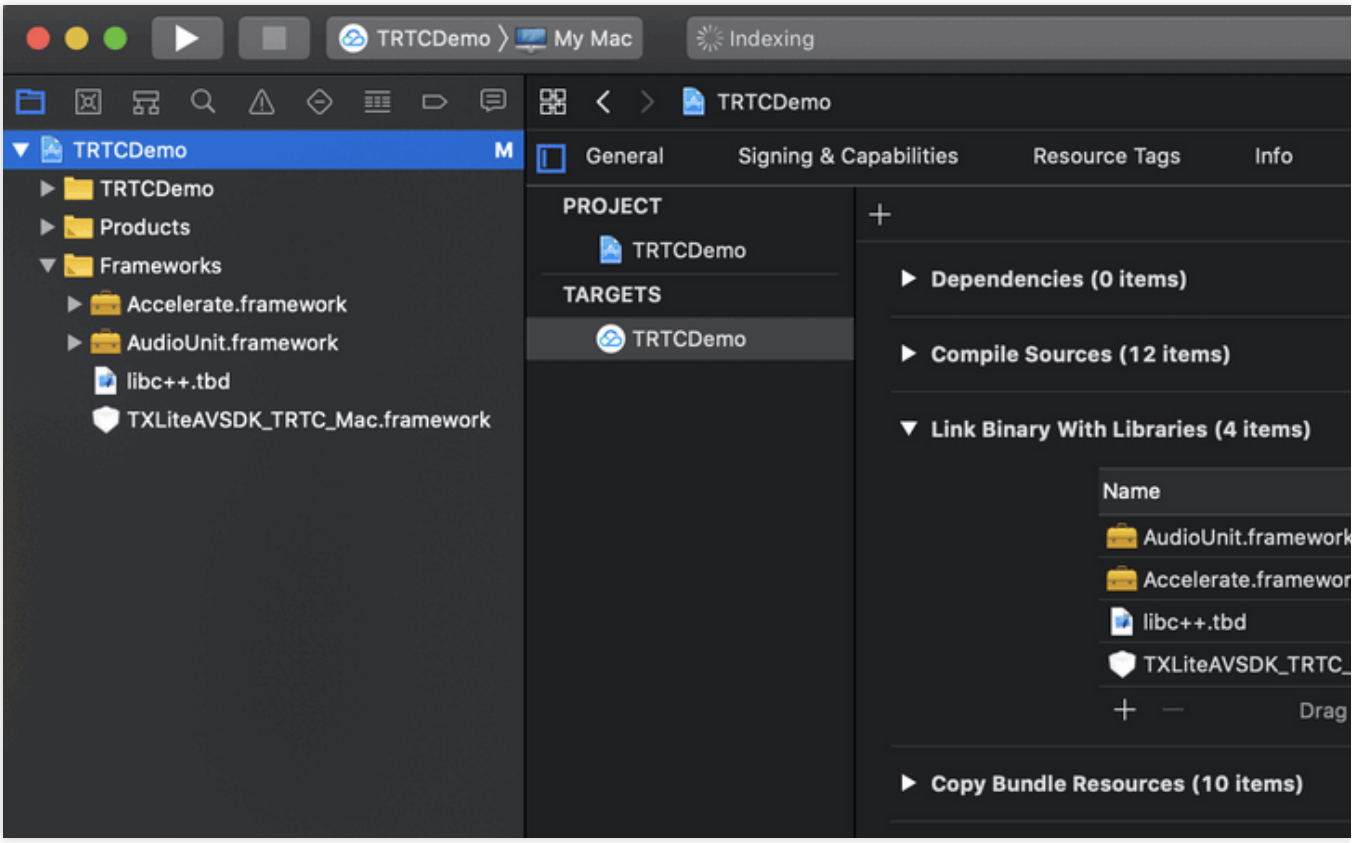


4. 单击 **Link Binary with Libraries** 项展开，单击底下的+号图标去添加依赖库。



5. 依次添加所下载的 SDK Framework 及其所需依赖库： `TXFFmpeg.xcframework` 、 `TXSoundTouch.xcframework` 和 `libc++.tbd` 、

`Accelerate.framework`、`SystemConfiguration.framework`、`MetalKit.framework`。添加后如下图所示：



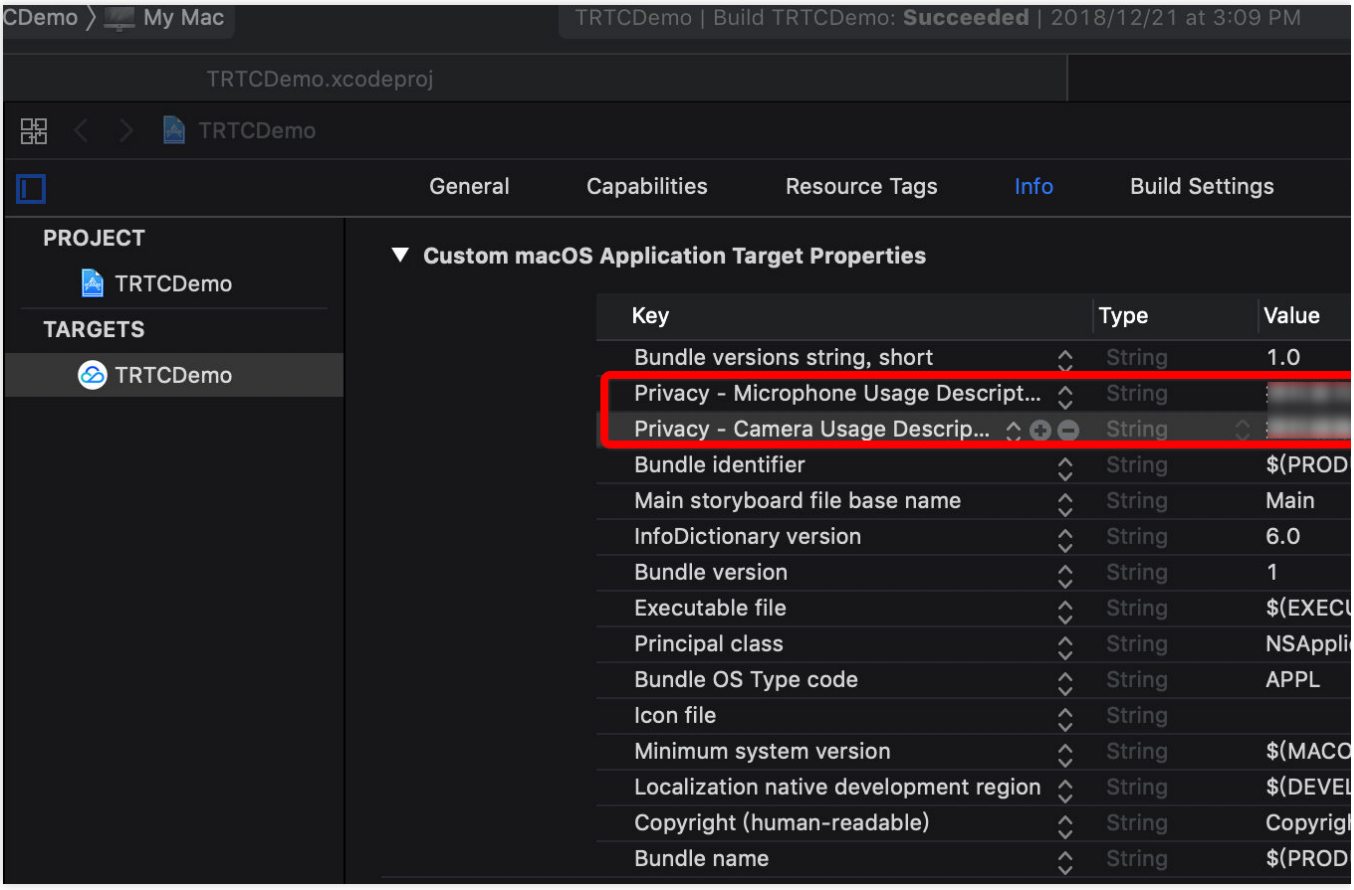
第二步：配置 App 权限

使用 SDK 的音视频功能，需要授权麦克风和摄像头的使用权限。在 App 的 Info.plist 中添加以下两项，分别对应麦克风和摄像头在系统弹出授权对话框时的提示信息。

Privacy - Microphone Usage Description，并填入麦克风使用目的的提示语。

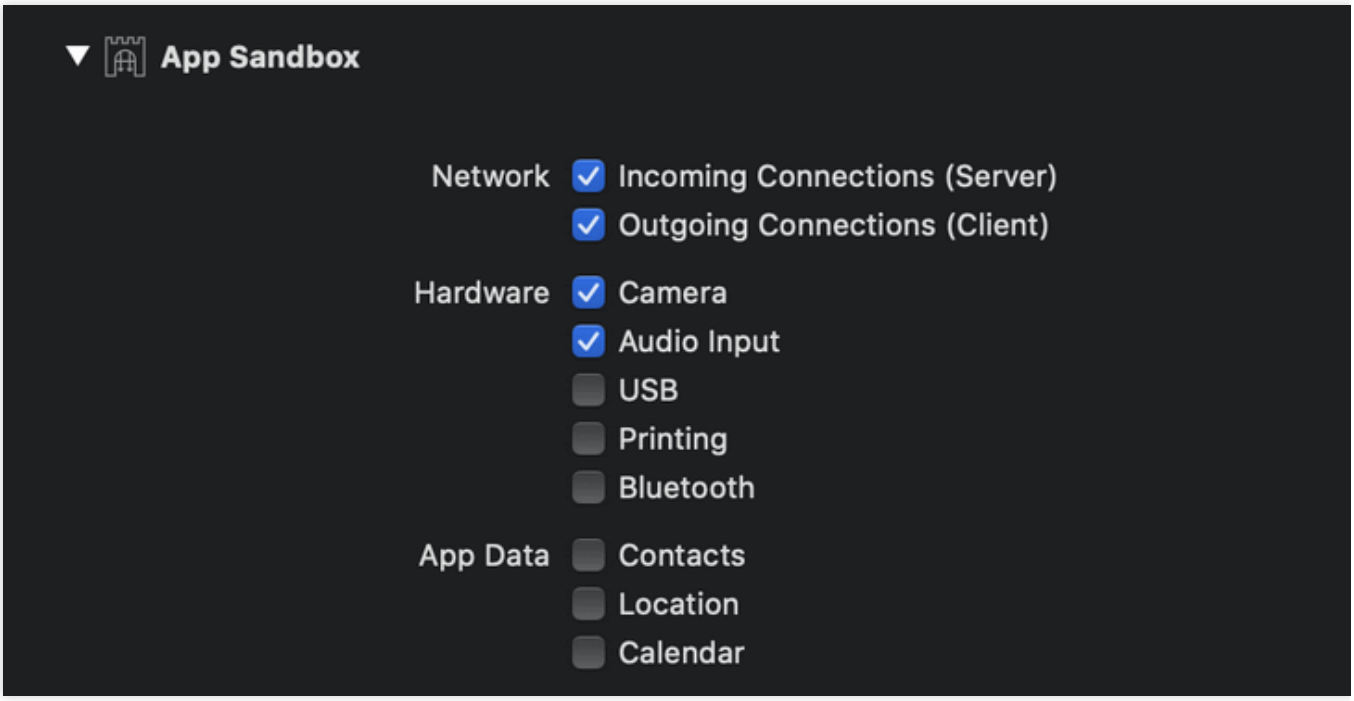
Privacy - Camera Usage Description，并填入摄像头使用目的的提示语 。

如下图所示：

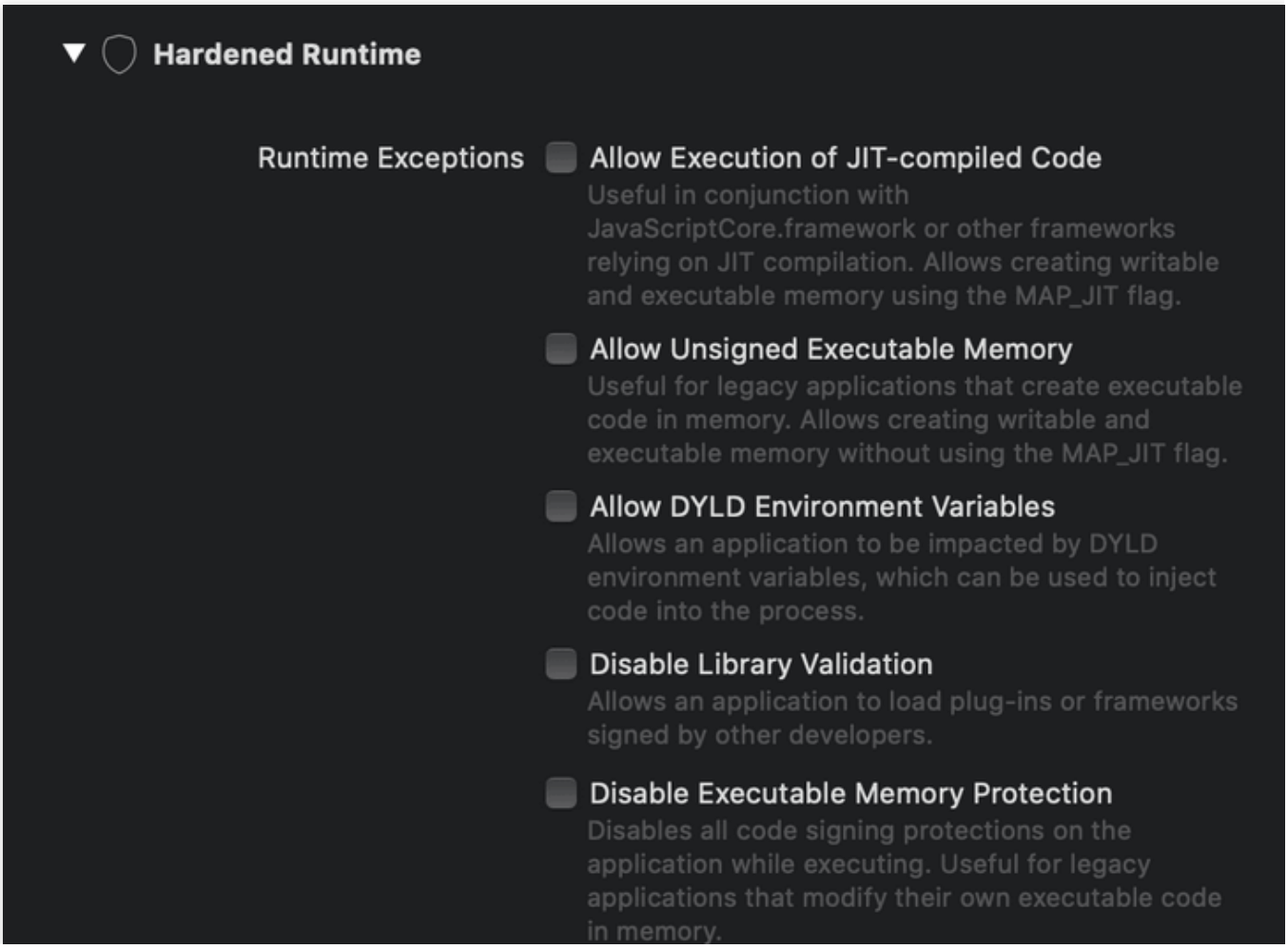


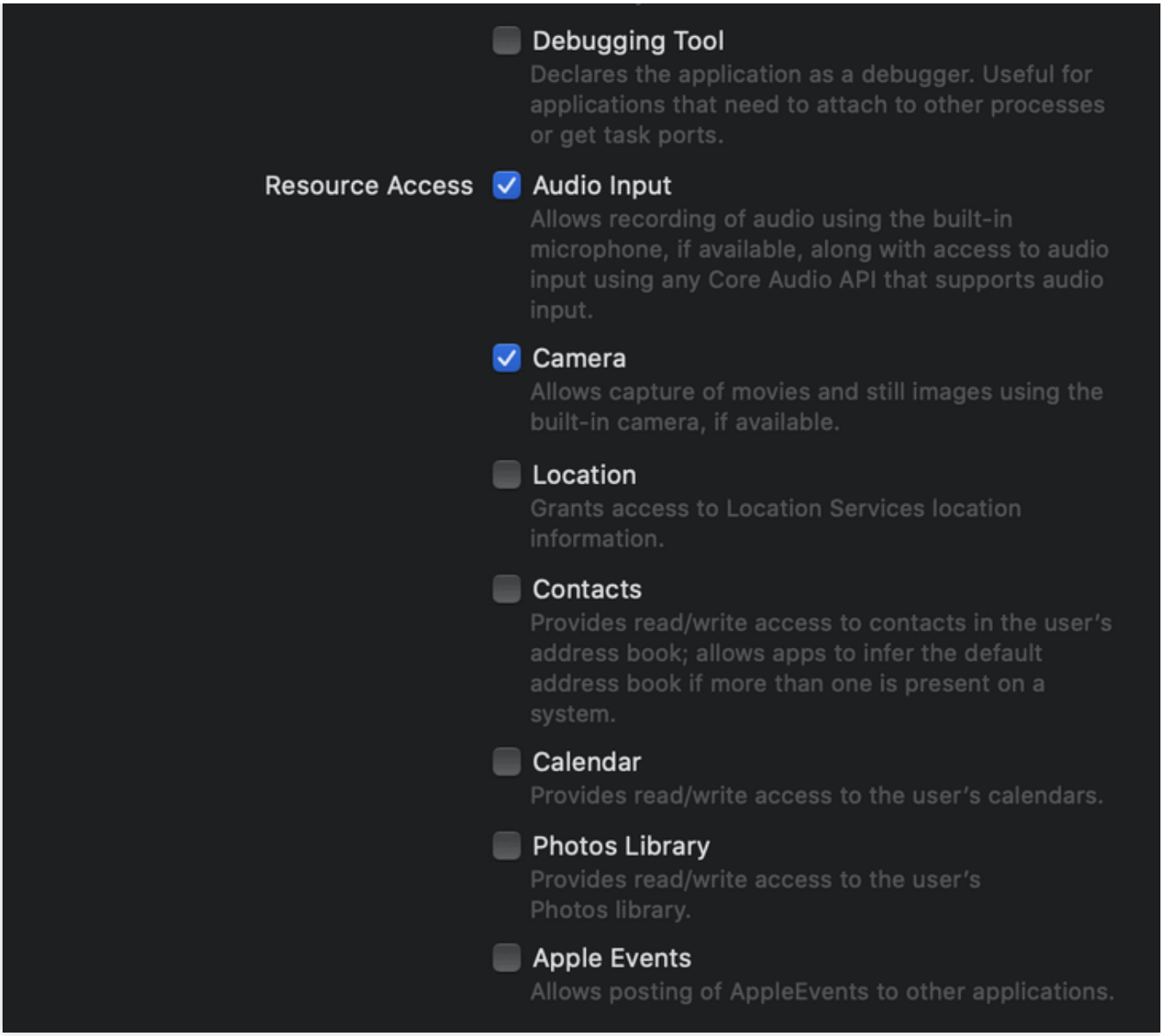
如果 App 启用了 **App Sandbox** 或 **Hardened Runtime**，需要勾选上 `Network`、`Camera` 和 `Audio Input` 这几个选项。

App Sandbox 配置如下图所示：



Hardened Runtime 配置如下图所示：



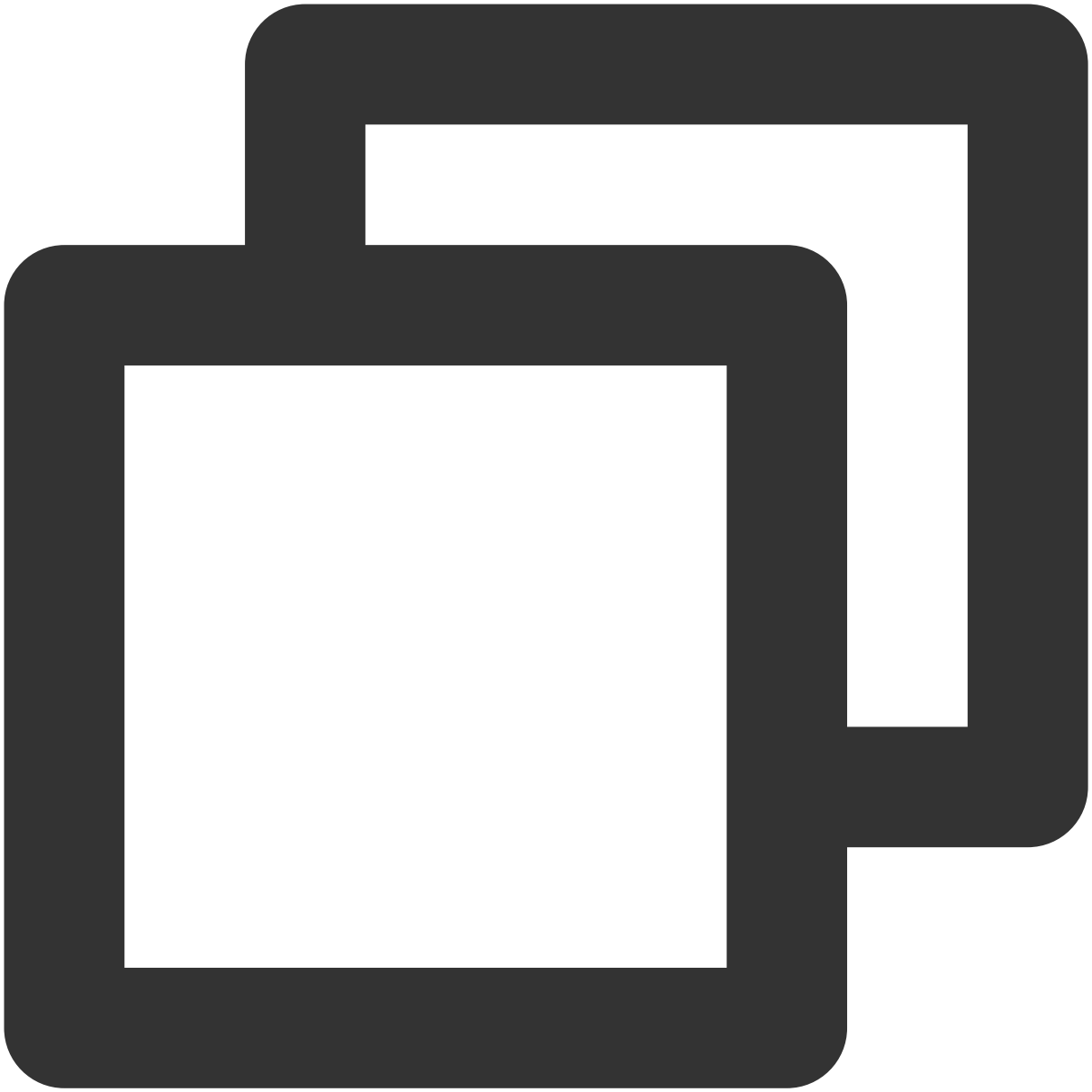


第三步：在项目中引用 SDK

在完成了第一步的导入和第二步的设备权限授权后，就可以在项目中引用 SDK 中提供的接口 API 了。

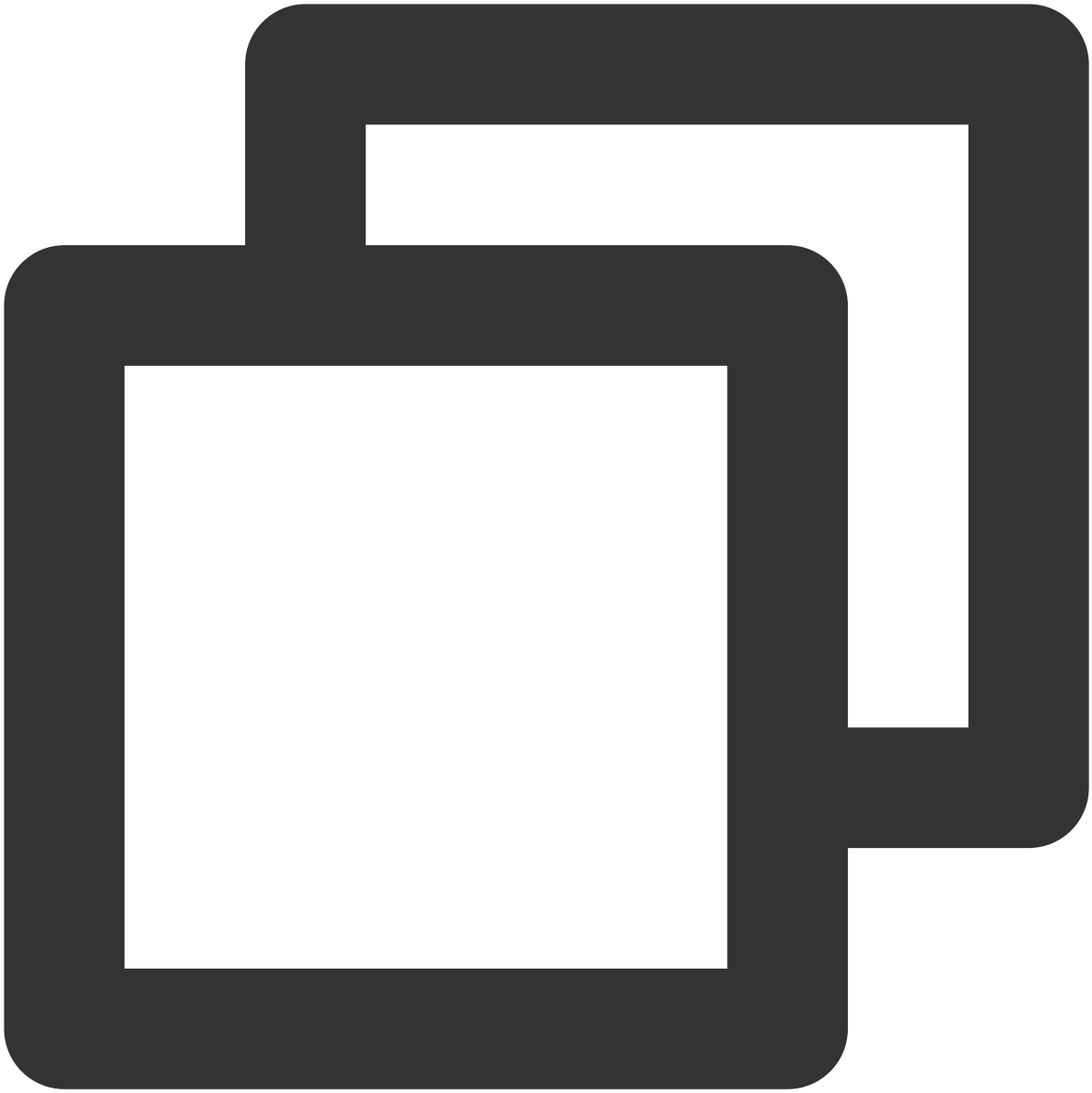
通过 Objective-C 或 Swift 接口引用 TRTC SDK

在 Objective-C 或 Swift 代码中使用 SDK 有两种方式：
模块引用：在项目需要使用 SDK API 的文件里，添加模块引用。



```
@import TXLiteAVSDK_TRTC_Mac;
```

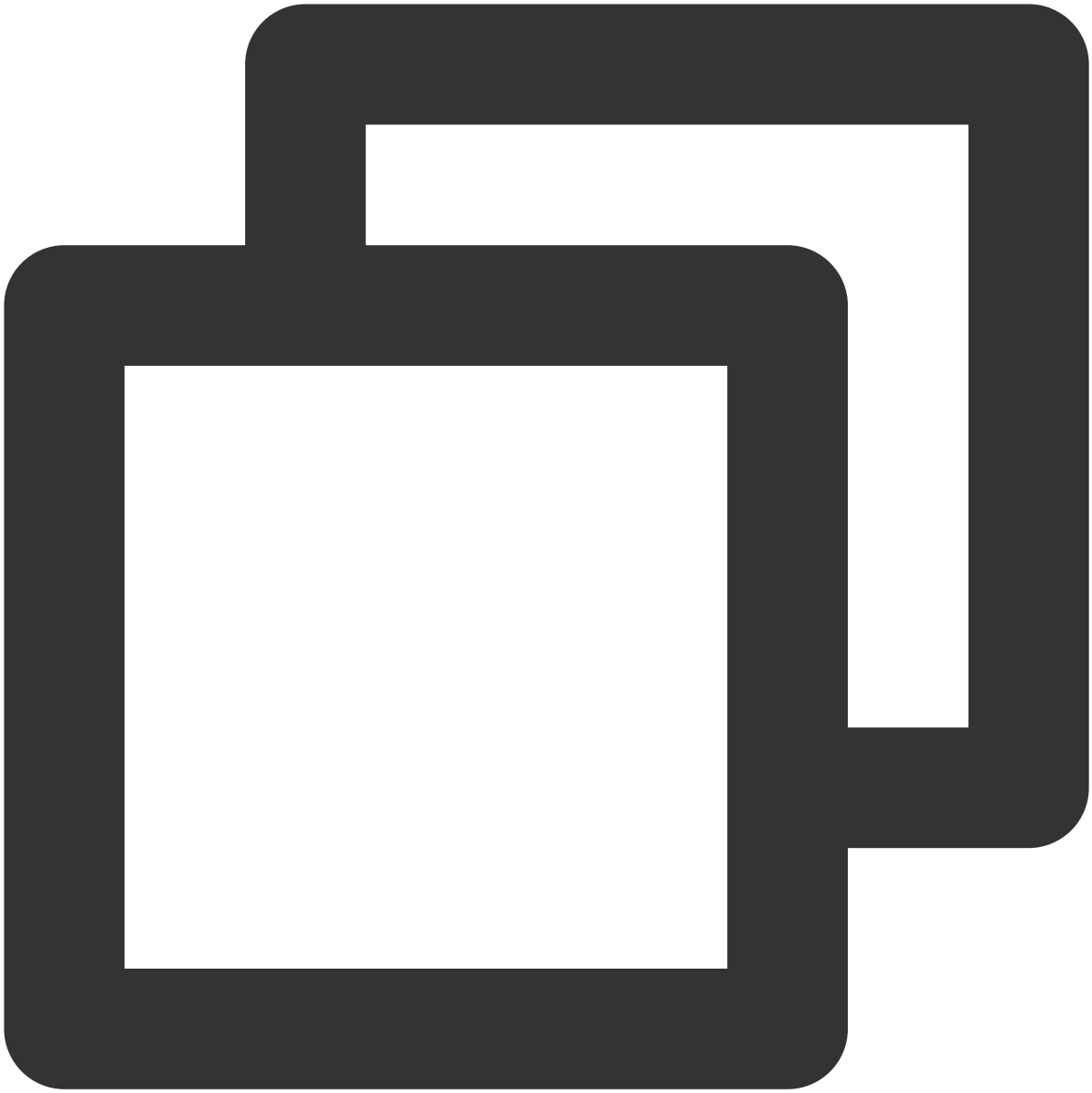
头文件引用：在项目需要使用 SDK API 的文件里，引入具体的头文件。



```
#import TXLiteAVSDK_TRTC_Mac/TRTCCloud.h
```

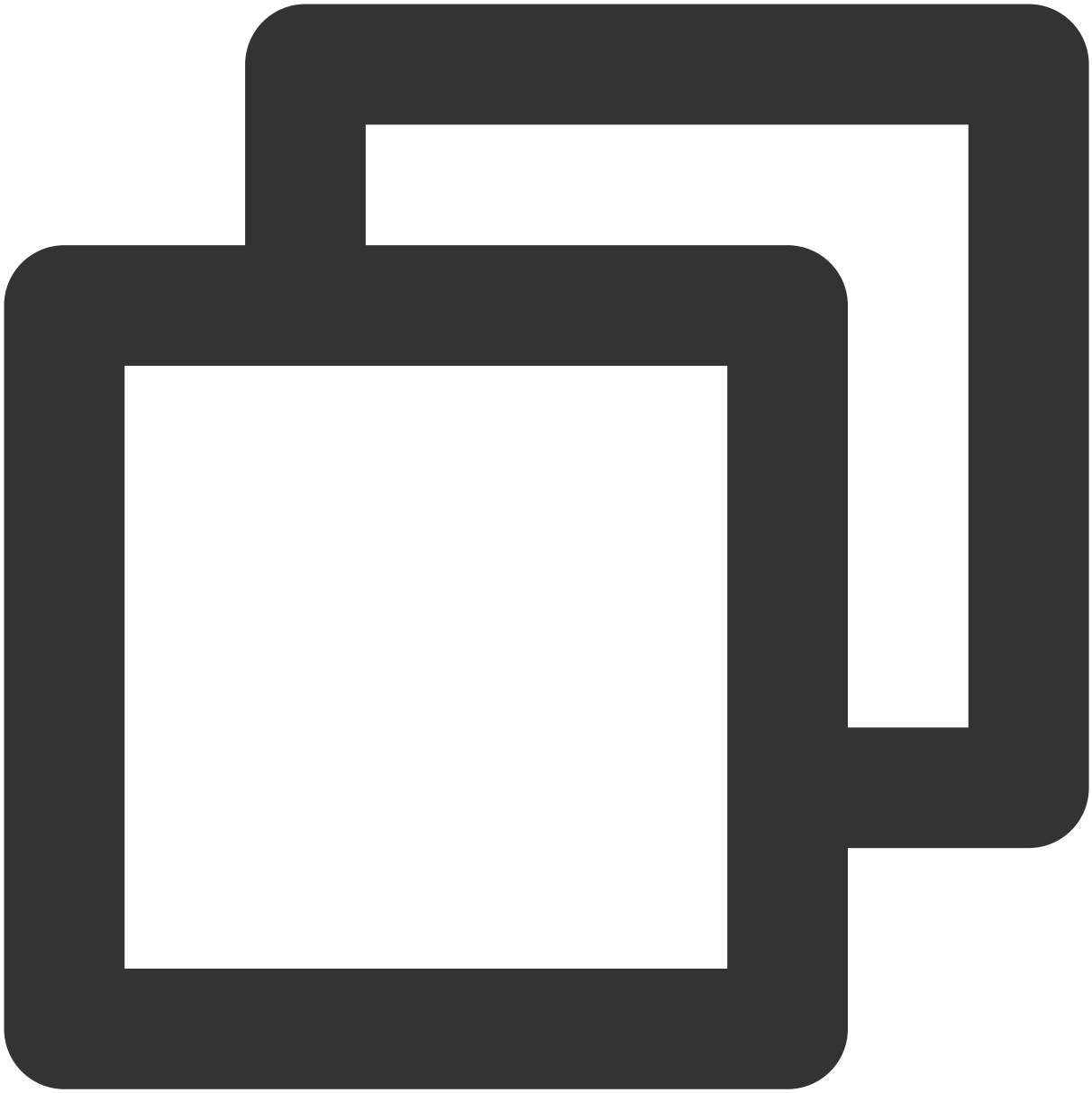
通过 C++ 接口引用 TRTC SDK（可选）

1. **引用头文件**：如果您要使用 C++ 接口来开发Mac应用，请引用 `TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface` 目录下的头文件。



```
#include TXLiteAVSDK_TRTC_Mac/cpp_interface/ITRTCCloud.h
```

2. **使用命名空间**：C++ 全平台接口的接口、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间。



```
using namespace trtc;
```

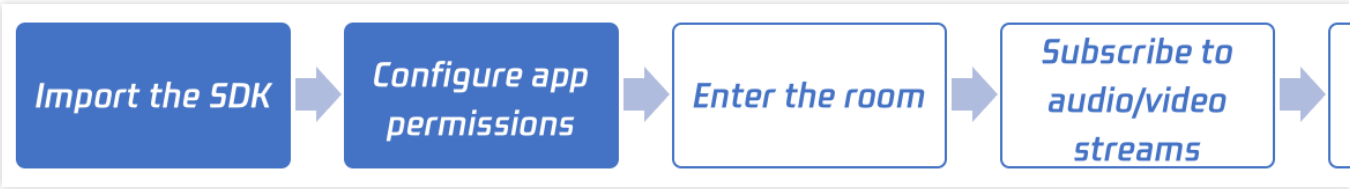
说明：

对于 C++ 接口的使用方式，请参见 全平台（C++）API 概览。

Windows C++

最近更新时间：2024-06-07 11:20:49

本文介绍如何快速地将腾讯云 TRTC SDK（Windows C++ 版本）集成到 MFC 项目中。



开发环境要求

操作系统：Windows 7及以上版本。
开发环境：Visual Studio 2010及以上版本，推荐使用 Visual Studio 2015。

通过 MFC 项目集成 C++ SDK

本节通过创建一个简单的 MFC 项目，介绍如何在 Visual Studio 工程中集成 C++ SDK。

步骤1：下载 SDK

下载 SDK，解压并打开，本文示例中，您只需要引用 SDK 目录下 C++ 版的 SDK 文件即可，以64位为例，其 SDK 位置为 `./SDK/CPlusPlus/Win64/` 下，主要包含以下几个部分：

目录名	说明
include	带有详细接口注释的 API 头文件
lib	编译用的 .lib 文件和运行时加载的 .dll 文件

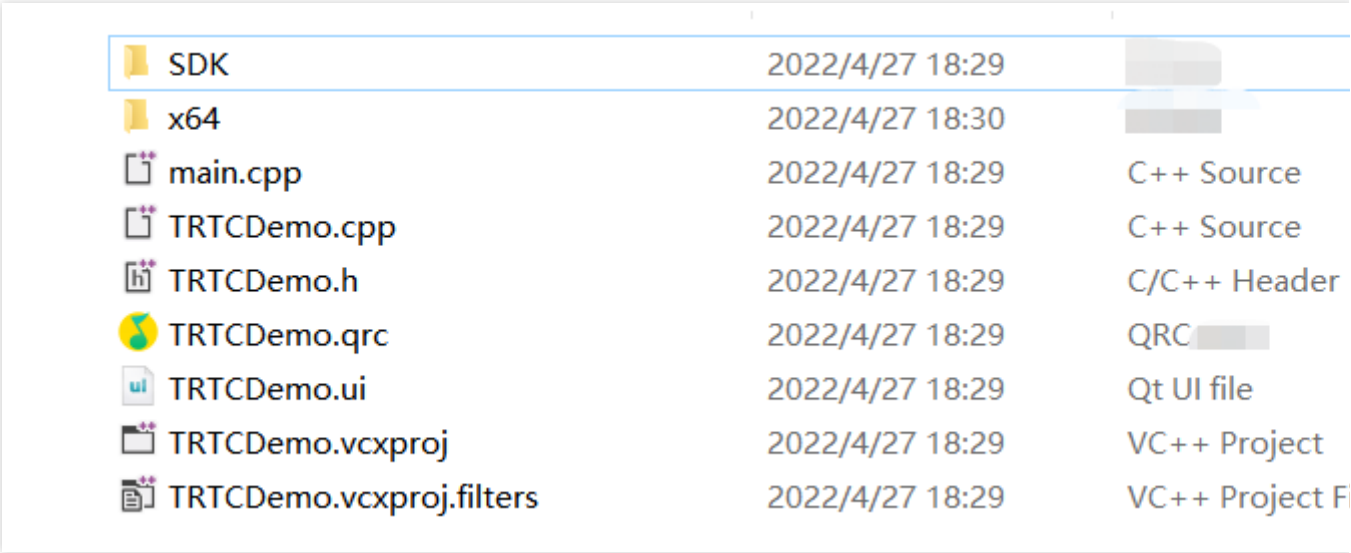
步骤2：新建工程

打开 Visual Studio，新建一个名字叫 TRTCDemo 的 MFC 应用程序。
为了便于介绍如何快速集成，在向导的**应用程序类型**页面，我们选择比较简单的**基于对话框**类型。
其他的向导配置，请选择默认的配置即可。

步骤3：拷贝文件

将解压后的 SDK 文件夹拷贝到 `TRTCDemo.vcxproj` 所在目录下，如下图所示：
说明：

当前只需要 C++ SDK，可以将 SDK 路径下的 CSharp 目录删除。



步骤4：修改工程配置

打开 TRTCDemo 属性页，在**解决方案资源管理器 > TRTCDemo 工程的右键菜单 > 属性**，请按照以下步骤进行配置：

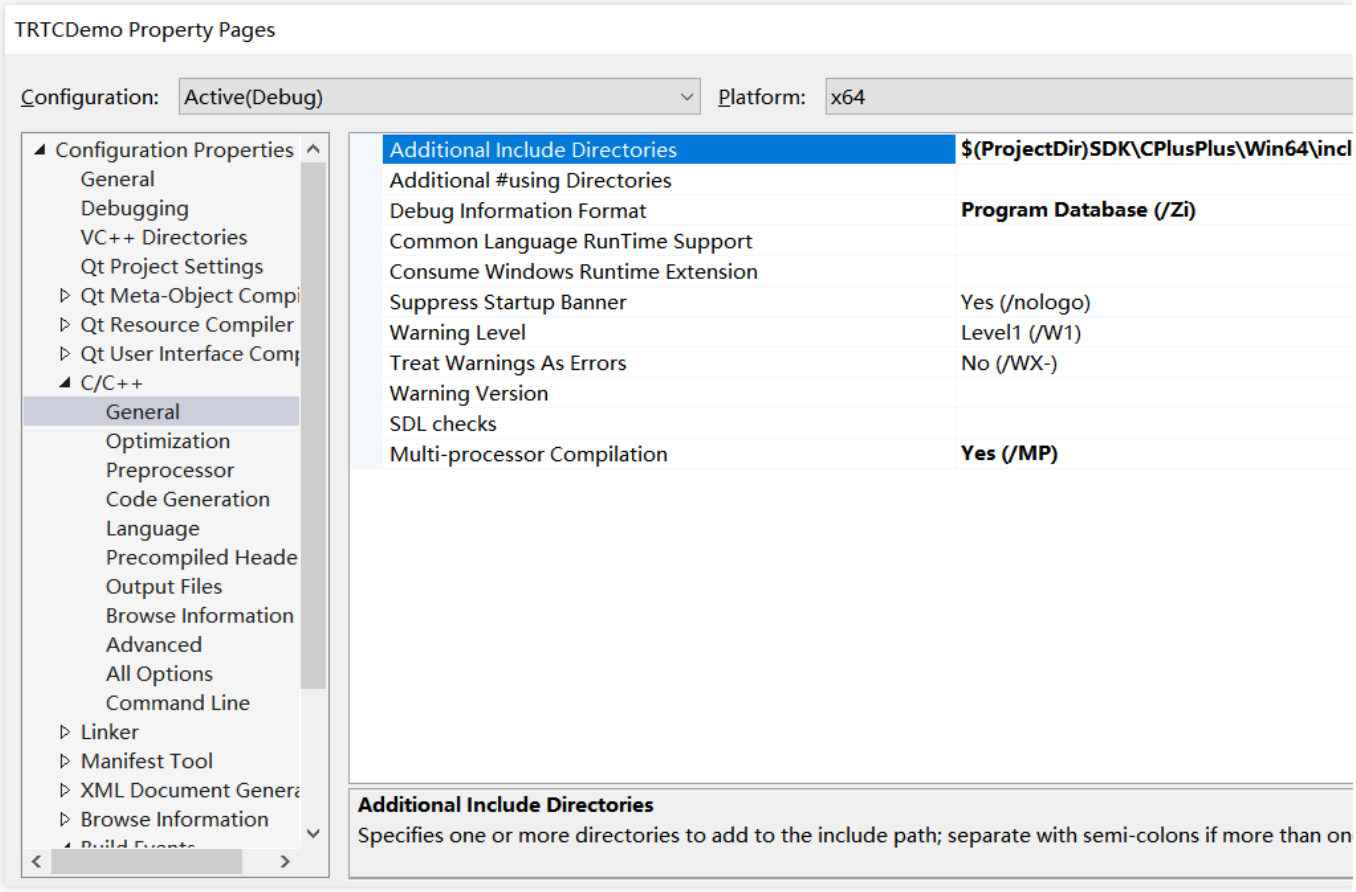
1. **添加包含目录：**

在**C/C++ > 常规 > 附件包含目录**，以64位为例，添加 SDK 头文件目录。

`$(ProjectDir)SDK\\CPlusPlus\\Win64\\include` 和 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include\\TRTC`，如下图所示：

说明：

如果为32位，则需要将 SDK 头文件目录设为 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include` 和 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include\\TRTC`。

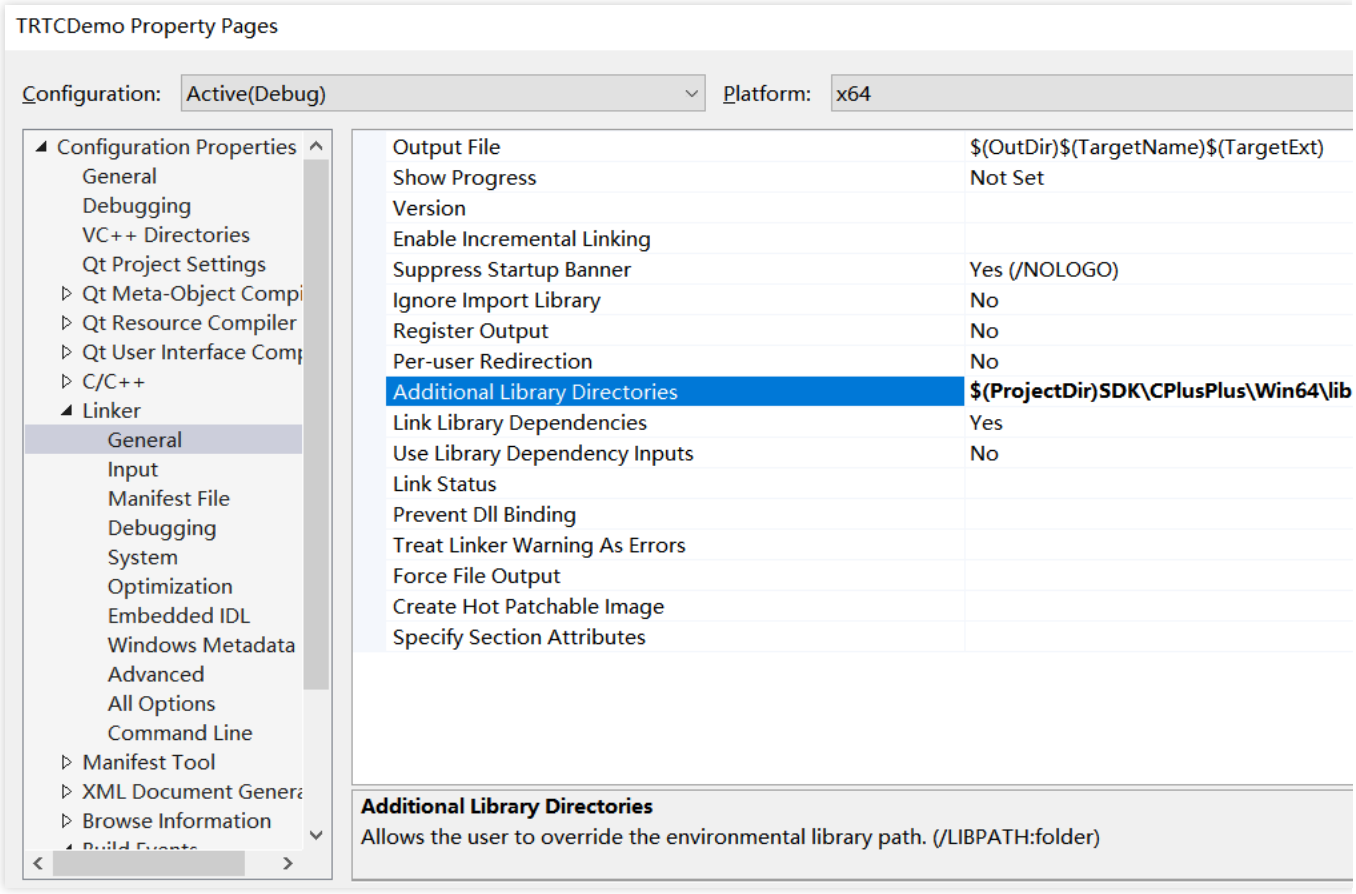


2. **添加库目录：**

在**链接器 > 常规 > 附加库目录**，添加 SDK 库目录 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\lib`，如下图所示：

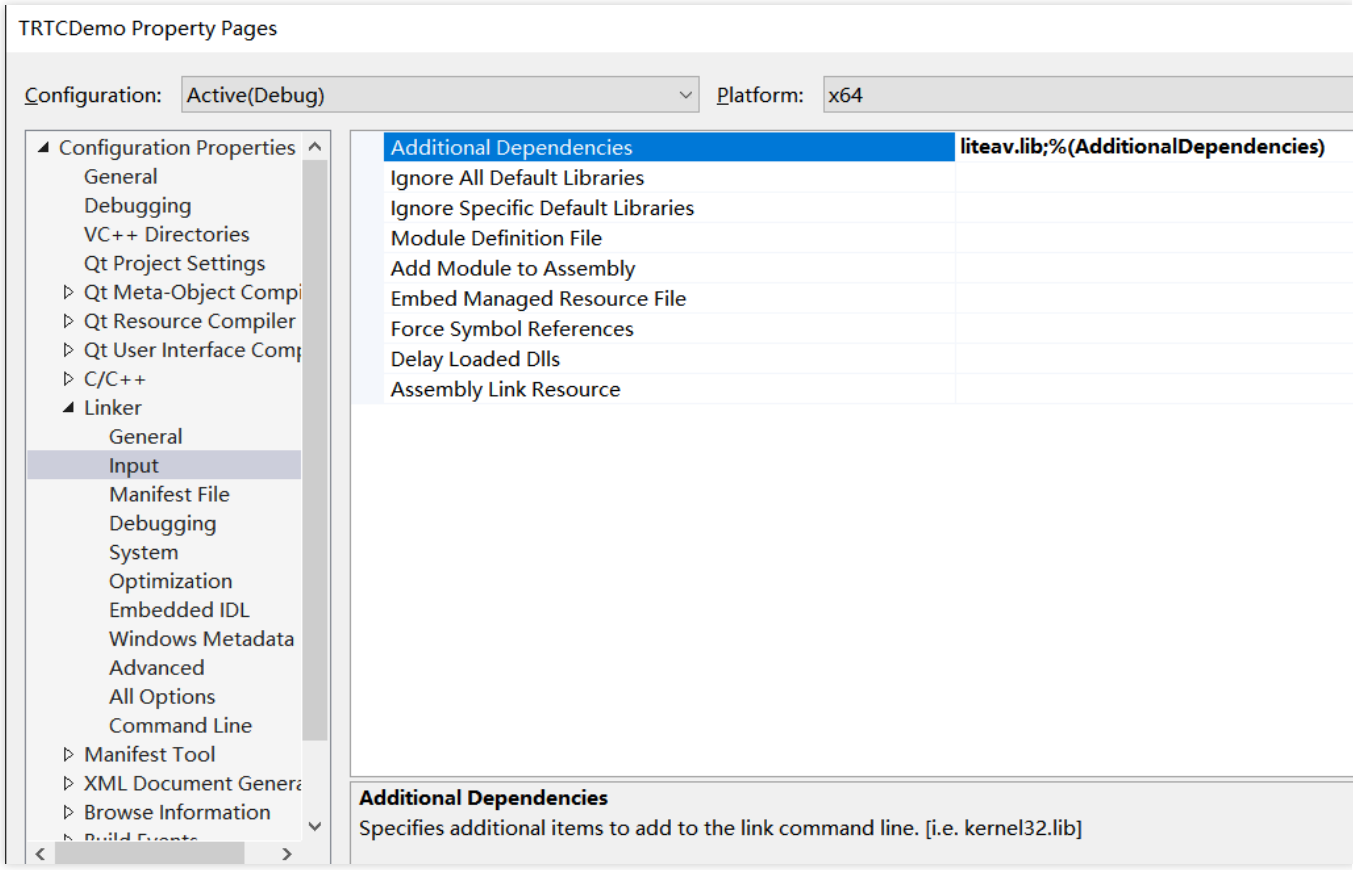
说明：

如果为32位，则需要将 SDK 库目录设为 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\lib` 。



3. **添加库文件：**

在**链接器** > **输入** > **附加依赖项**，添加 SDK 库文件 `liteav.lib`，如下图所示：

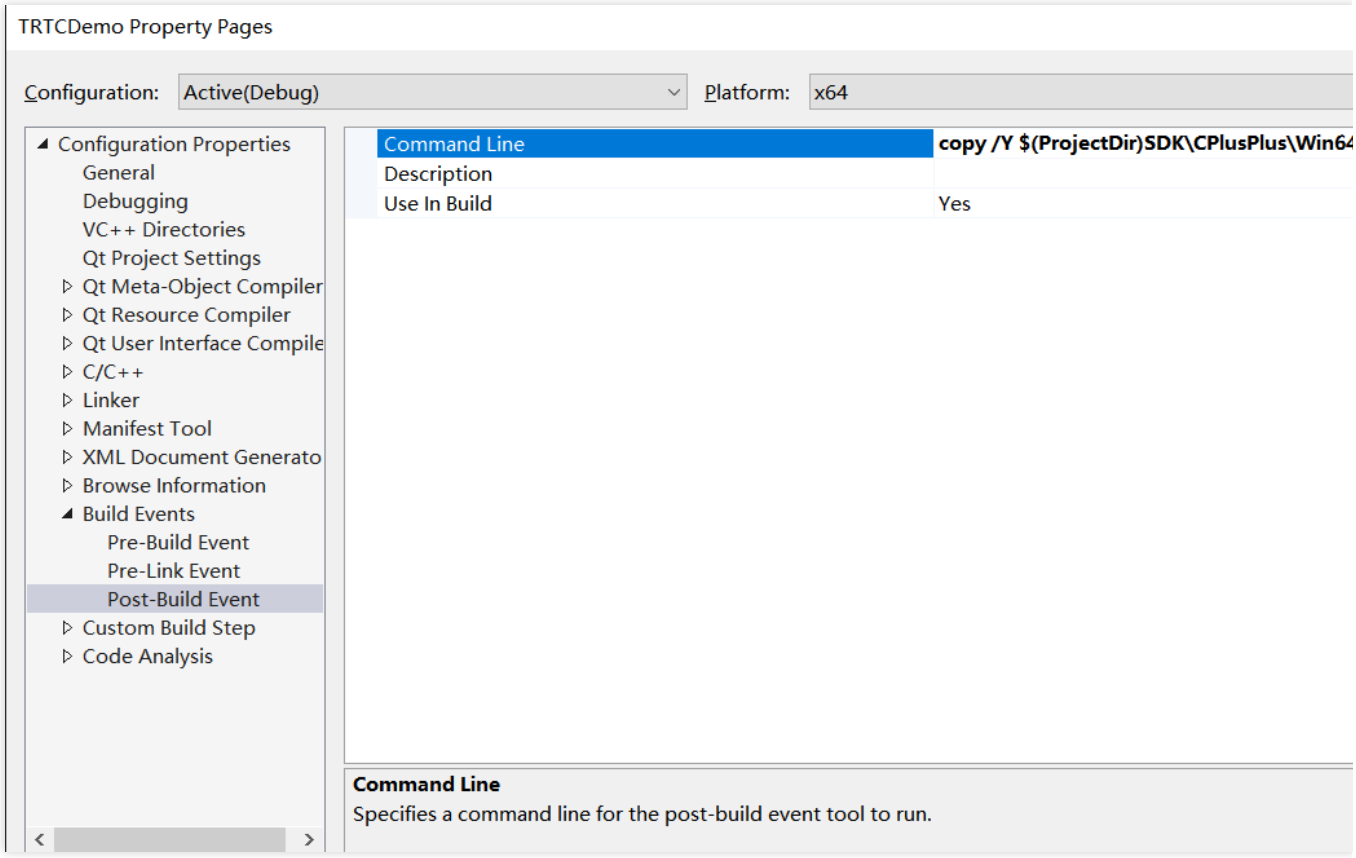


4. **添加 copy 命令：**

在**生成事件 > 后期生成事件 > 命令行**，添加拷贝命令 `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win64\\lib\\*.dll  $(OutDir)` ，能够在编译完成后，自动将 SDK 的 .dll 文件拷贝到程序的运行目录下，如下图所示：

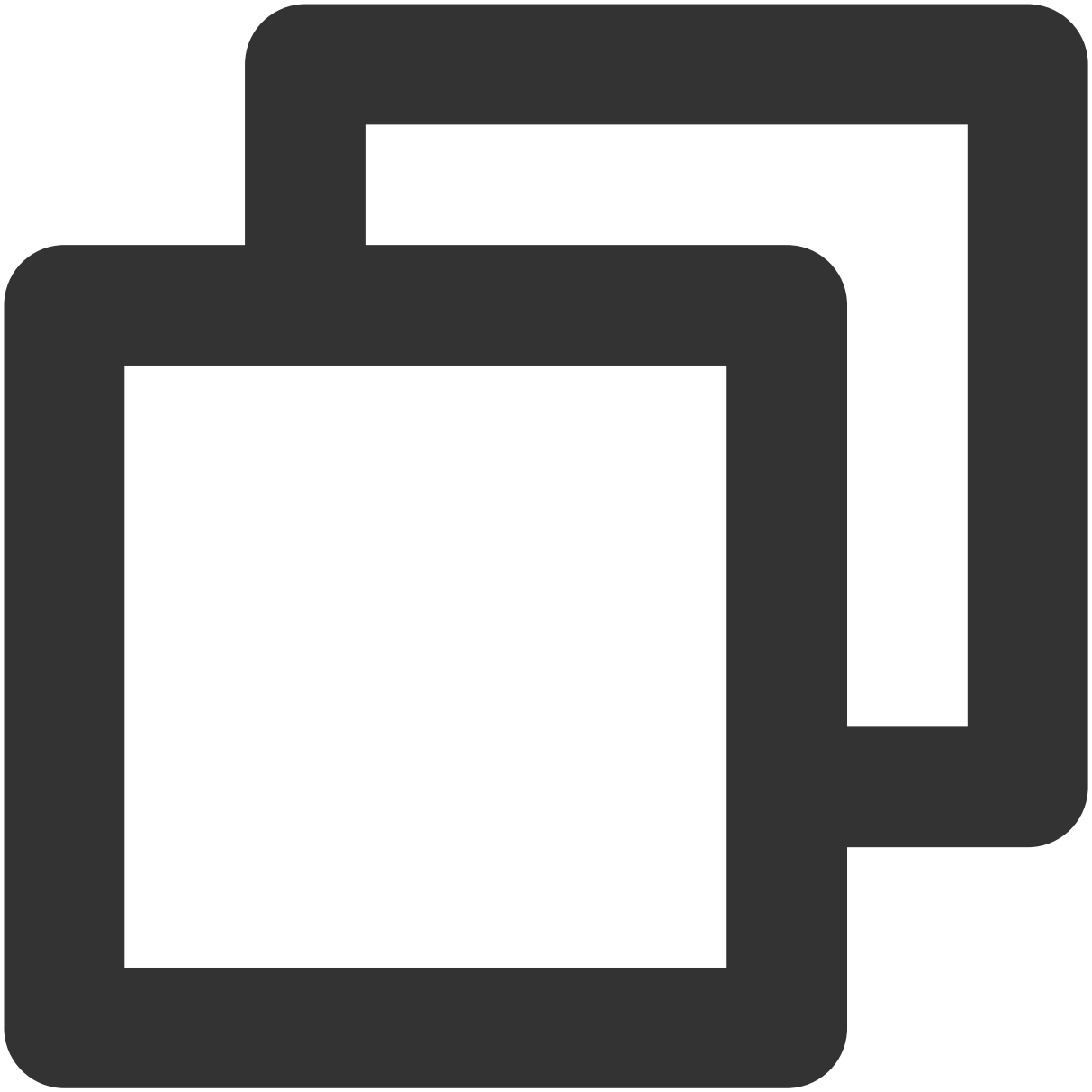
说明：

如果为32位，则添加拷贝命令为 `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win32\\lib\\*.dll $(OutDir)` 。



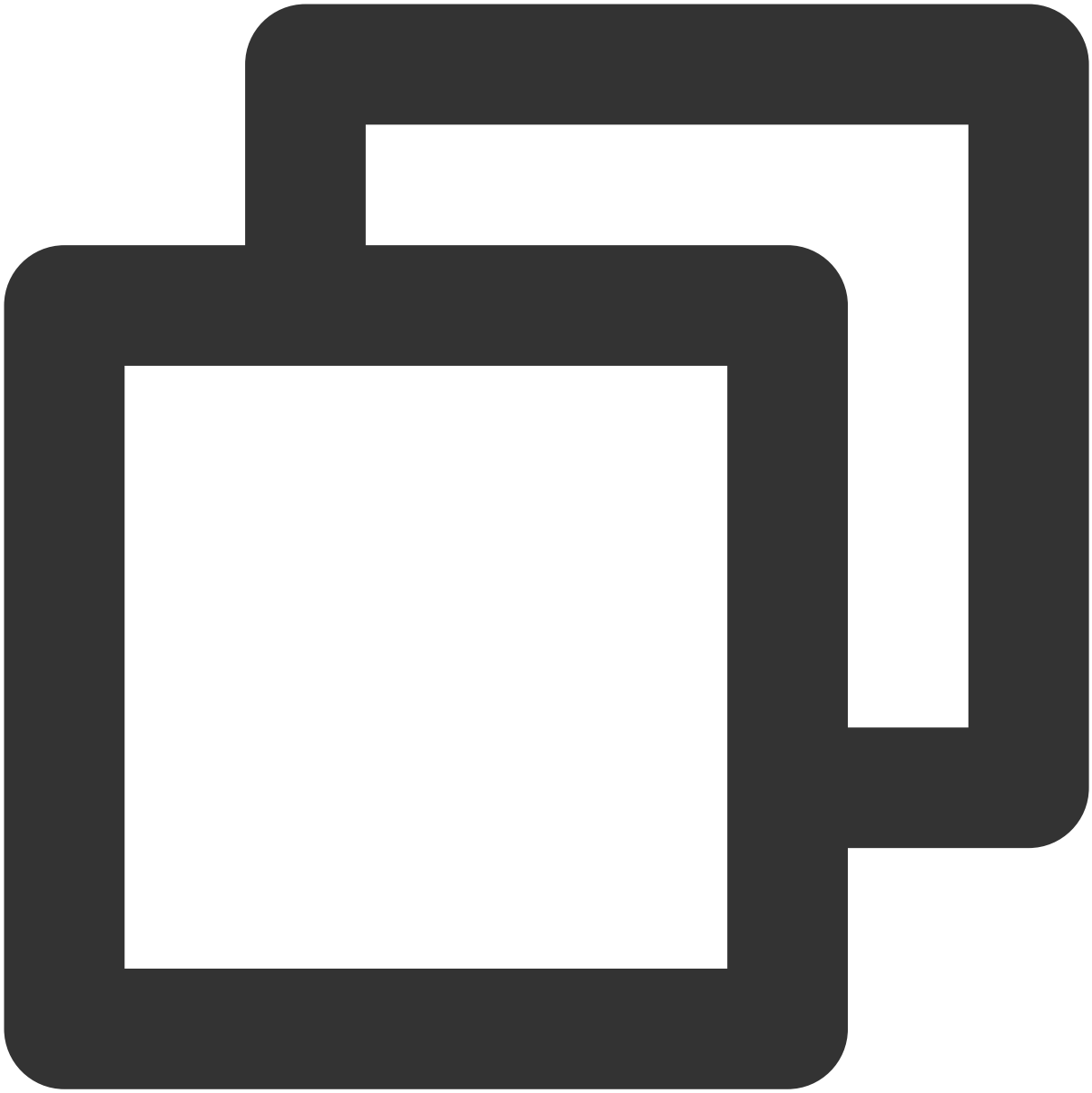
步骤5：打印 SDK 版本号

1. 在 TRTCDemoDlg.cpp 文件顶部增加头文件引入，代码如下：



```
#include "ITRTCCloud.h"
```

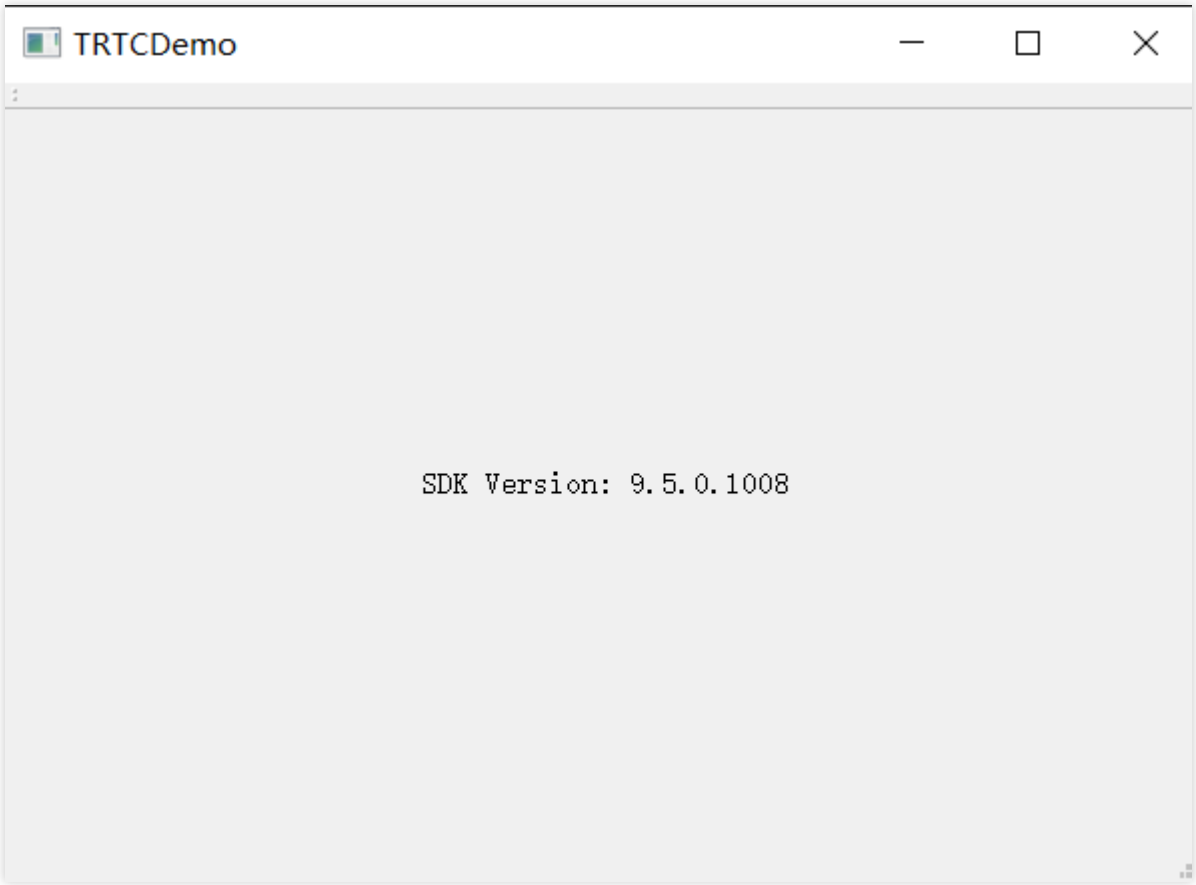
2. 在 `CTRTCDemoDlg::OnInitDialog` 函数中，添加下面的测试代码：



```
ITRTCCloud * pTRTCCloud = getTRTCShareInstance();
CString szText;
szText.Format(L"SDK version: %hs", pTRTCCloud->getSDKVersion());

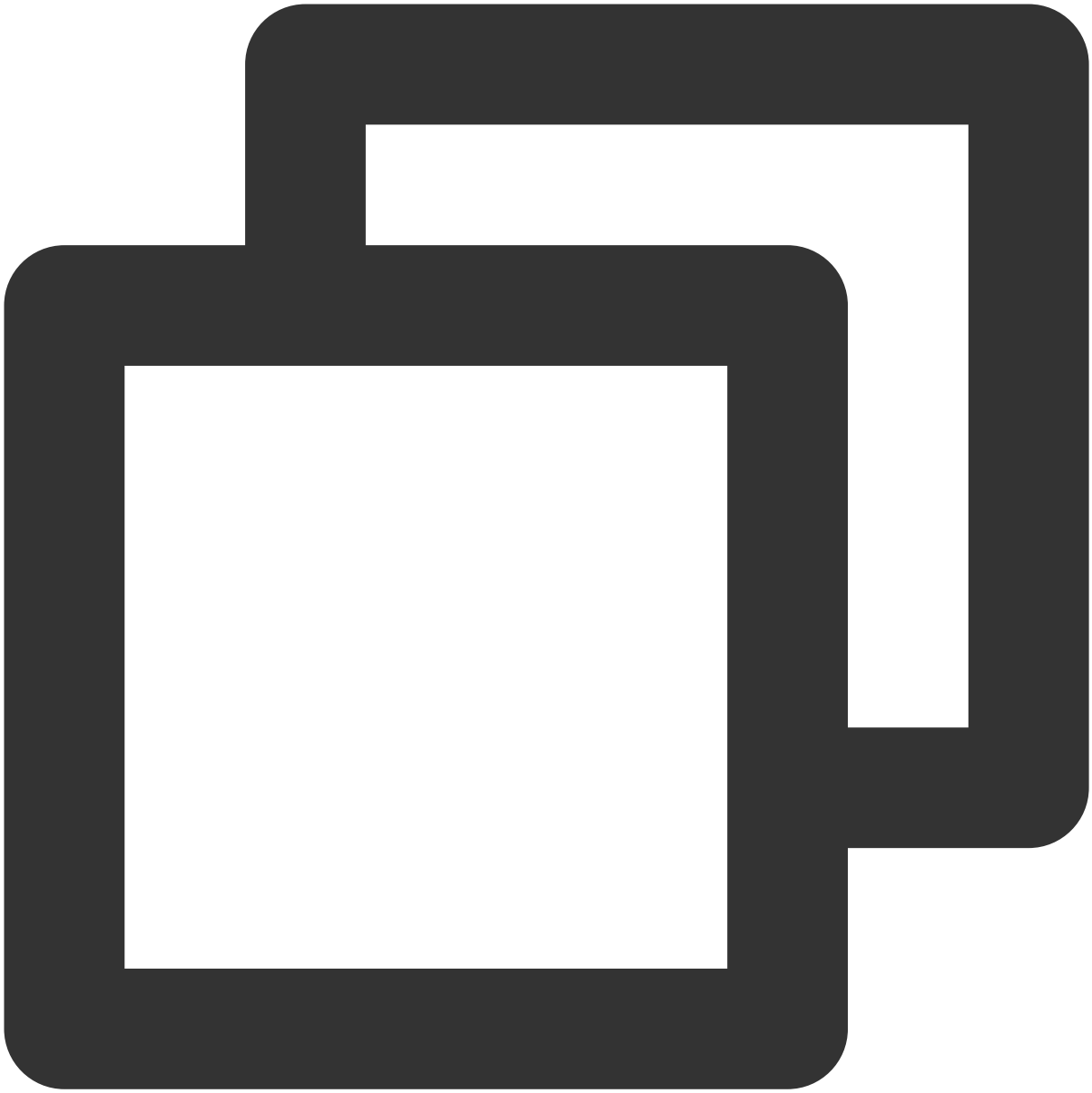
CWnd *pStatic = GetDlgItem(IDC_STATIC);
pStatic->SetWindowTextW(szText);
```

3. 按键盘 F5 运行，打印 SDK 的版本号，如下图所示：



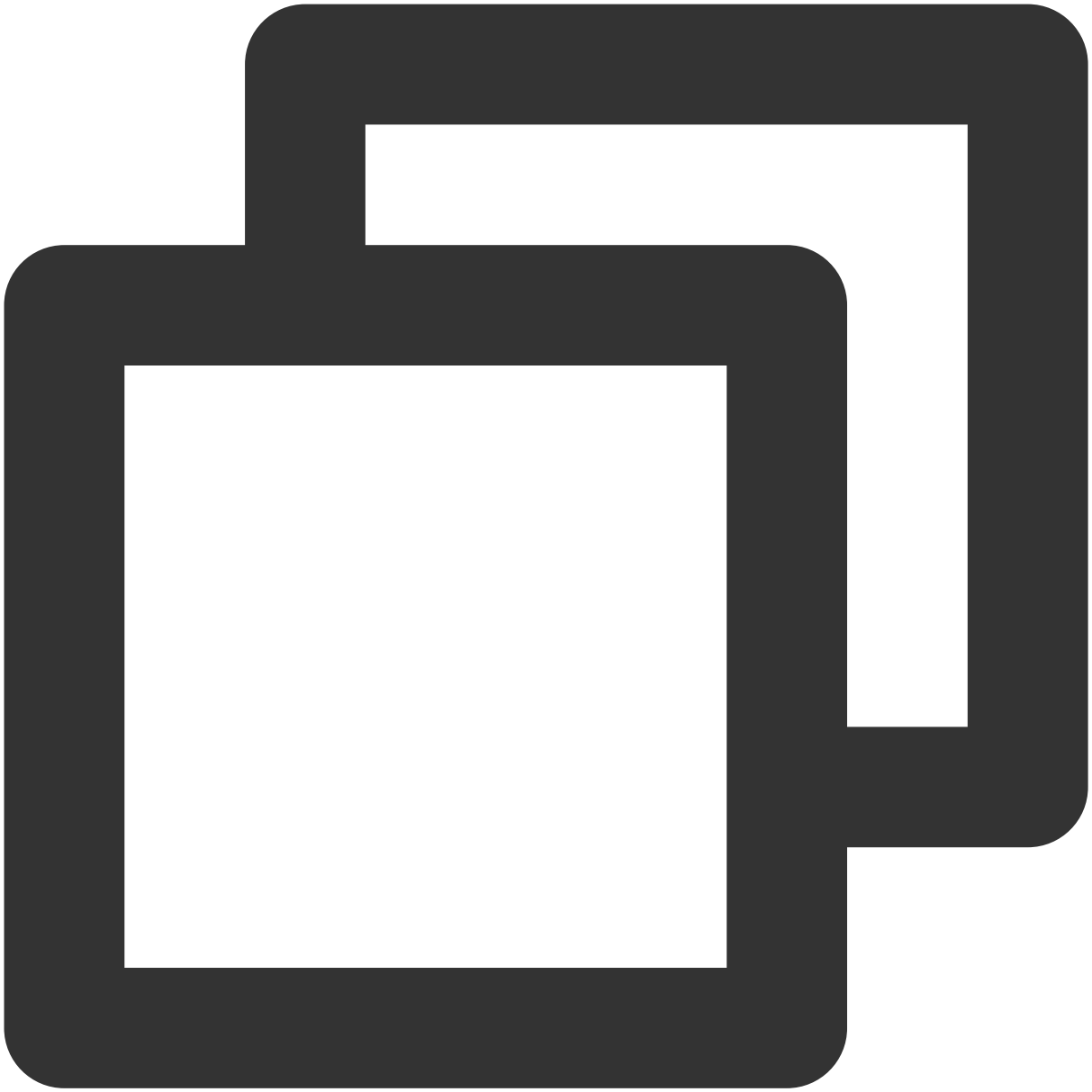
常见问题

若出现以下错误，请按照前面的工程配置，检查 SDK 头文件的目录是否正确添加。



```
fatal error C1083: 无法打开包括文件: “TRTCCloud.h”: No such file or directory
```

若出现以下错误，请按照前面的工程配置，检查 SDK 库目录和库文件是否正确添加。



```
error LNK2019: 无法解析的外部符号 "__declspec(dllimport) public: static class TXString
```

Web

最近更新时间：2023-08-22 17:26:09

本文主要介绍如何快速地将腾讯云 TRTC Web SDK 集成到您的项目中。



支持的平台

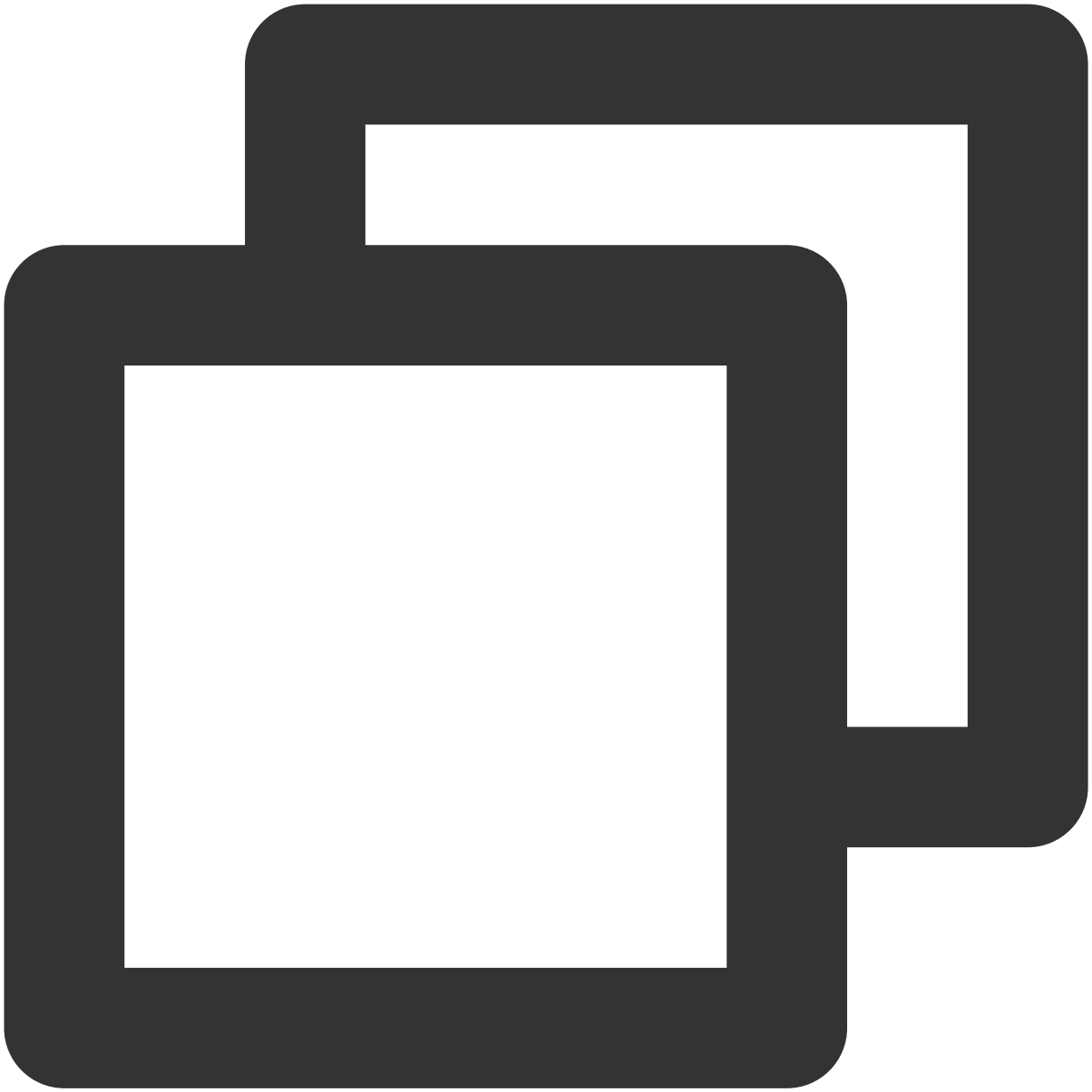
腾讯云 TRTC Web SDK 支持市场上主流的浏览器，**Chrome、Edge、Firefox、Safari、Opera** 等。详细支持度表格请参见 支持的平台。

集成 SDK

SDK 提供了 UMD、ES Module 类型的模块，以及 TypeScript Type Definition 文件，满足在不同类型项目中集成。

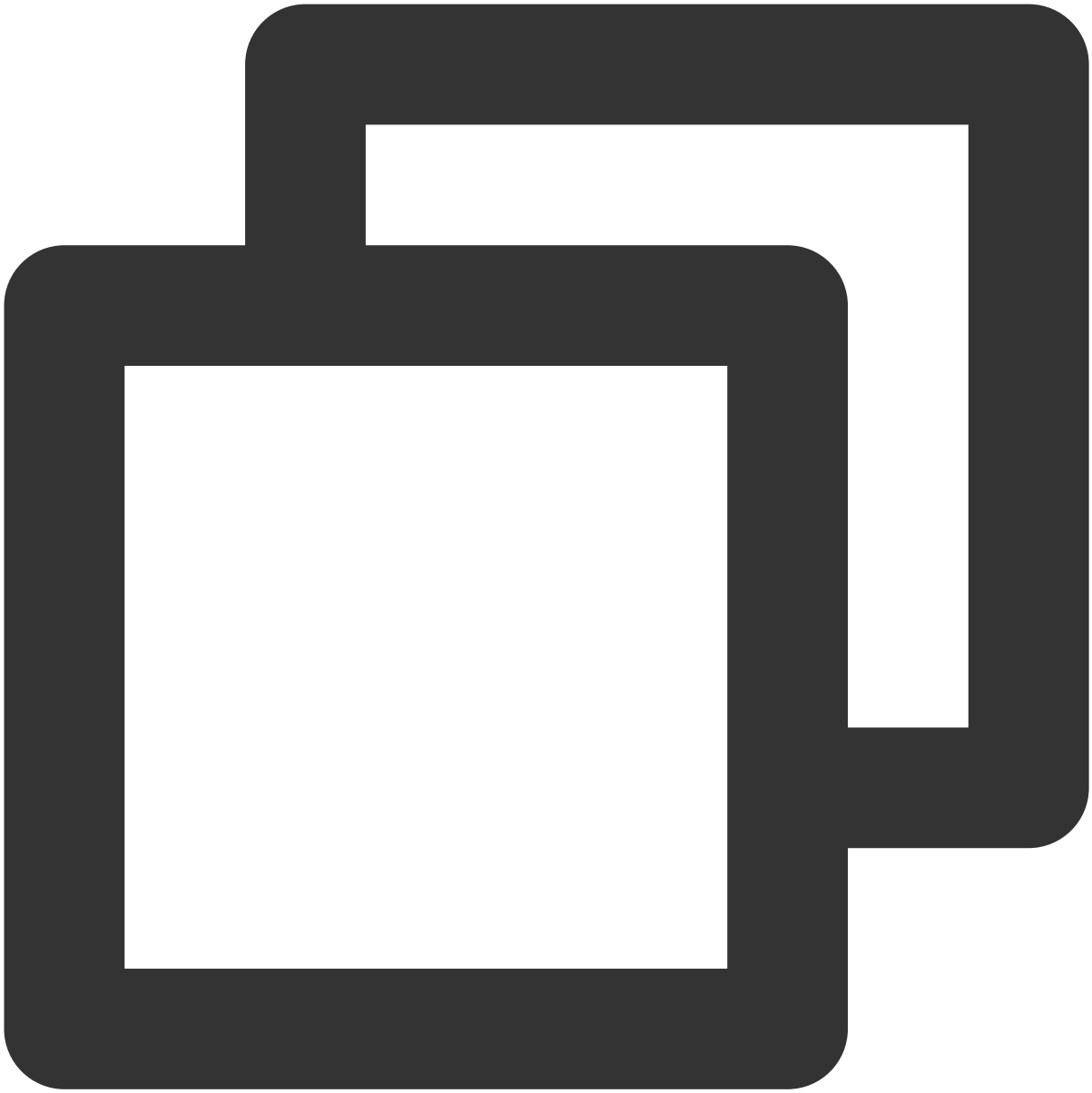
通过 npm 集成

1. 您可以在项目中使用 npm 安装 trtc-sdk-v5。



```
npm install trtc-sdk-v5 --save
```

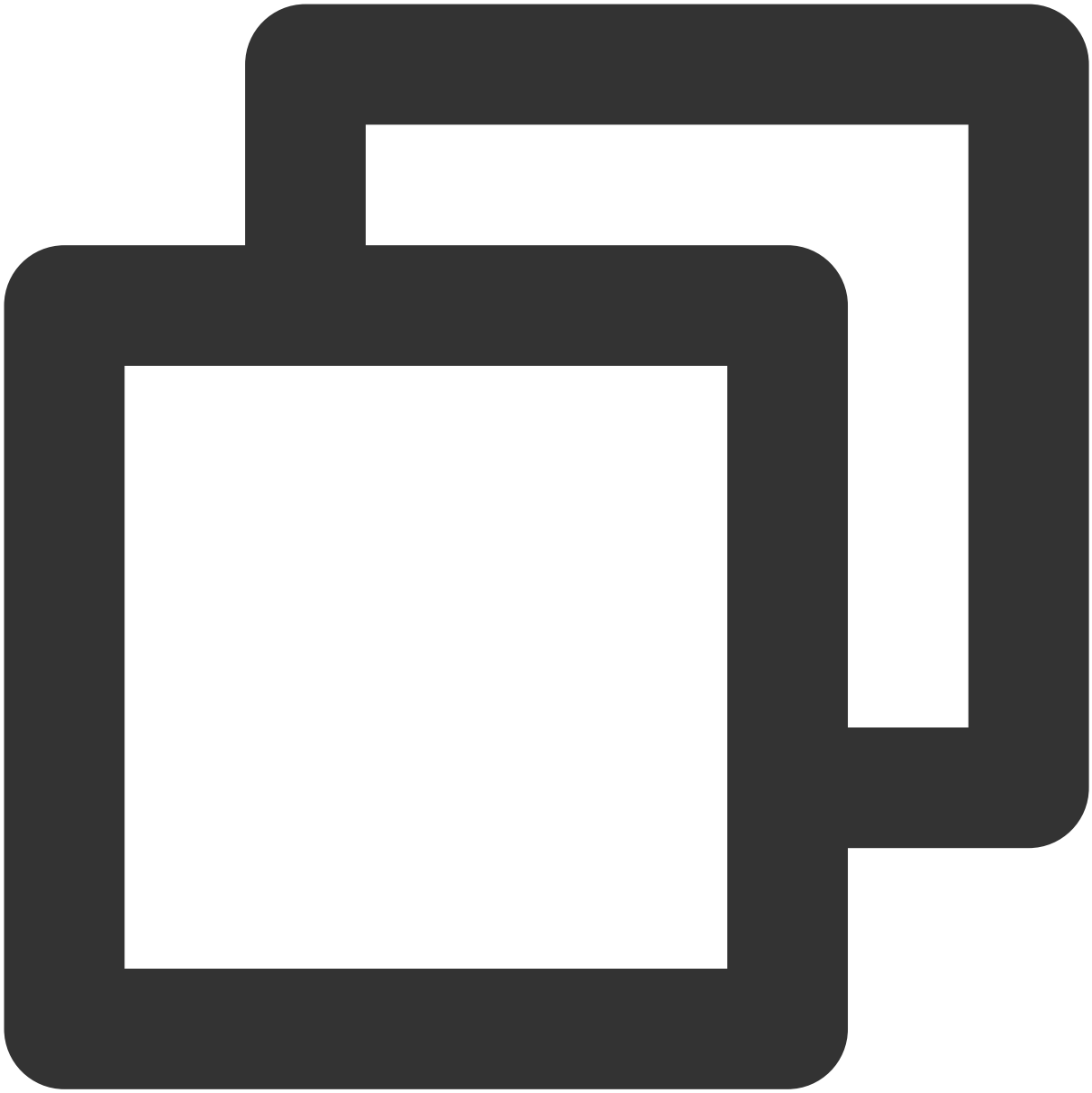
2. 在项目脚本里导入模块。



```
import TRTC from 'trtc-sdk-v5';
```

通过 script 集成

1. 在 Github 下载 trtc.js 文件。
2. 在您的 Web 页面中添加如下代码：



```
<script src="trtc.js"></script>
```

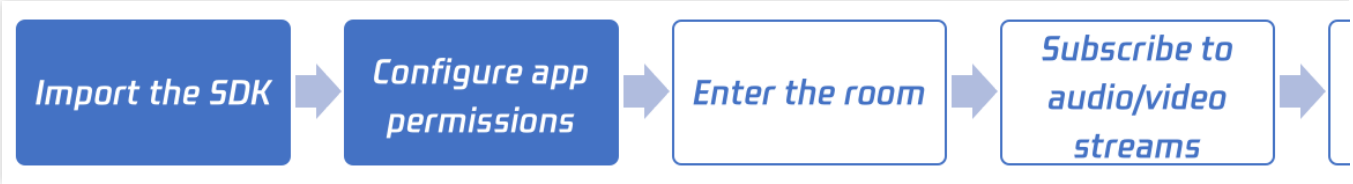
进入房间

在导入 SDK 后，您需要参考进入房间 教程实现后续流程。

Electron

最近更新时间：2024-06-07 11:20:49

本文主要介绍如何快速地将腾讯云 TRTC Electron SDK 集成到您的项目中，只要按照如下步骤进行配置，就可以完成 SDK 的集成工作。



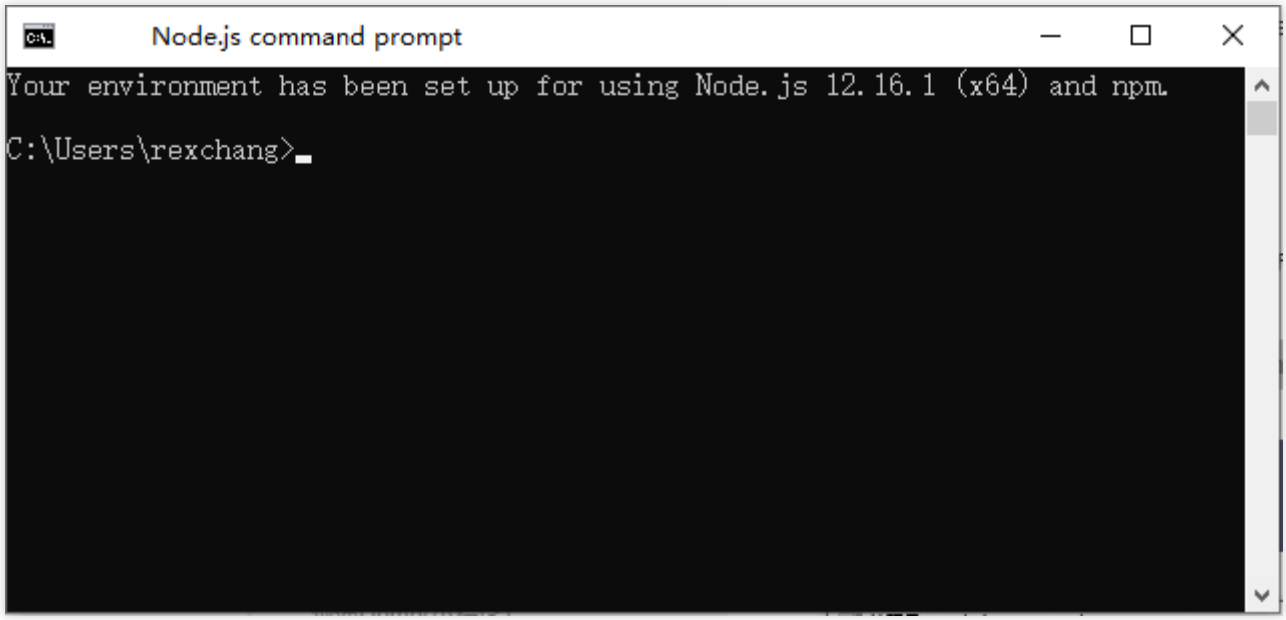
支持的平台

Windows（PC）
Mac

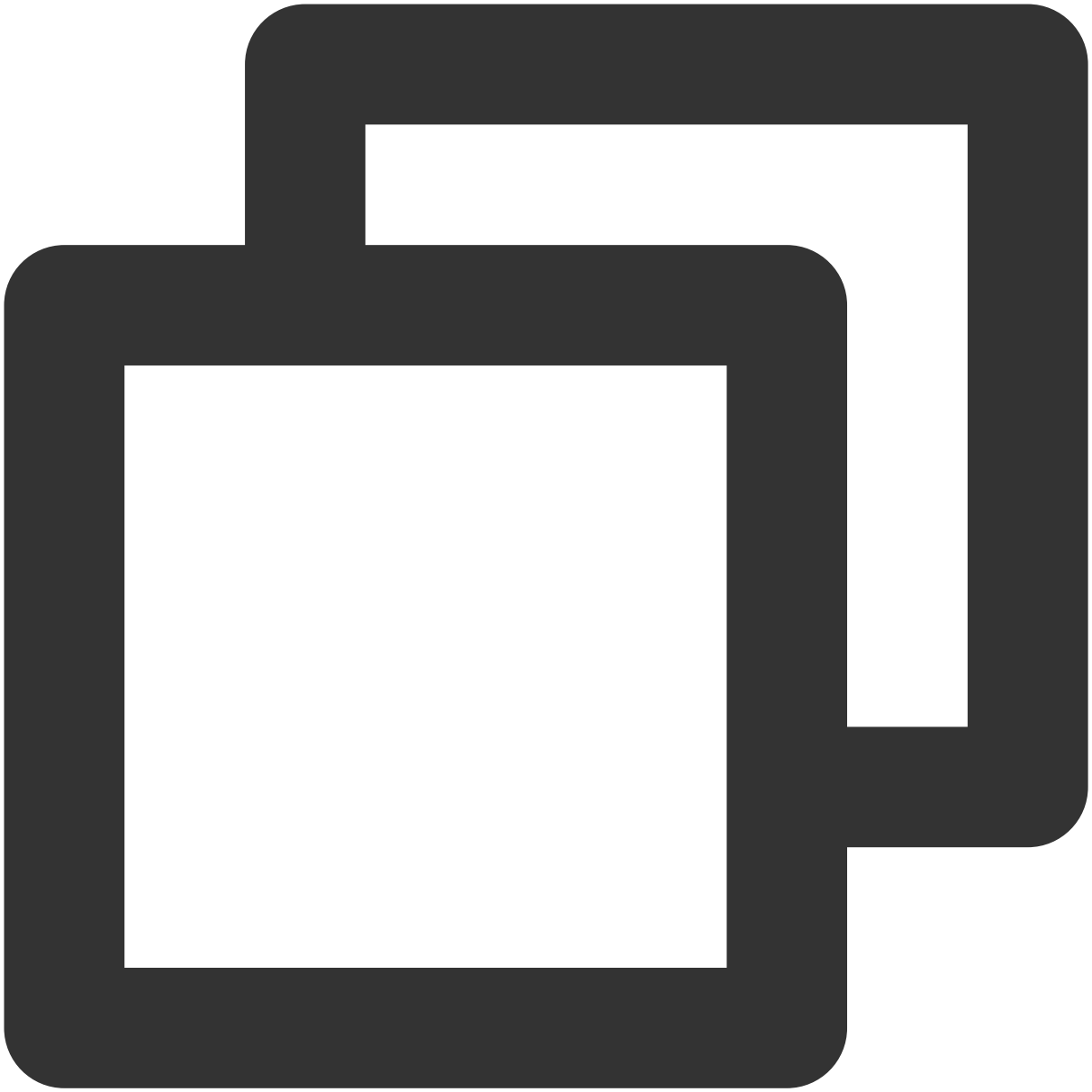
导入 SDK

步骤1. 安装 Node.js

Windows 平台安装指引
MacOS 平台安装指引
1. 根据 Windows 操作系统选择下载最新版本的 Node.js 安装包 `Windows Installer (.msi) 64-bit` 。
2. 打开应用程序列表中的 Node.js command prompt，启动命令行窗口，用于输入后续步骤中的各项命令。

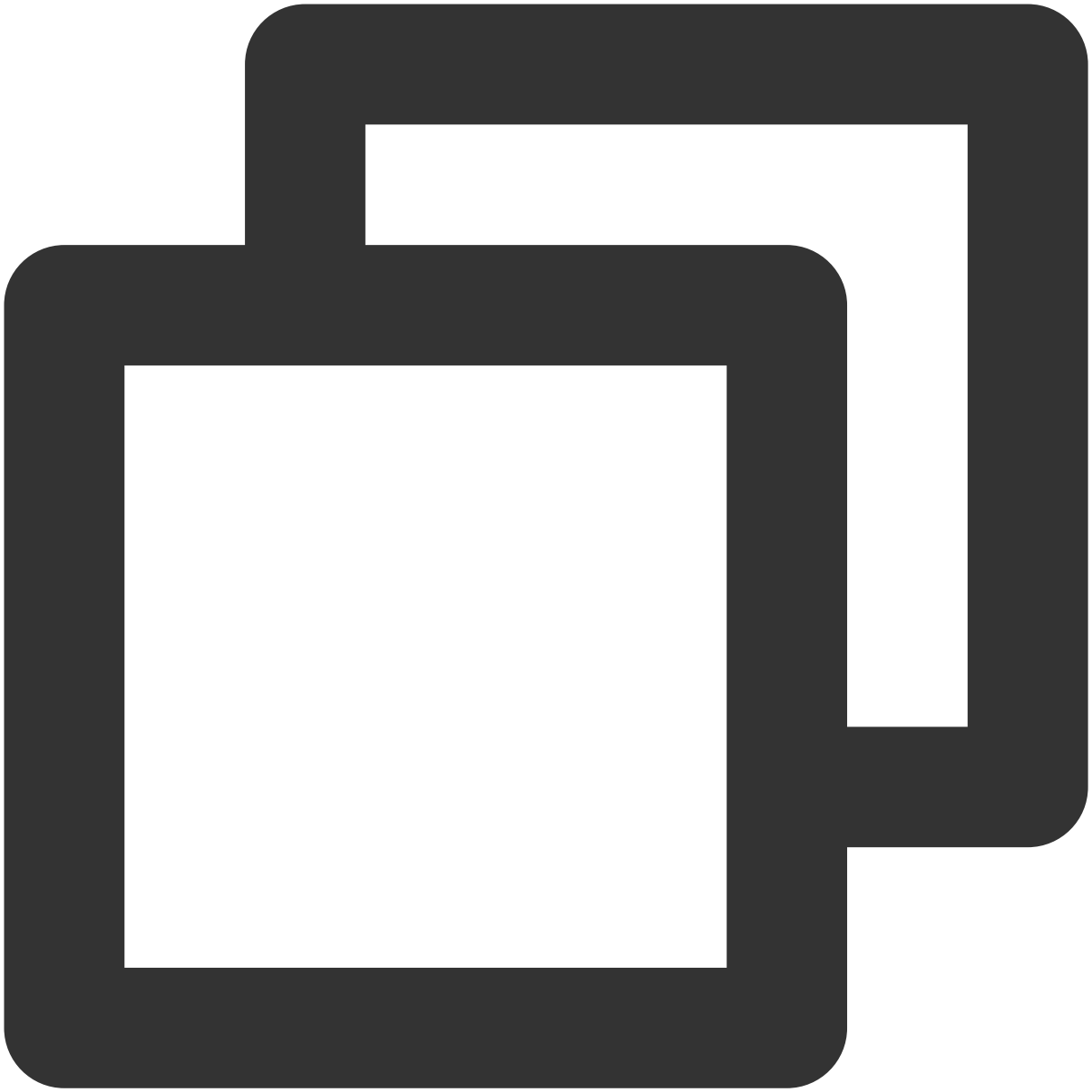


1. 打开终端（Terminal）窗口，执行以下命令安装 Homebrew，如果已经安装请跳过此步骤。



```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/i
```

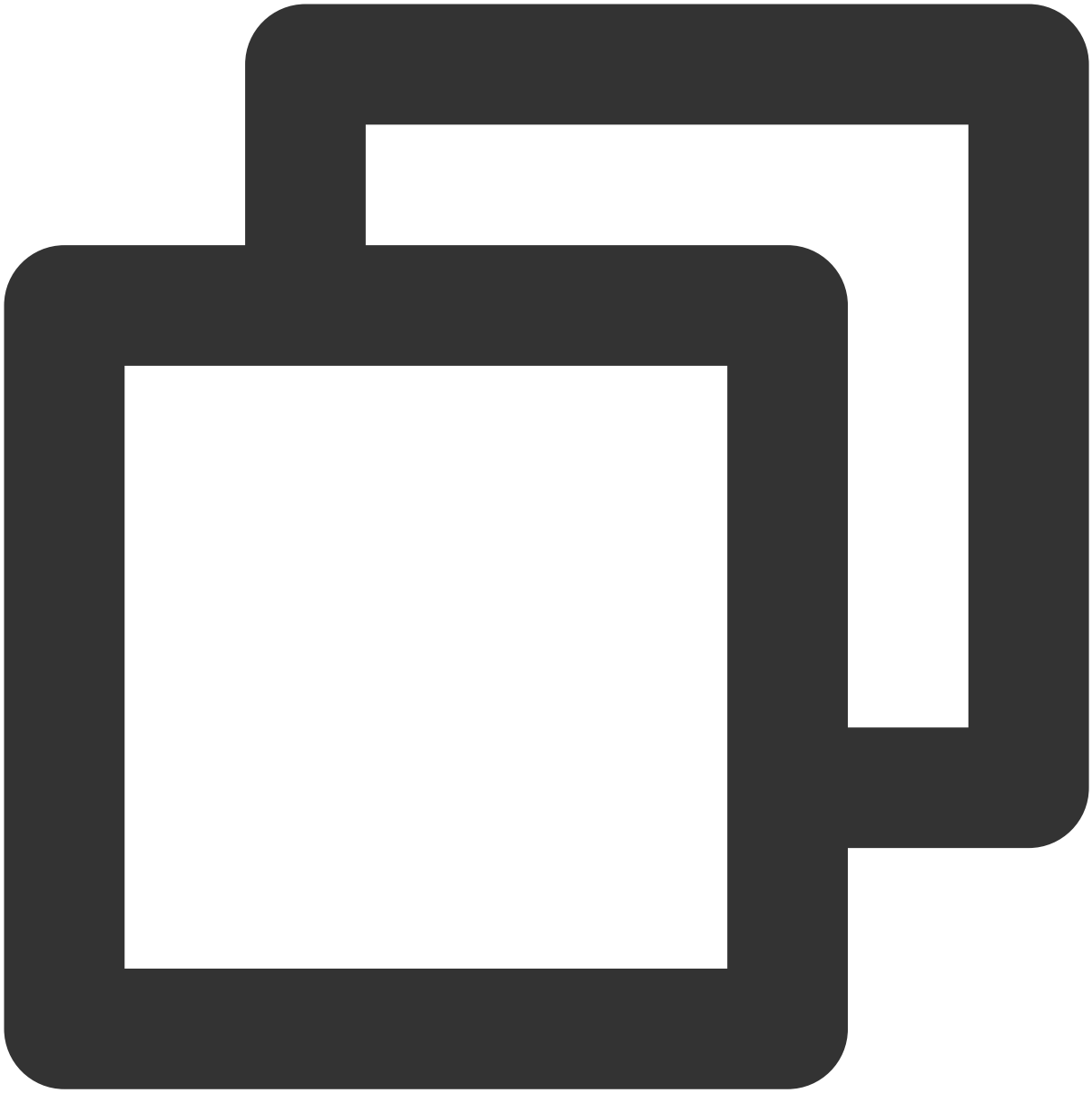
2. 执行以下命令，安装 Node.js，必须大于 10.0 版本。



```
$ brew install node
```

步骤2：安装 Electron

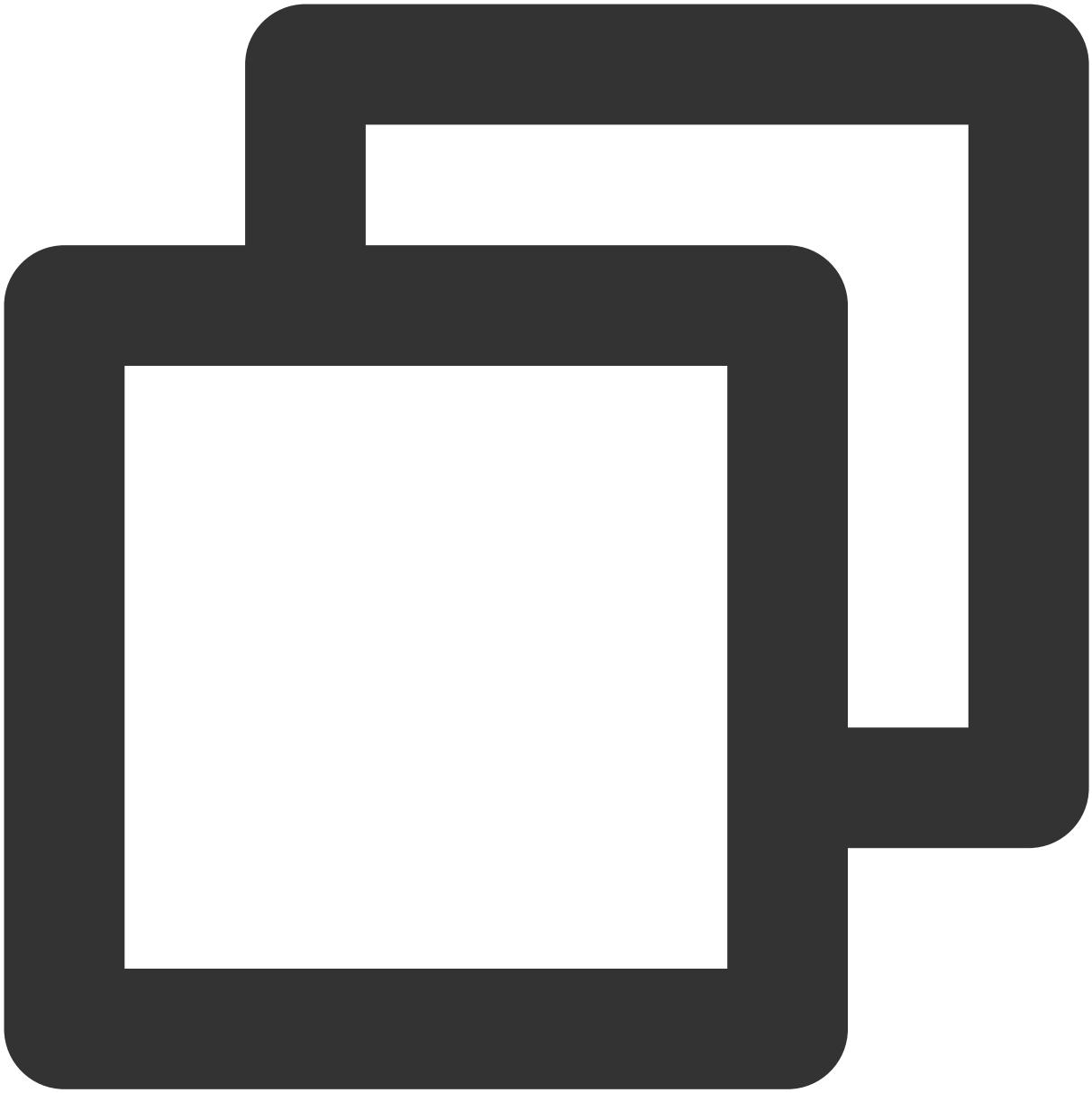
在命令行窗口中执行如下命令，安装 Electron，建议版本号 >= 4.0.0。



```
$ npm install electron@latest --save-dev
```

步骤3：安装 Electron 版的 TRTC SDK

1. 在您的 Electron 项目中使用 npm 命令安装 SDK 包：

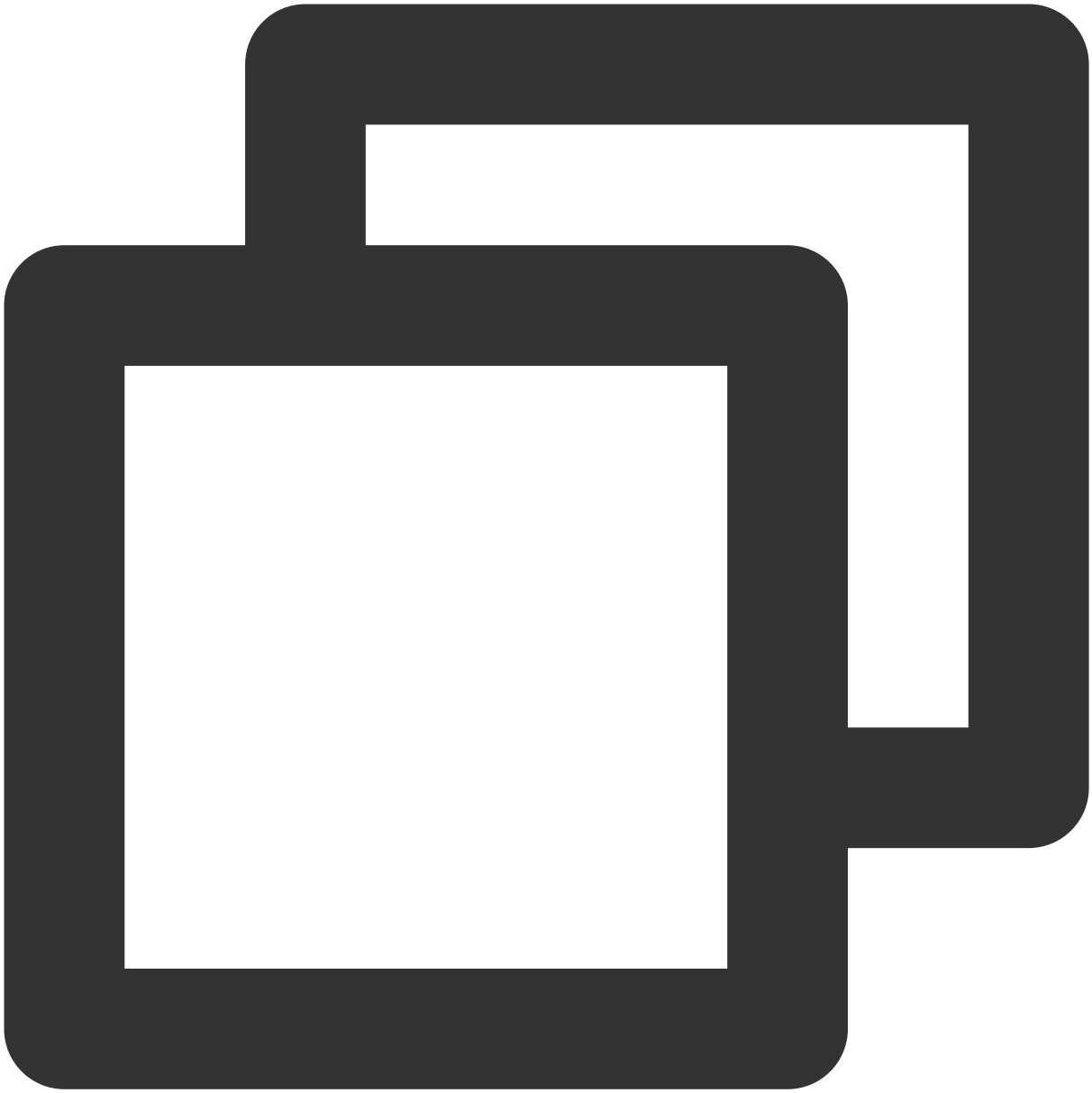


```
$ npm install trtc-electron-sdk@latest --save
```

说明：

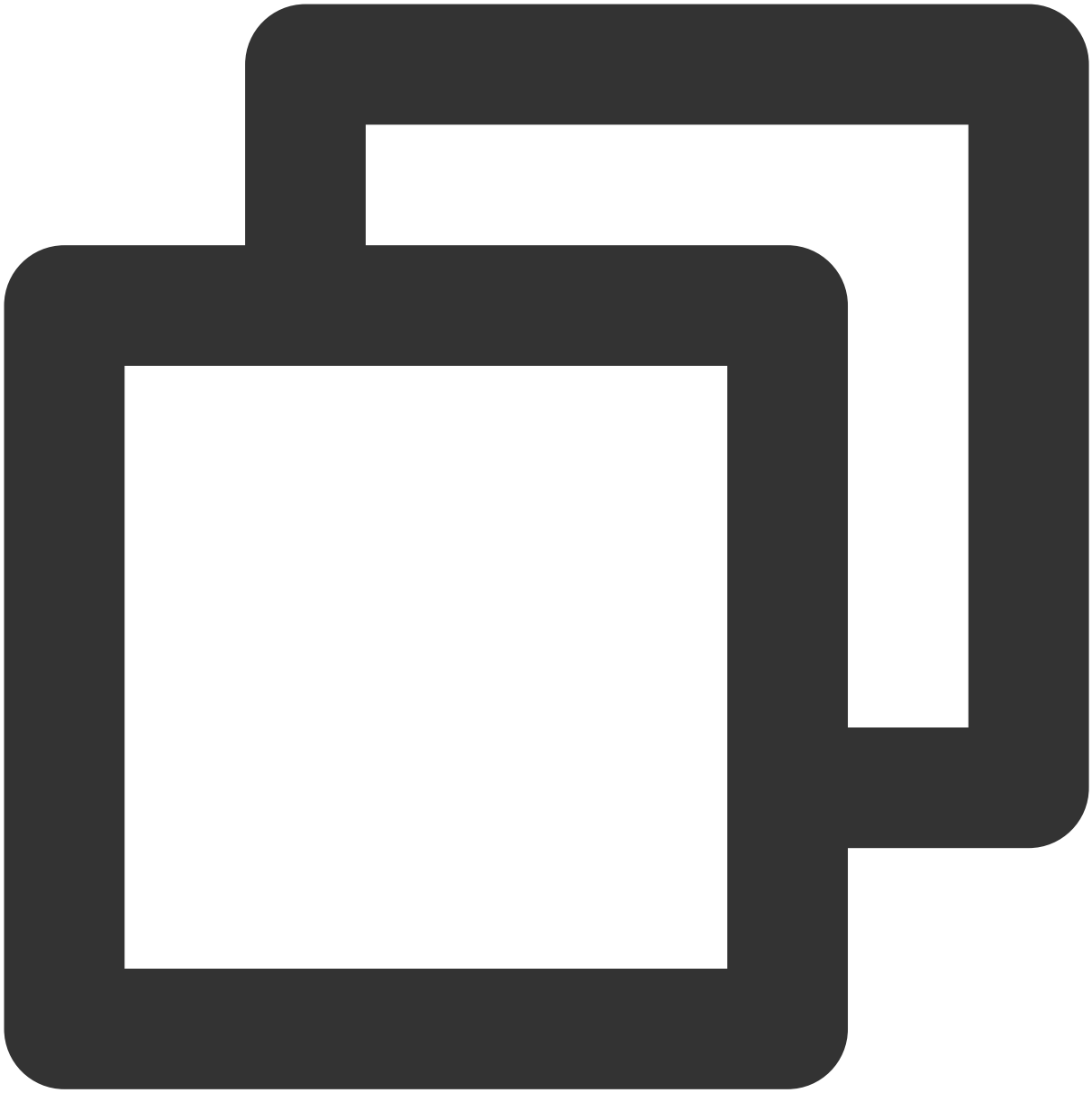
TRTC Electron SDK 最新版可在 trtc-electron-sdk 中查看。

2. 在项目脚本里引入模块并使用：



```
const TRTCCloud = require('trtc-electron-sdk').default;
// import TRTCCloud from 'trtc-electron-sdk';
this.rtcCloud = new TRTCCloud();
// 获取 SDK 版本号
this.rtcCloud.getSDKVersion();
```

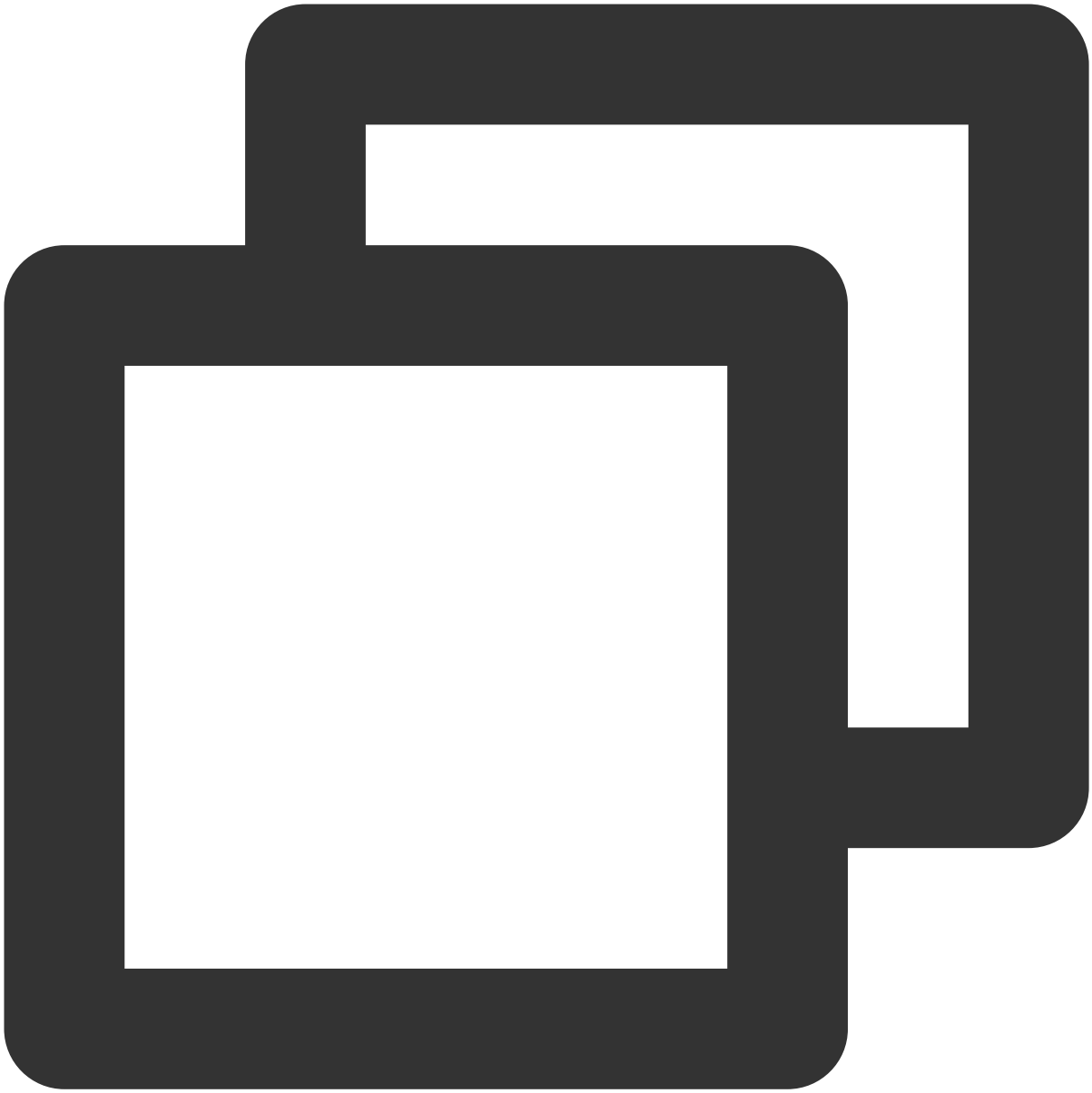
从v7.9.348起，TRTC Electron SDK 增加了 trtc.d.ts 文件，方便使用 TypeScript 的开发者：



```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud: TRTCCloud = new TRTCCloud();
// 获取 SDK 版本号
rtcCloud.getSDKVersion();
```

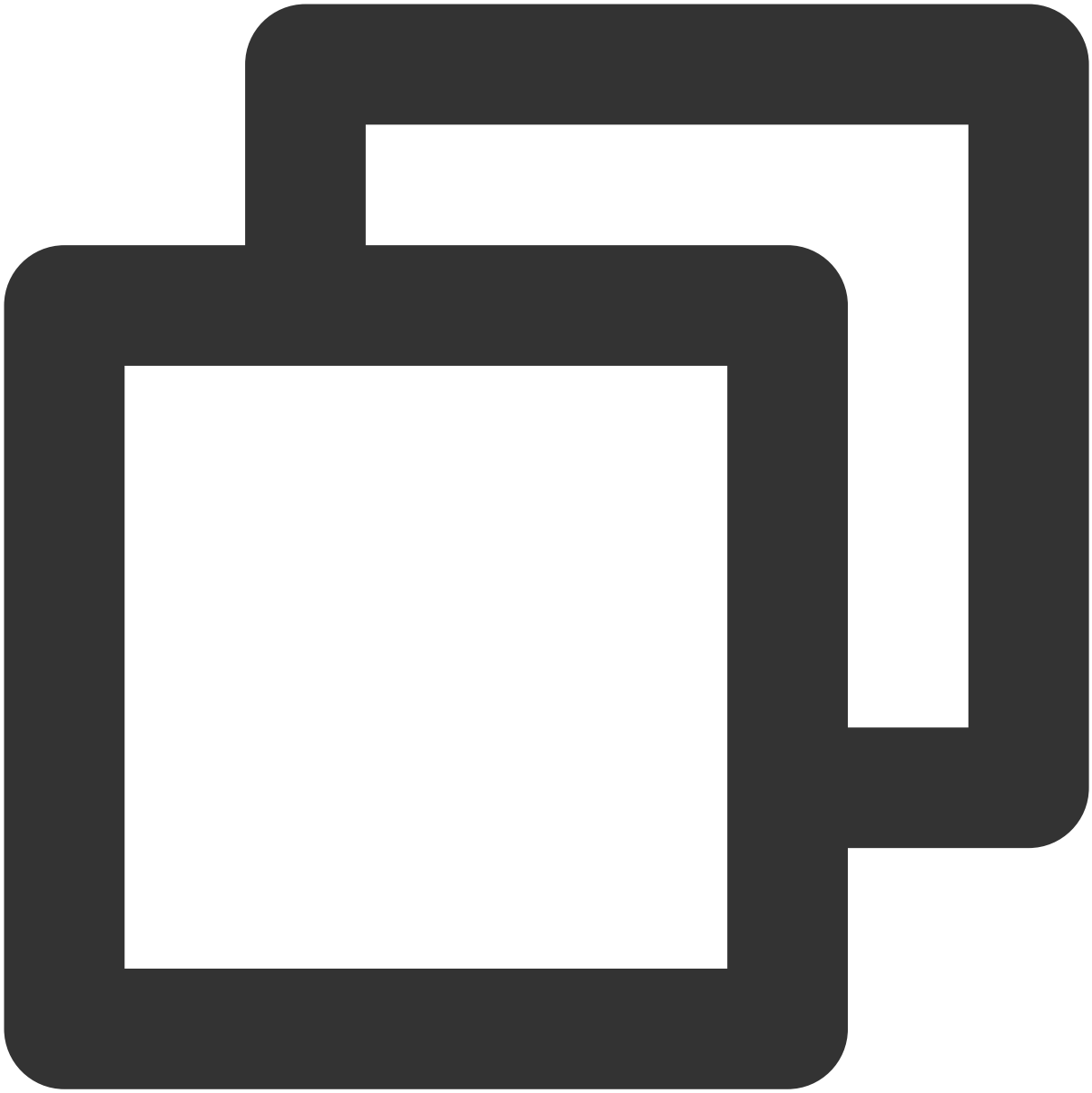
步骤4：打包可执行程序

安装打包工具：推荐使用打包工具 `electron-builder` 进行打包，您可以执行如下命令安装 `electron-builder` 。



```
$ npm install electron-builder@latest --save-dev
```

为了正确打包 Electron 版本的 TRTC SDK（也就是 `trtc_electron_sdk.node` 文件），您还需要执行如下命令以安装 `native-ext-loader` 工具。



```
$ npm install native-ext-loader@latest --save-dev
```

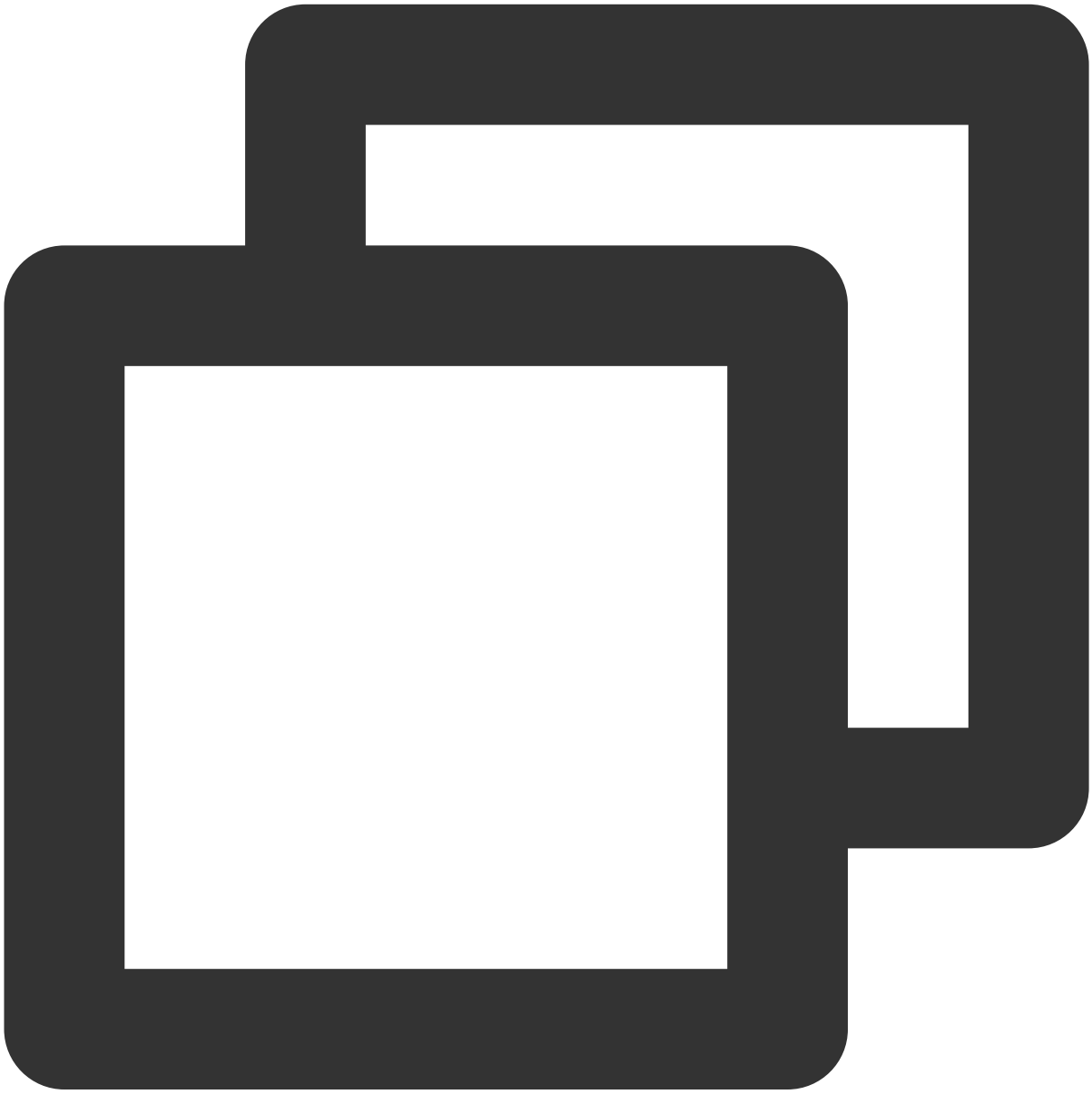
步骤5：修改打包配置，以 webpack.config.js 为例

`webpack.config.js` 包含了项目构建的配置信息， `webpack.config.js` 文件的位置如下：
通常情况下， `webpack.config.js` 位于项目的根目录。
使用 `create-react-app` 创建项目的情况下，此配置文件为 `node_modules/react-scripts/config/webpack.config.js` 。

使用 `vue-cli` 创建项目的情况下，webpack 的配置存放在 `vue.config.js` 配置中的 `configureWebpack` 属性中。

如您的工程文件经过了定制化，还请自行查找 webpack 配置。

1. 首先使 `webpack.config.js` 在构建时可以接收名为 `--target_platform` 的命令行参数，以使代码构建过程按不同的目标平台特点正确打包，在 `module.exports` 之前添加以下代码：



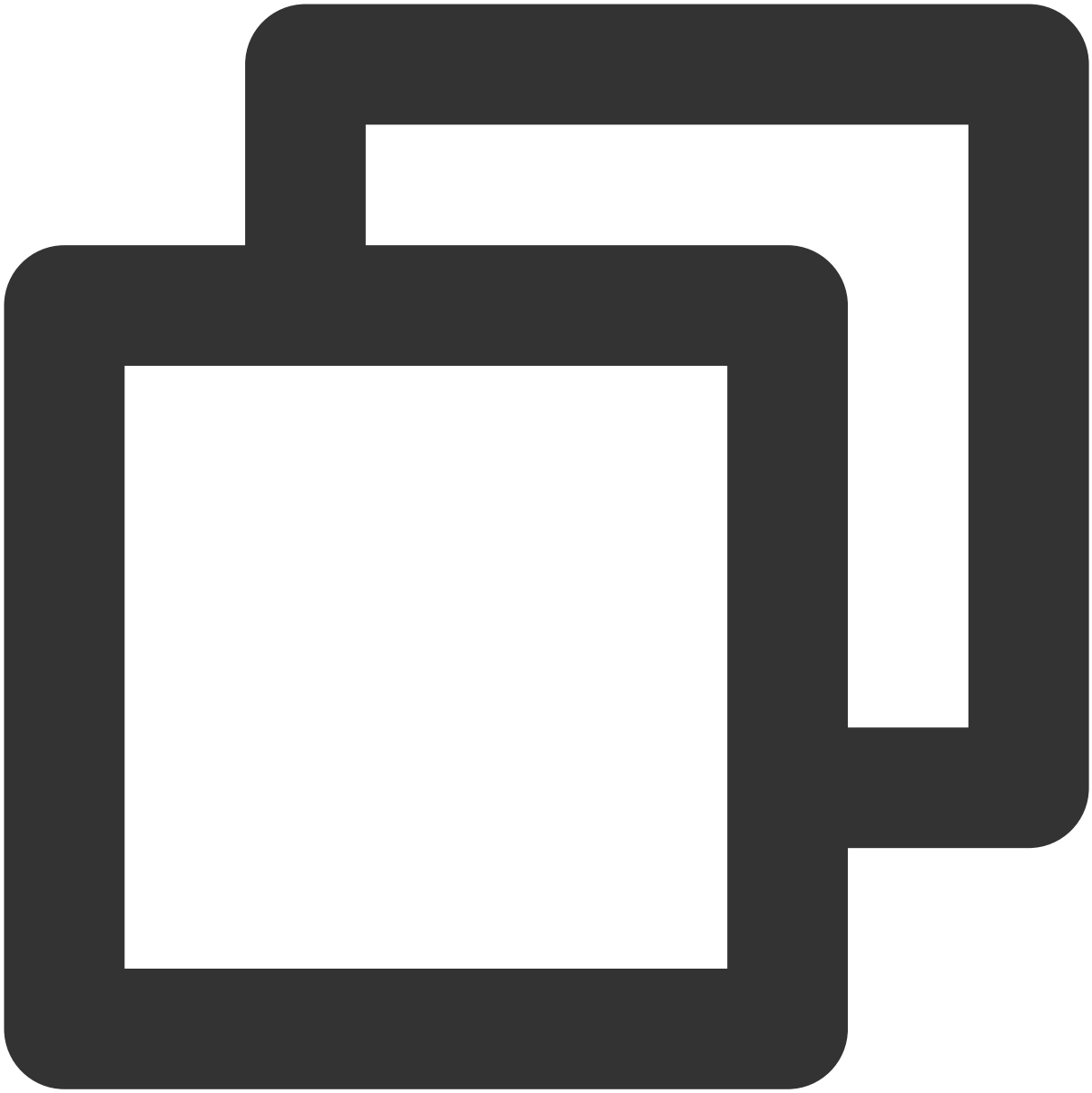
```
const os = require('os');
const targetPlatform = (function(){
     let target = os.platform();
     for (let i=0; i<process.argv.length; i++) {
```

```
            if (process.argv[i].includes('--target_platform=')) {
                target = process.argv[i].replace('--target_platform=', '');
                break;
            }
        }
        if (!['win32', 'darwin'].includes) target = os.platform();
        return target;
    })();
```

注意：

`os.platform()` 返回的结果中，"darwin" 表示 Mac 平台。"win32" 表示 Windows 平台，不论 64 位还是 32 位。

2. 然后在 `rules` 选项中添加以下配置， `targetPlatform` 变量可以使 `rewritePath` 可以根据不同的目标平台切换不同的配置：



```
rules: [
  {
          test: /\\.node$/,
          loader: 'native-ext-loader',
          options: {
              rewritePath: targetPlatform === 'win32' ? './resources' : '../Resource
              // 针对开发环境
              // rewritePath: './node_modules/trtc-electron-sdk/build/Release'
          }
      },
]
```

该配置的含义是：

打包 Windows 下的 `.exe` 文件时，让 `native-ext-loader` 到 `[应用程序根目录]/resources` 目录下加载 TRTC SDK。

打包 Mac 下的 `.dmg` 时，让 `native-ext-loader` 到 `[应用程序目录]/Contents/Frameworsk/../Resources` 目录下加载 TRTC SDK。

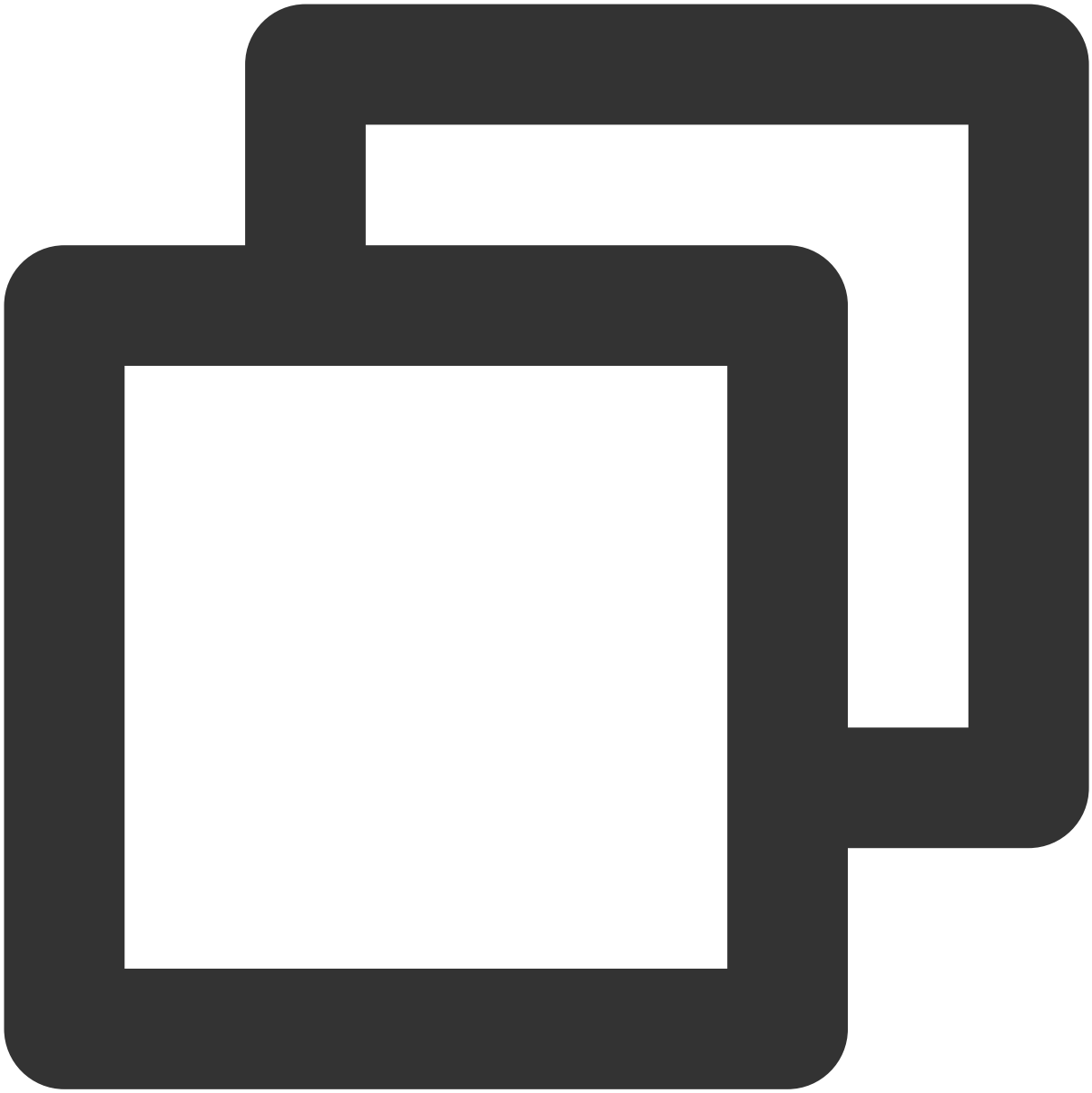
本地开发运行时，让 `native-ext-loader` 到 `./node_modules/trtc-electron-sdk/build/Release` 目录下加载 TRTC SDK，请参见 simple demo 配置。

还需要在 `package.json` 中的构建脚本中添加 `--target_platform` 参数，将在下一步进行。

步骤6：修改 package.json 配置

`package.json` 位于项目的根目录，其中包含了项目打包所必须的信息。但默认情况下， `package.json` 中的路径是需要修改才能顺利实现打包的，我们可以按如下步骤修改此文件：

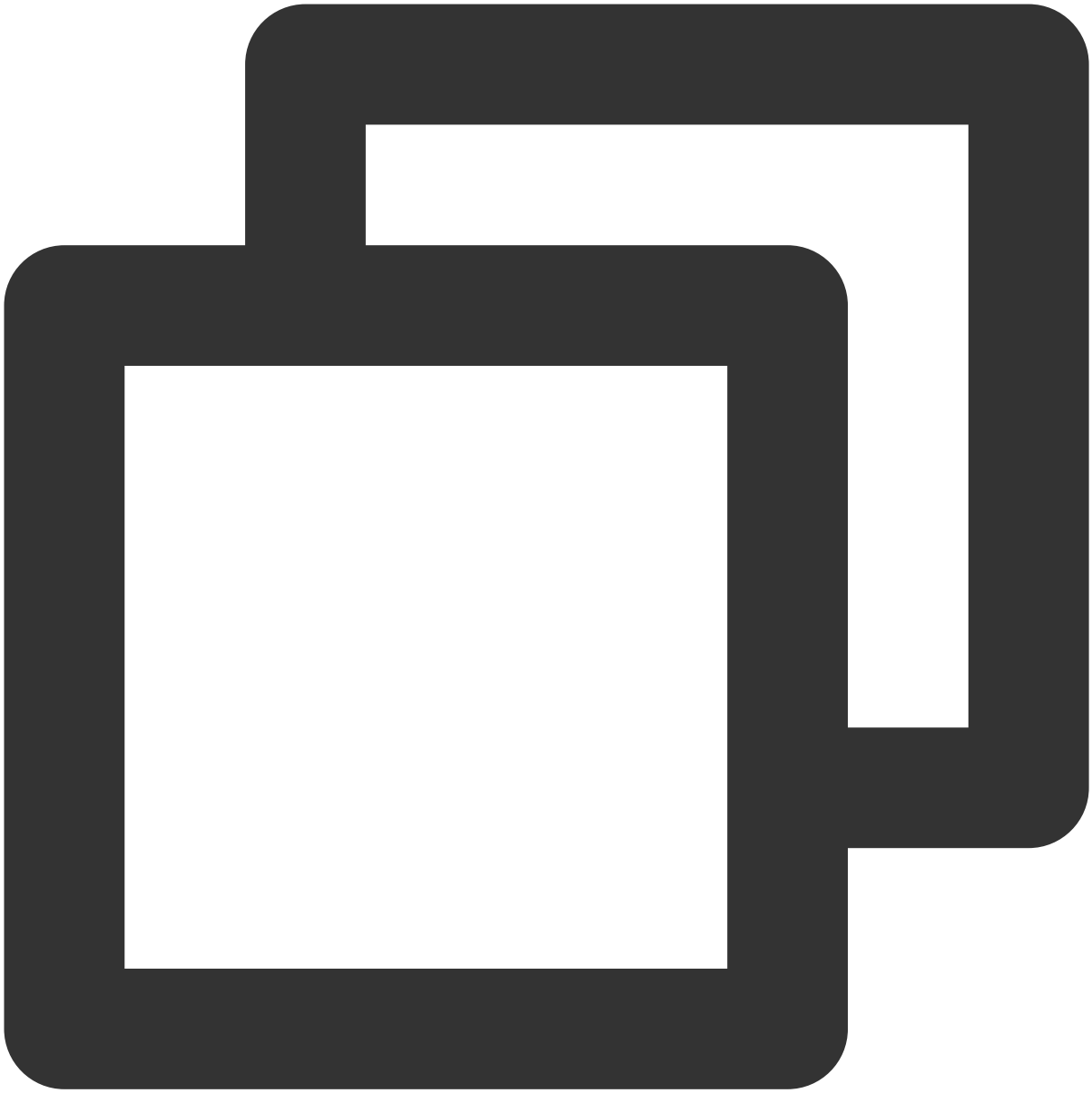
1. 修改 `main` 配置。



```
// 多数情况下，main 文件名称可以任意配置，例如 TRTCSimpleDemo 中的可以配置为：
"main": "main.electron.js",

// 但是，使用 create-react-app 脚手架创建的项目，main 文件必须配置为：
"main": "public/electron.js",
```

2. 复制以下 `build` 配置，添加到您的 `package.json` 文件中，这是 `electron-builder` 需要读取到的配置信息。

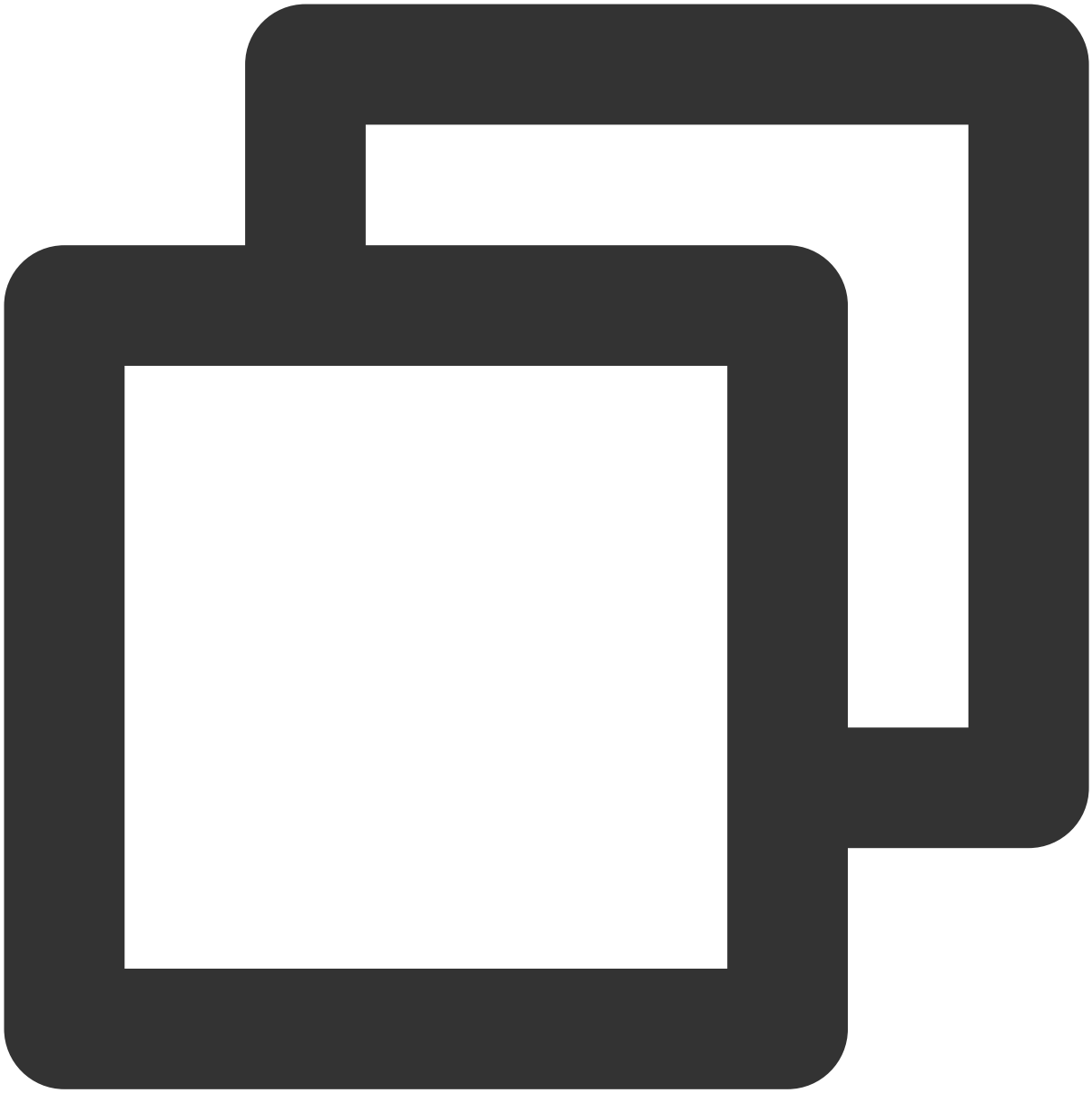


```
"build": {
  "appId": "[appId 请自行定义]",
  "directories": {
 "output": "./bin"
  },
  "win": {
 "extraFiles": [
   {
     "from": "node_modules/trtc-electron-sdk/build/Release/",
     "to": "./resources",
     "filter": ["**/*"]
```

```
    }
  ]
   },
   "mac": {
  "extraFiles": [
    {
      "from": "node_modules/trtc-electron-sdk/build/Release/trtc_electron_sdk.node",
      "to": "./Resources"
    }
  ]
   }
},
```

3. 在 `scripts` 节点下添加以下构建和打包的命令脚本：

本文以 `create-react-app` 和 `vue-cli` 项目为例，其它工具创建的项目也可以参考此配置：



```
// create-react-app 项目请使用此配置
"scripts": {
  "build:mac": "react-scripts build --target_platform=darwin",
  "build:win": "react-scripts build --target_platform=win32",
  "compile:mac": "node_modules/.bin/electron-builder --mac",
  "compile:win64": "node_modules/.bin/electron-builder --win --x64",
  "pack:mac": "npm run build:mac && npm run compile:mac",
  "pack:win64": "npm run build:win && npm run compile:win64"
}

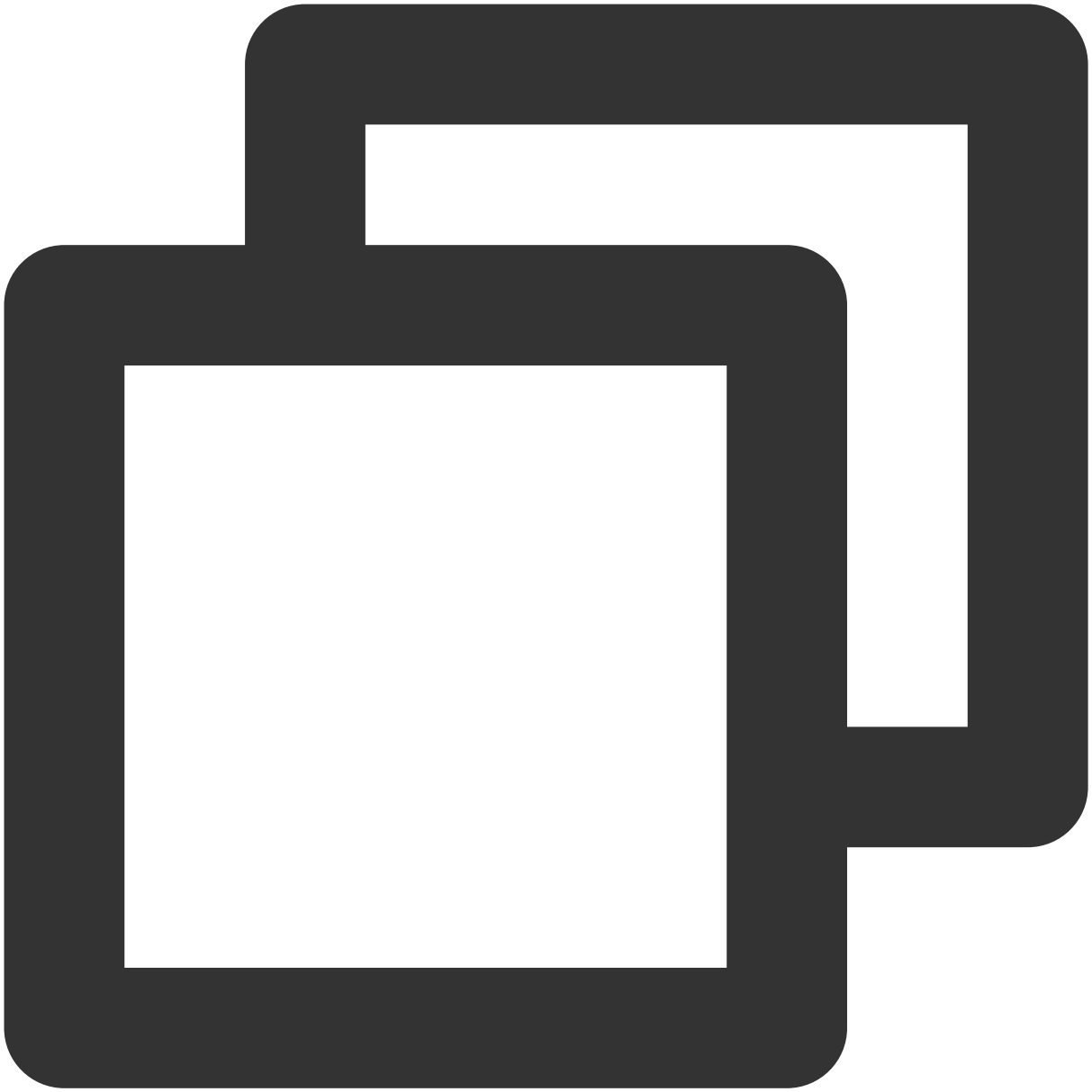
// vue-cli 项目请使用此配置
```

```
"scripts": {
  "build:mac": "vue-cli-service build --target_platform=darwin",
  "build:win": "vue-cli-service build --target_platform=win32",
  "compile:mac": "node_modules/.bin/electron-builder --mac",
  "compile:win64": "node_modules/.bin/electron-builder --win --x64",
  "pack:mac": "npm run build:mac && npm run compile:mac",
  "pack:win64": "npm run build:win && npm run compile:win64"
}
```

参数	说明
main	Electron 的入口文件，一般情况下可以自由配置。但如果项目使用 create-react-app 脚手架创建，则入口文件必须配置为 public/electron.js
build.win.extraFiles	打包 Windows 程序时，electron-builder 会把 from 所指目录下的所有文件复制到 bin/win-unpacked/resources（全小写）
build.mac.extraFiles	打包 Mac 程序时，electron-builder 会把 from 指向的 trtc_electron_sdk.node 文件复制到 bin/mac/your-app-name.app/Contents/Resources（首字母大写）
build.directories.output	打包文件的输出路径。例如这个配置会输出到 bin 目录下，可根据实际需要修改
build.scripts.build:mac	以 Mac 平台为目标构建脚本
build.scripts.build:win	以 Windows 平台为目标构建脚本
build.scripts.compile:mac	编译为 Mac 下的 .dmg 安装文件
build.scripts.compile:win64	编译为 Windows 下的 .exe 安装文件
build.scripts.pack:mac	先调用 build:mac 构建代码，再调用 compile:mac 打包成 .dmg 安装文件
build.scripts.pack:win64	先调用 build:win 构建代码，再调用 compile:win64 打包成 .exe 安装文件

步骤7：执行打包命令

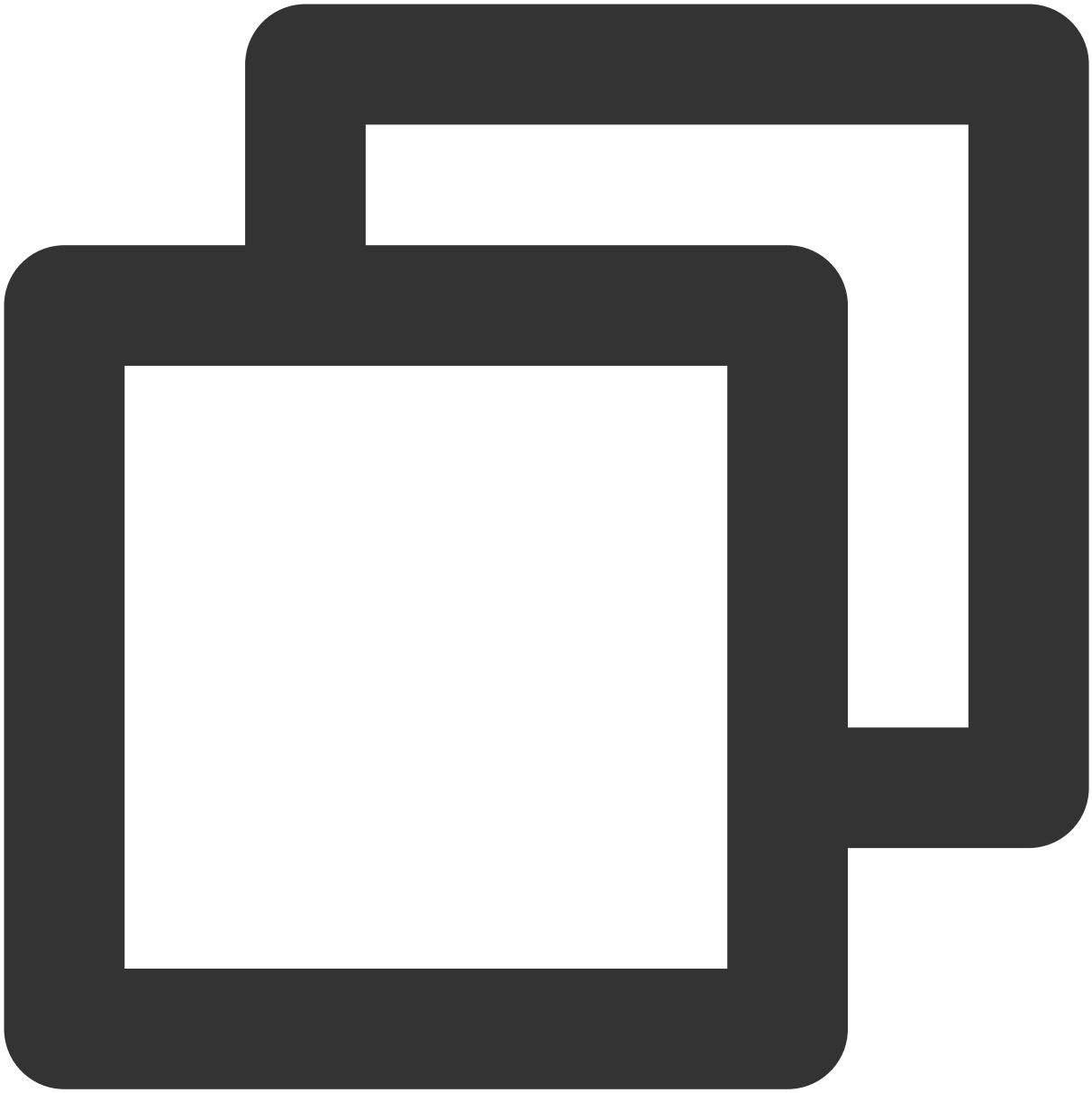
打包 Mac.dmg 安装文件：



```
$ cd [项目目录]
$ npm run pack:mac
```

成功执行后，打包工具会生成 `bin/your-app-name-0.1.0.dmg` 安装文件，请选择此文件发布。

打包 Windows.exe 安装文件：



```
$ cd [项目目录]
$ npm run pack:win64
```

成功执行后，打包工具会生成 `bin/your-app-name Setup 0.1.0.exe` 安装文件，请选择此文件发布。

注意：

TRTC Electron SDK 暂不支持跨平台打包（例如在 Mac 下打包 Windows 的 .exe 文件，或在 Windows 平台下打包 Mac 的 .dmg 文件）。目前我们正在研究跨平台打包方案，敬请期待。

常见问题

防火墙有什么限制？

由于 SDK 使用 UDP 协议进行音视频传输，所以对 UDP 有拦截的办公网络下无法使用，如遇到类似问题，请参见应对公司防火墙限制。

Electron 安装或打包异常？

如果您在集成 Electron 过程中遇到异常：例如安装超时或失败，打包后出现 trtc_electron_sdk.node 文件加载失败等情况，相关问题解答请联系我们。

参考文档

SDK API 手册
SDK 更新日志
Simple Demo 源码
API Example 源码
Electron 常见问题

Flutter

最近更新时间：2024-06-07 11:20:49

本文主要介绍如何快速地将腾讯云 TRTC SDK（Flutter）集成到您的项目中，只要按照如下步骤进行配置，就可以完成 SDK 的集成工作。

注意：

目前 Windows/MacOs 端暂不支持屏幕分享及设备选择功能。

环境要求

Flutter 2.0 及以上版本。

Android 端开发：

Android Studio 3.5及以上版本。

App 要求 Android 4.1及以上版本设备。

iOS & macOS 端开发：

Xcode 11.0及以上版本。

osx 系统版本要求 10.11 及以上版本

请确保您的项目已设置有效的开发者签名。

Windows 开发：

操作系统：Windows 7 SP1 或更高的版本（基于 x86-64 的 64 位操作系统）。

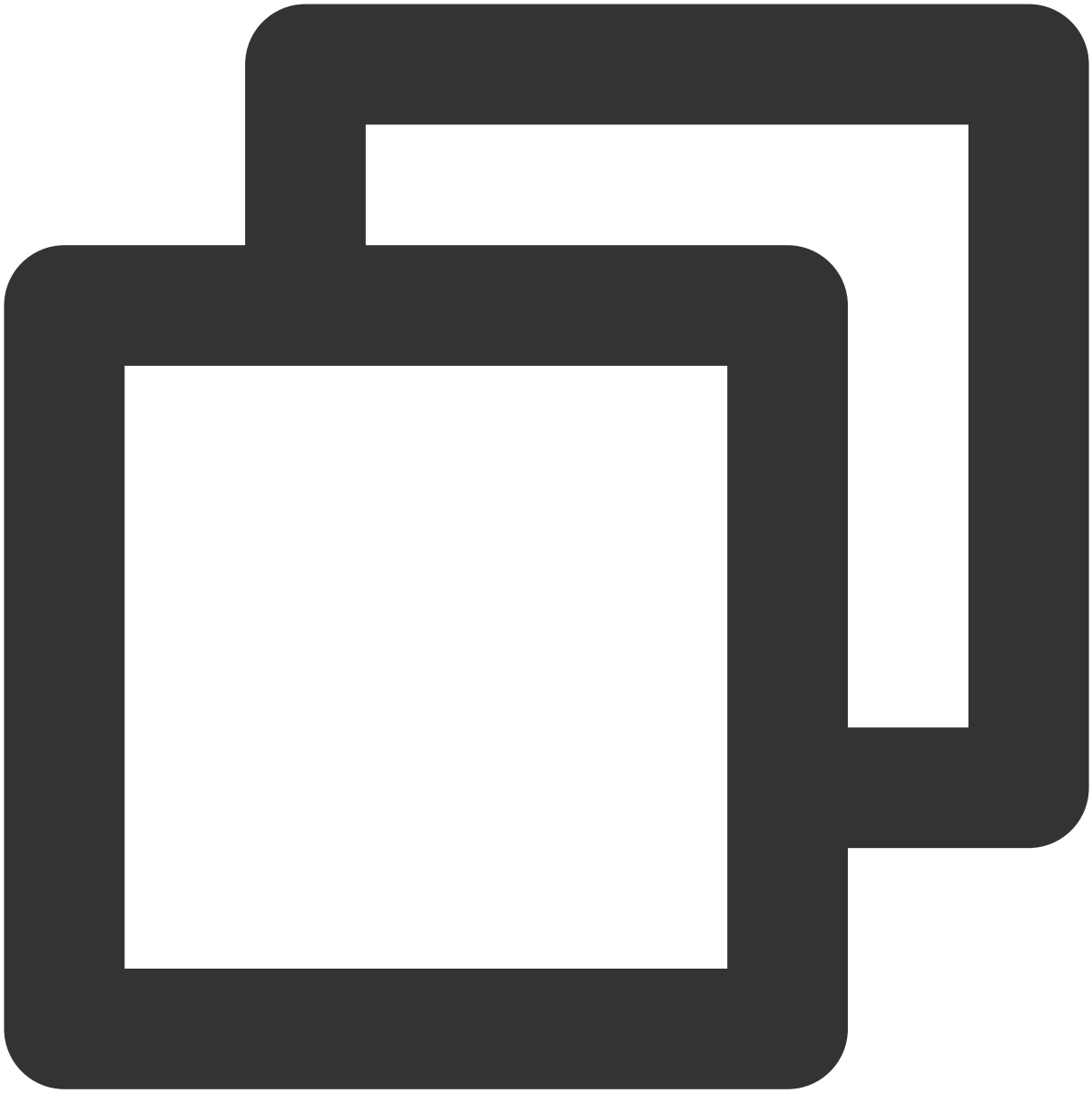
磁盘空间：除安装 IDE 和一些工具之外还应有至少 1.64 GB 的空间。

安装 Visual Studio 2019。

集成 SDK

Flutter SDK 已经发布到 pub 库，您可以通过配置 `pubspec.yaml` 自动下载更新。

1. 在项目的 `pubspec.yaml` 中写如下依赖：



```
dependencies:
  tencent_trtc_cloud: 最新版本号
```

2. 开通**摄像头**和**麦克风**的权限，即可开启语音通话功能。

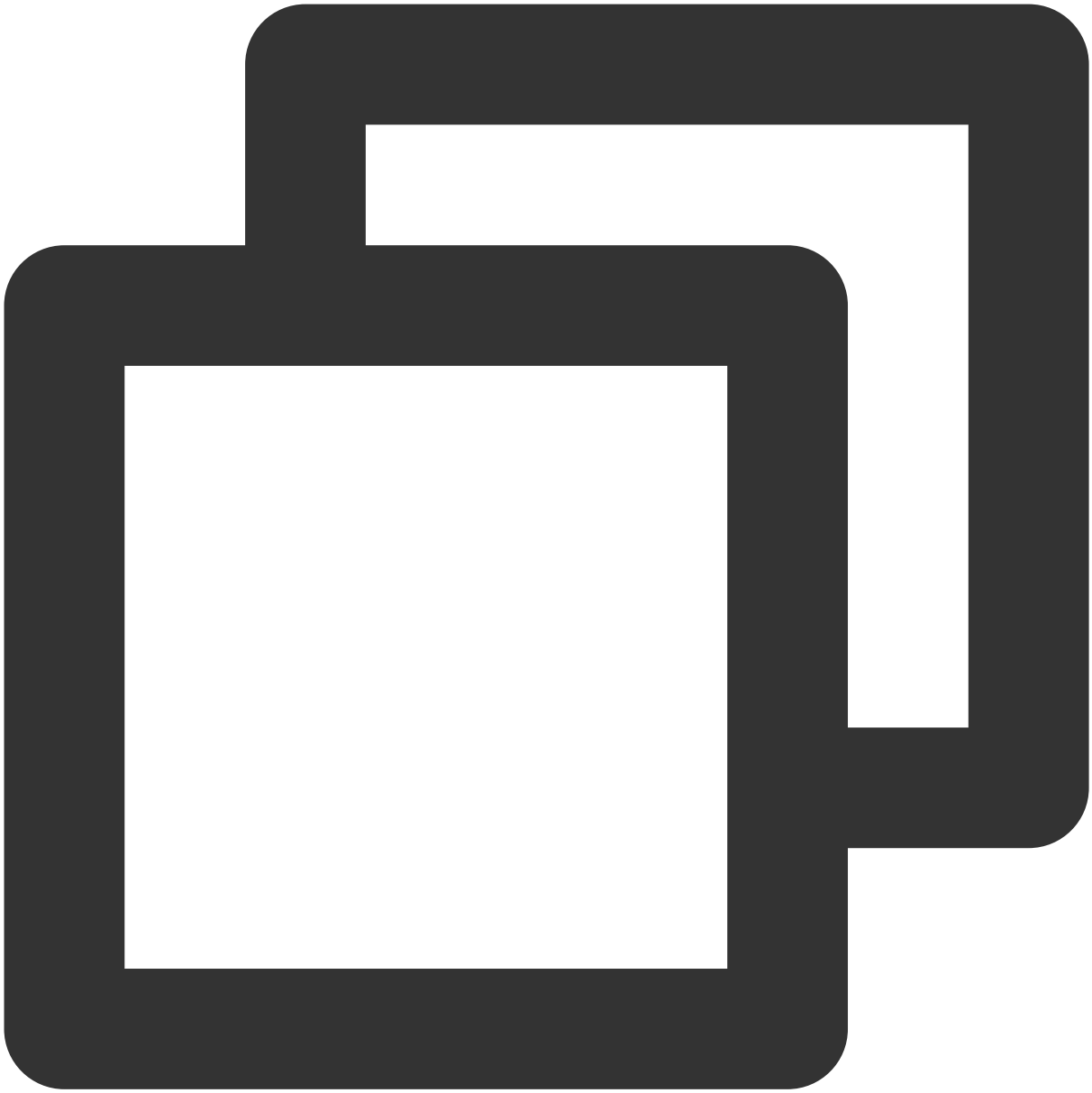
iOS 端

macOS 端

Android 端

Windows 端

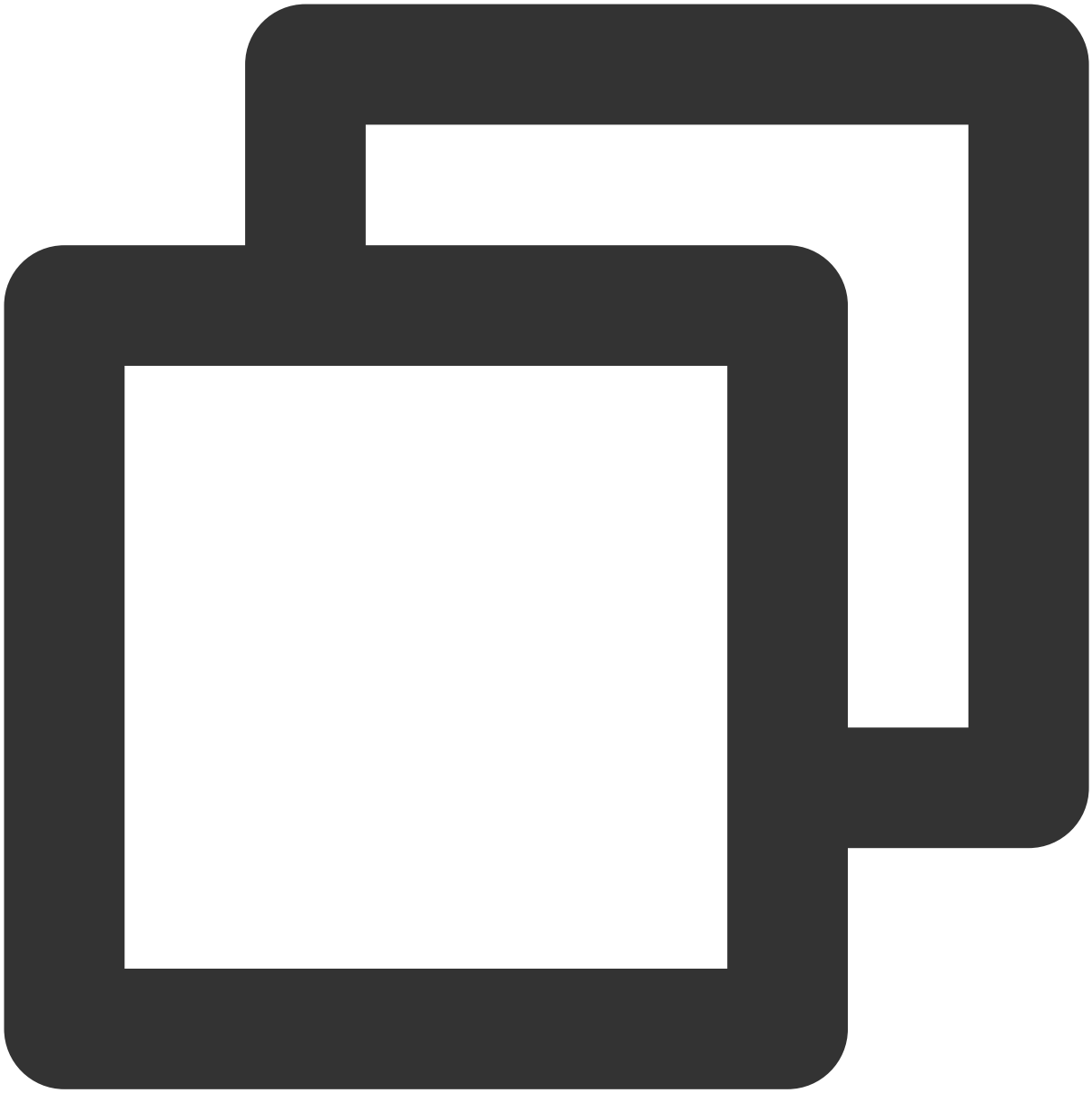
1. 需要在 `Info.plist` 中加入对相机和麦克风的权限申请：



```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
```

2. 添加字段 `io.flutter.embedded_views_preview`，并设定值为 YES。

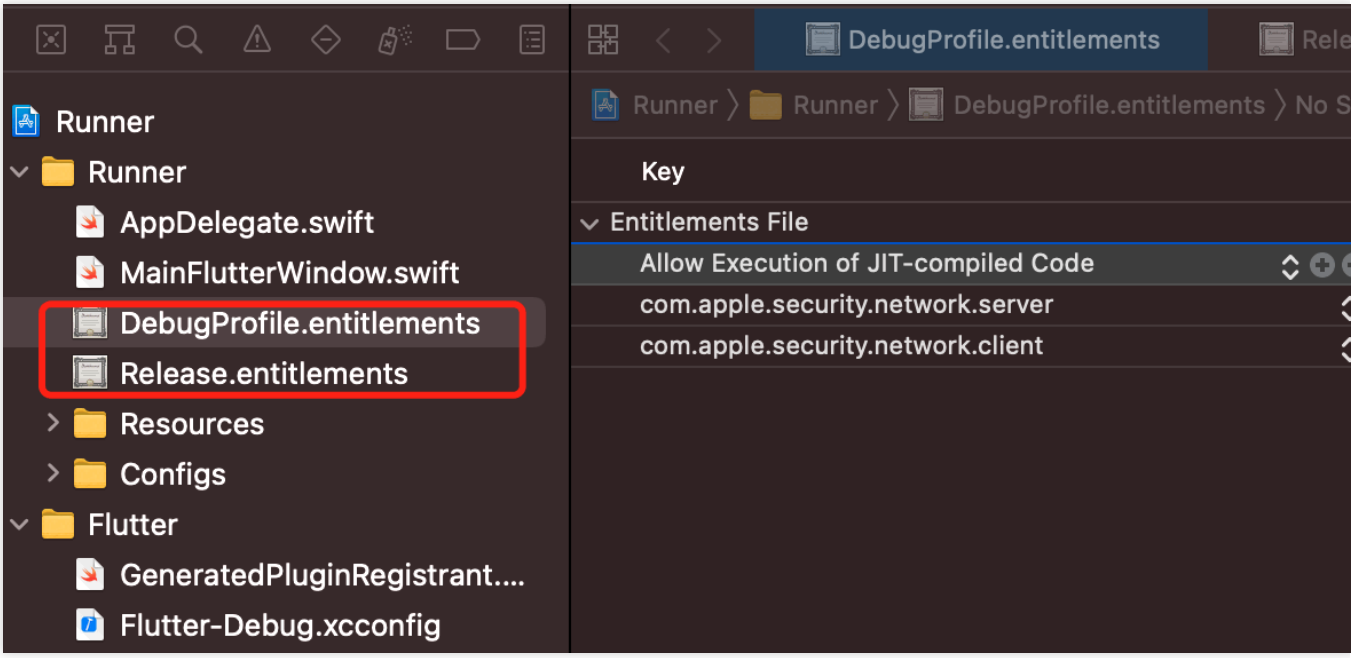
1. 需要在 `Info.plist` 中加入对相机和麦克风的权限申请：



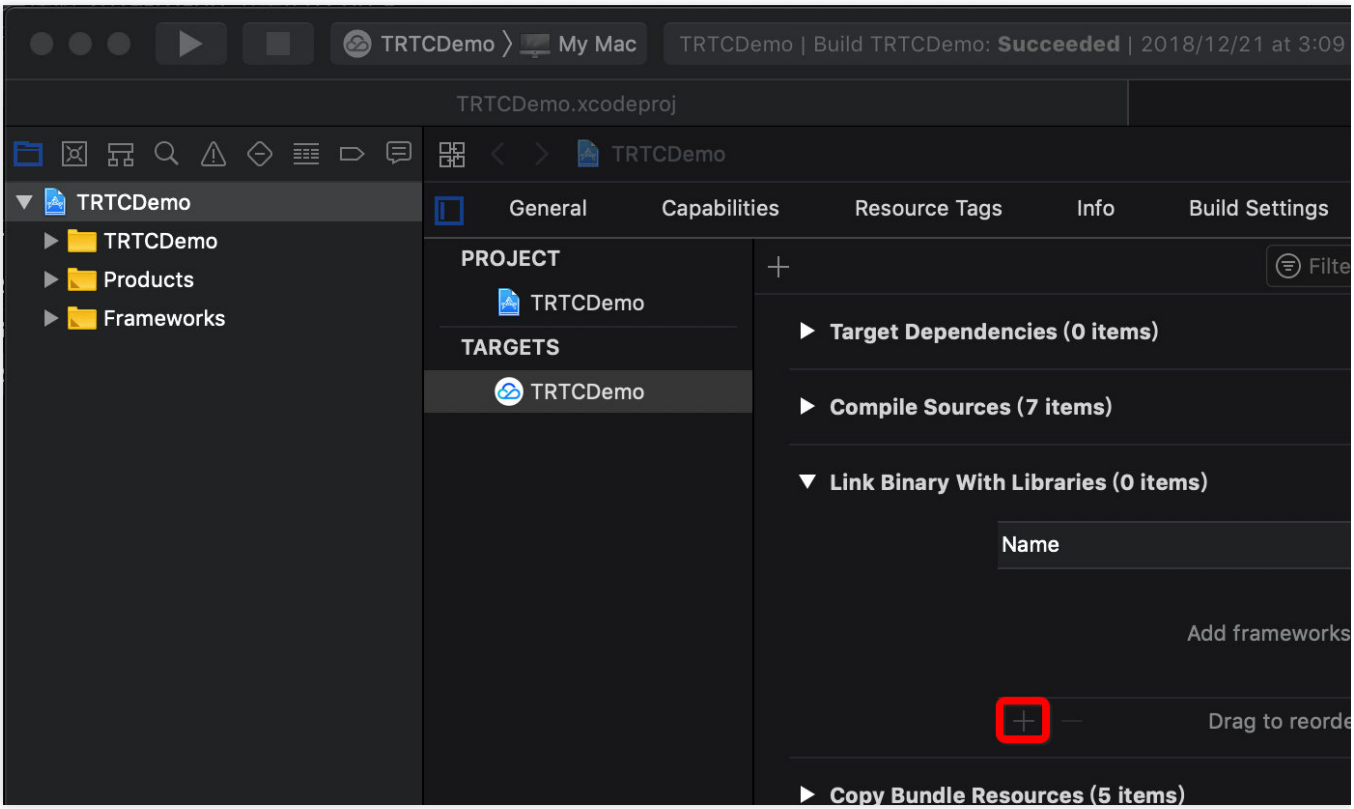
```
<key>NSCameraUsageDescription</key>
<string>授权摄像头权限才能正常视频通话</string>
<key>NSMicrophoneUsageDescription</key>
<string>授权麦克风权限才能正常语音通话</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>App需要您的同意,才能访问相册</string>
```

2. 需要在 `macos/Runner/*.entitlements` 文件中添加 `com.apple.security.network.client` 、`com.apple.security.network.server`

添加后如下图所示：

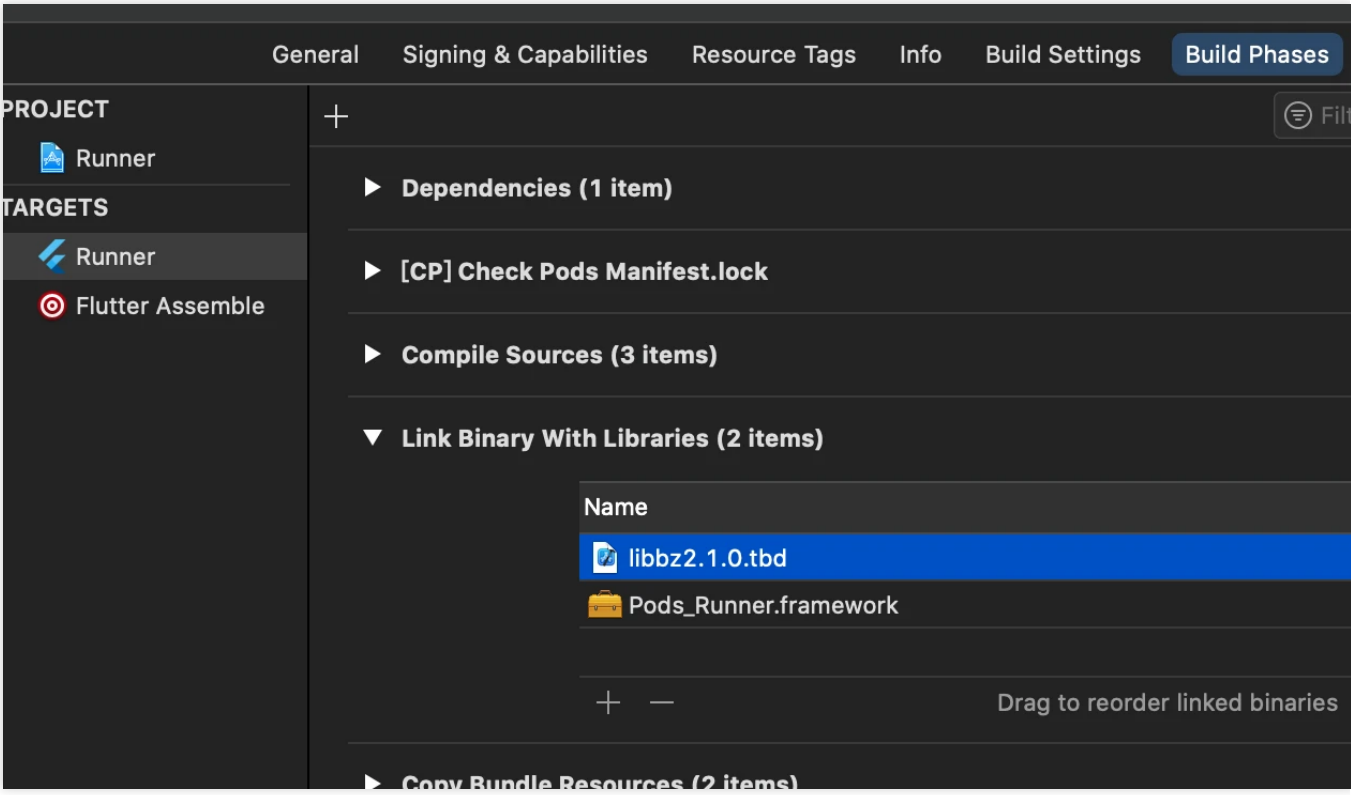


3. 单击 Link Binary with Libraries 项展开，单击底下的+号图标去添加依赖库。



4. 添加其所需依赖库： `libbz2.1.0.tbd` 。

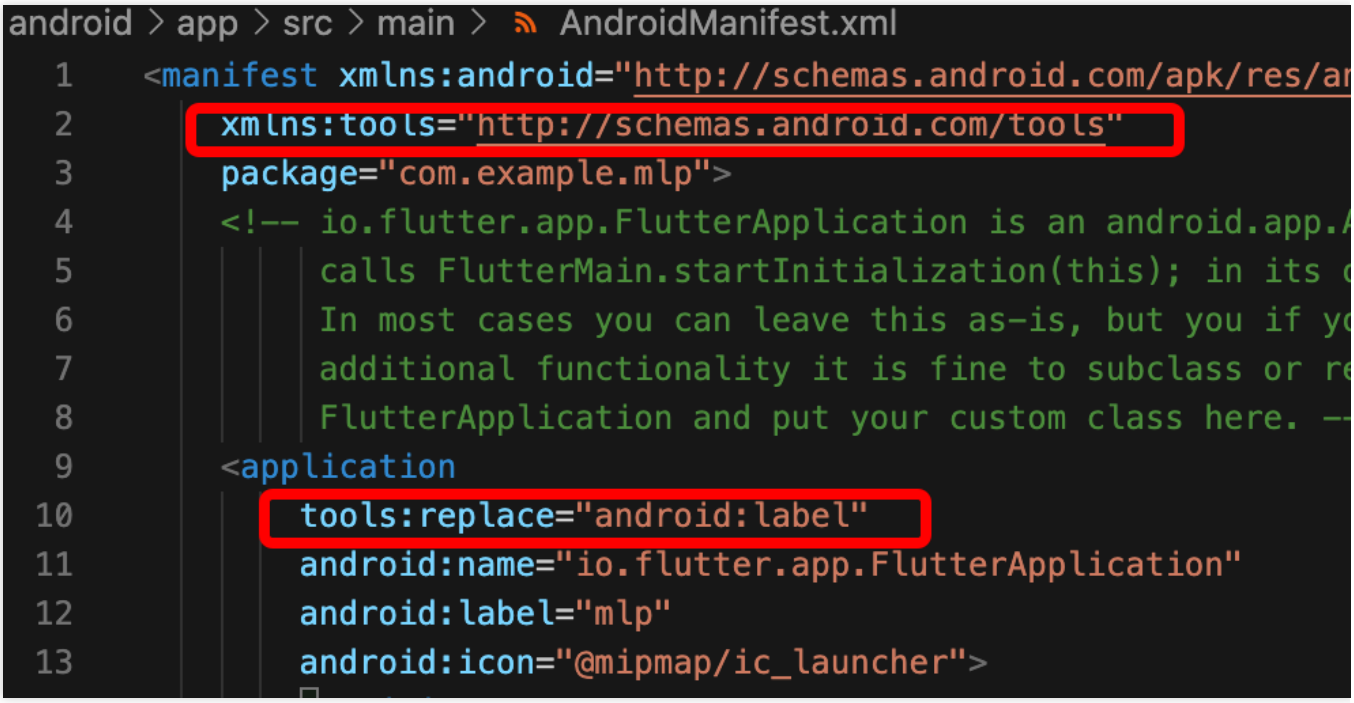
添加后如下图所示：



1. 打开 `/android/app/src/main/AndroidManifest.xml` 文件。
2. 将 `xmlns:tools="http://schemas.android.com/tools"` 加入到 manifest 中。
3. 将 `tools:replace="android:label"` 加入到 application 中。

说明：

若不执行此步，会出现 Android Manifest merge failed 编译失败 问题。



1. 启用 Windows 支持：`flutter config --enable-windows-desktop`。
2. `flutter run -d windows`。

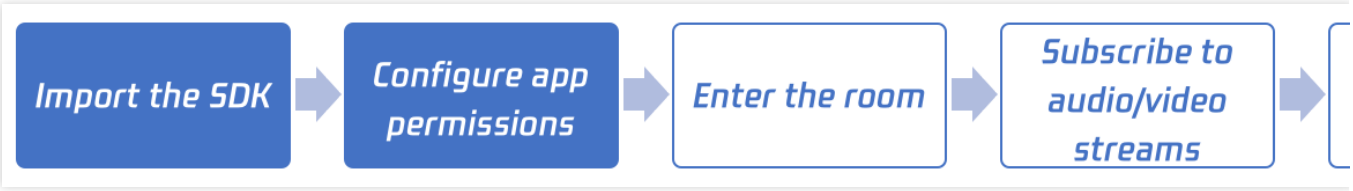
常见问题

iOS 打包运行 Crash？
iOS 无法显示视频（Android 正常）？
更新 SDK 版本后，iOS CocoaPods 运行报错？
Android Manifest merge failed 编译失败？
因为没有签名，真机调试报错?
对插件内的 swift 文件做了增删后，build 时查找不到对应文件？
Run 报错“Info.plit, error: No value at that key path or invalid key path: NSBonjourServices”？
Pod install 报错？
Run 的时候 iOS 版本依赖报错？

QT

最近更新时间：2024-06-07 11:20:49

本文主要介绍如何快速地将腾讯云 TRTC SDK（QT 的 Windows 和 Mac 版本）集成到您的项目中，只要按照如下步骤进行配置，就可以快速完成 SDK 的集成工作。



Windows 端集成

开发环境要求

操作系统：Windows 7及以上版本。
开发环境：Visual Studio 2015及以上版本，推荐使用 Visual Studio 2015，前提是您已经配置好 VS 相关的 QT 开发环境。
说明：
如果您不熟悉配置 VS 相关 QT 开发环境的步骤，请参见 README 中的操作步骤的步骤4相关内容。

操作步骤

本节通过创建一个简单的 QT 项目，介绍如何在 Visual Studio 工程中集成 C++ SDK。

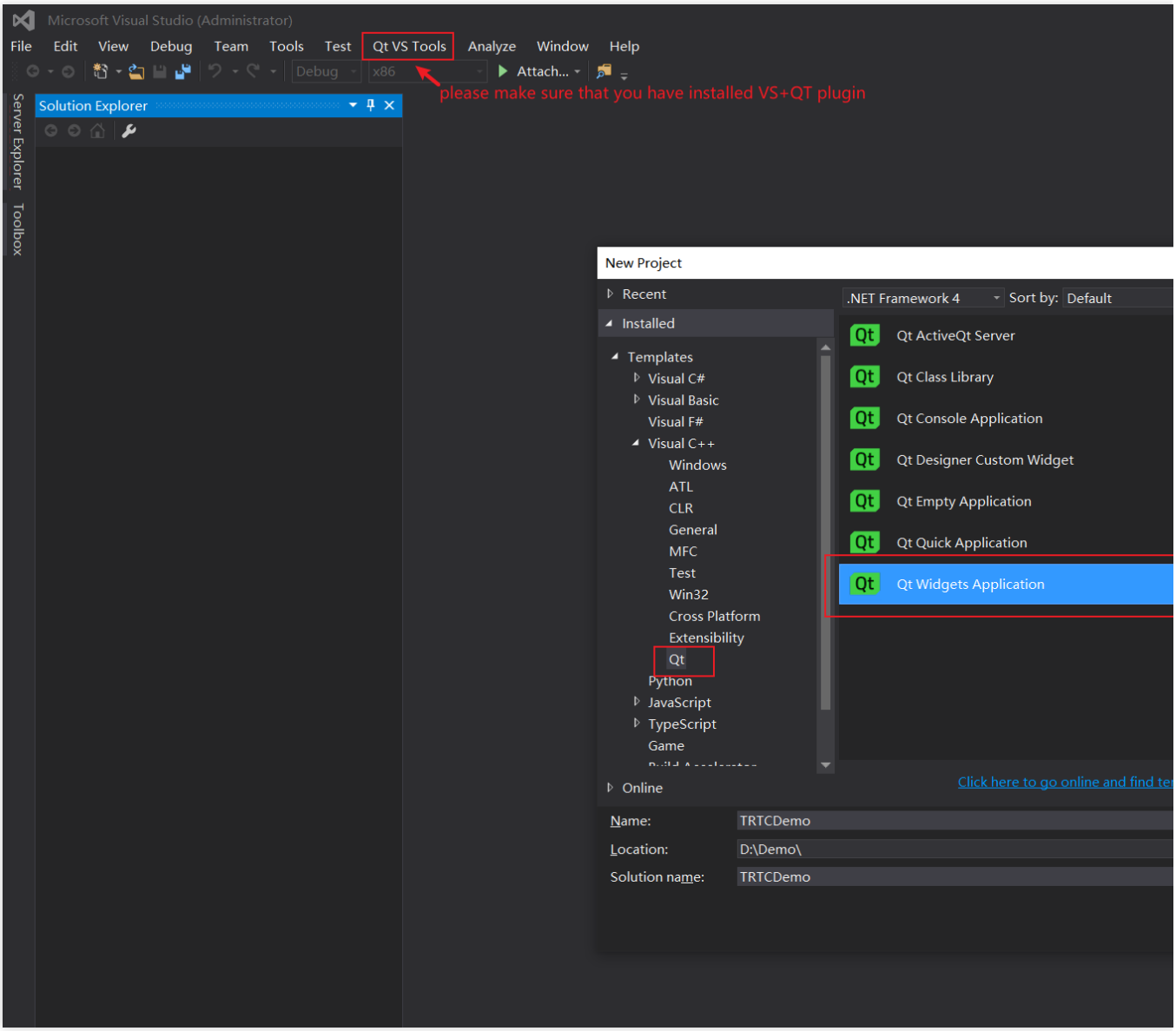
步骤1：下载 SDK

下载 SDK，解压并打开。
本文示例中，您只需要引用 SDK 目录下 C++ 版的 SDK 文件即可。以64位为例，其 SDK 位置为 `./SDK/CPlusPlus/Win64/` 下，主要包含以下几个部分：

目录名	说明
include	带有详细接口注释的 API 头文件
lib	编译用的 .lib 文件和运行时加载的 .dll 文件

步骤2：新建工程

以 Visual Studio 2015 为例，在确保本地已经安装了 QT 和 VS 开发插件 的前提下，打开 Visual Studio。新建一个名字叫 `TRTCDemo` 的 QT 应用程序，如下图所示：



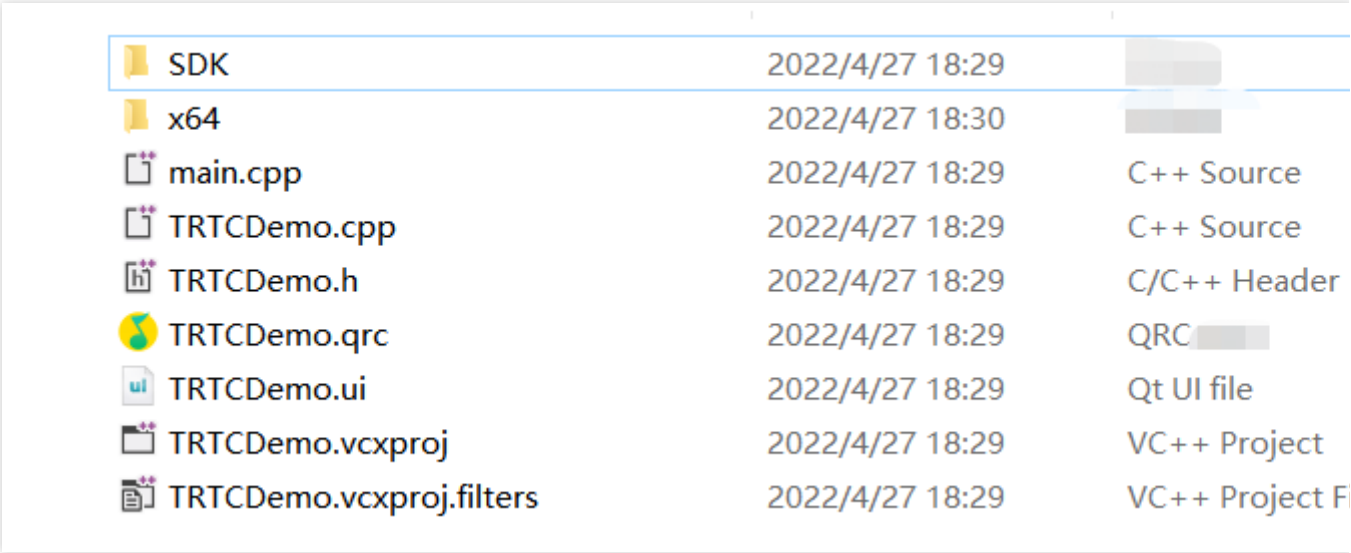
为了便于介绍如何快速集成，在向导中我们选择 **Qt Widgets Application** 类型，单击**确定**，在之后的页面中单击 **Next** 直到工程创建完成即可。

步骤3：拷贝文件

将解压后的 SDK 文件夹拷贝到 TRTCDemo.vcxproj 所在目录下，如下图所示：

说明：

当前只需要 C++ SDK，可以将 SDK 路径下的 CSharp 目录删除。



步骤4：修改工程配置

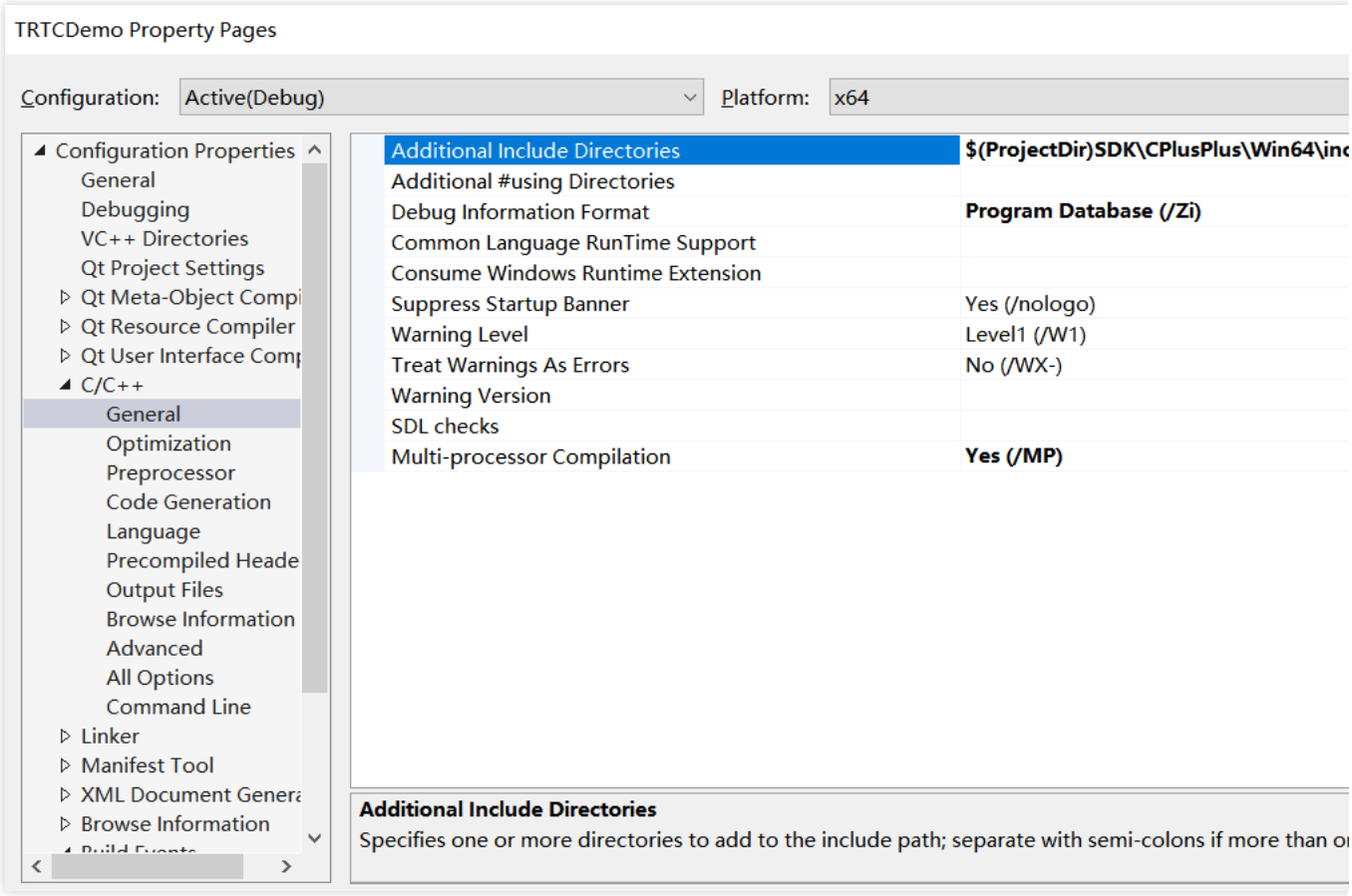
打开 TRTCDemo 属性页，在**解决方案资源管理器 >TRTCDemo 工程的右键菜单 > 属性**，请按照以下步骤进行配置：

1. **添加包含目录：**

在 **C/C++ > 常规 > 附加包含目录**，以64位为例，添加 SDK 头文件目录 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include` 和 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\include\\TRTC`，如下图所示：

说明：

如果为32位，则需要将 SDK 头文件目录设为 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include` 和 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\include\\TRTC`。

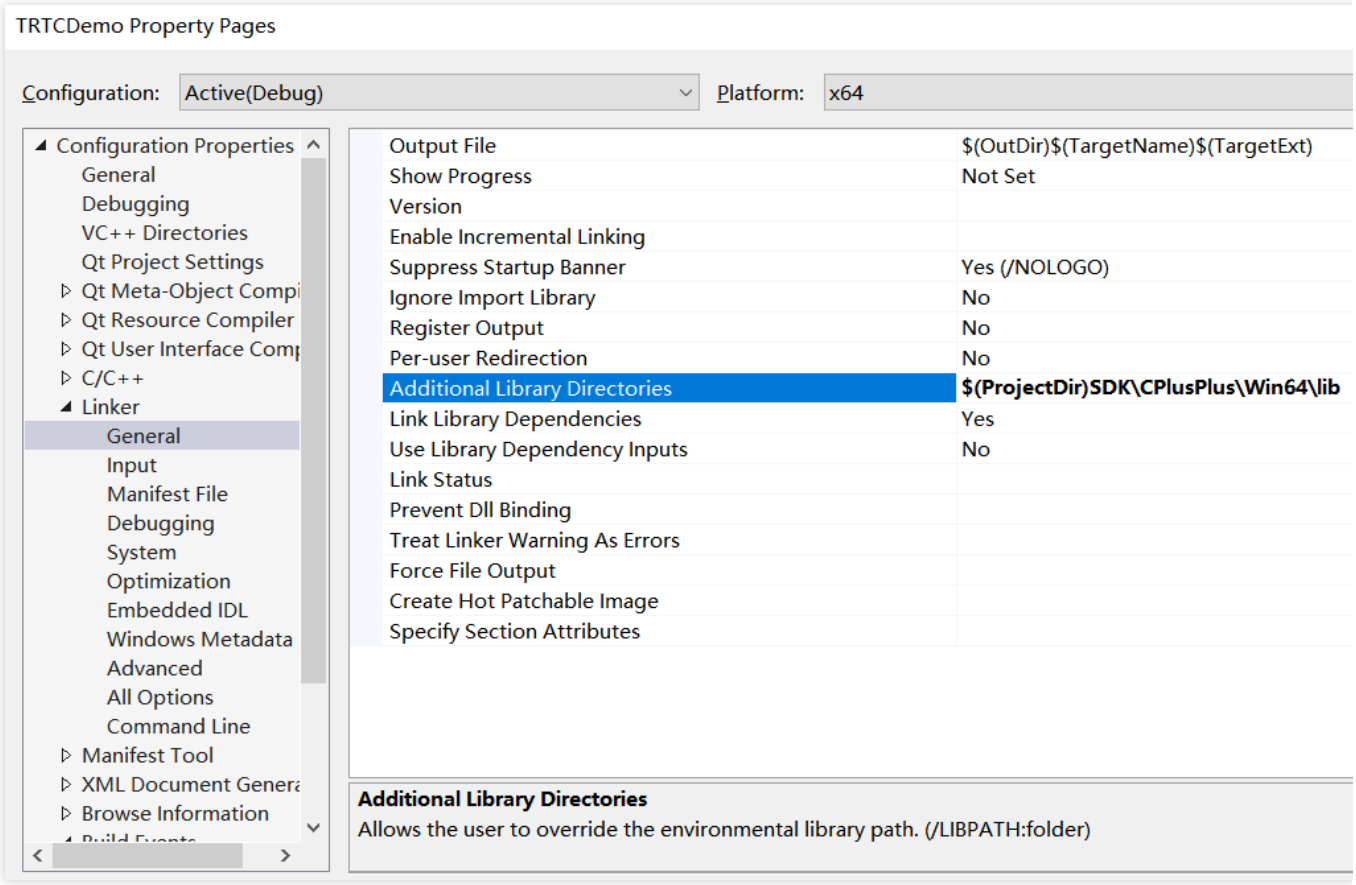


2. **添加库目录：**

在**链接器** > **常规** > **附加库目录**，以64位为例，添加 SDK 库目录 `$(ProjectDir)SDK\\CPlusPlus\\Win64\\lib`，如下图所示：

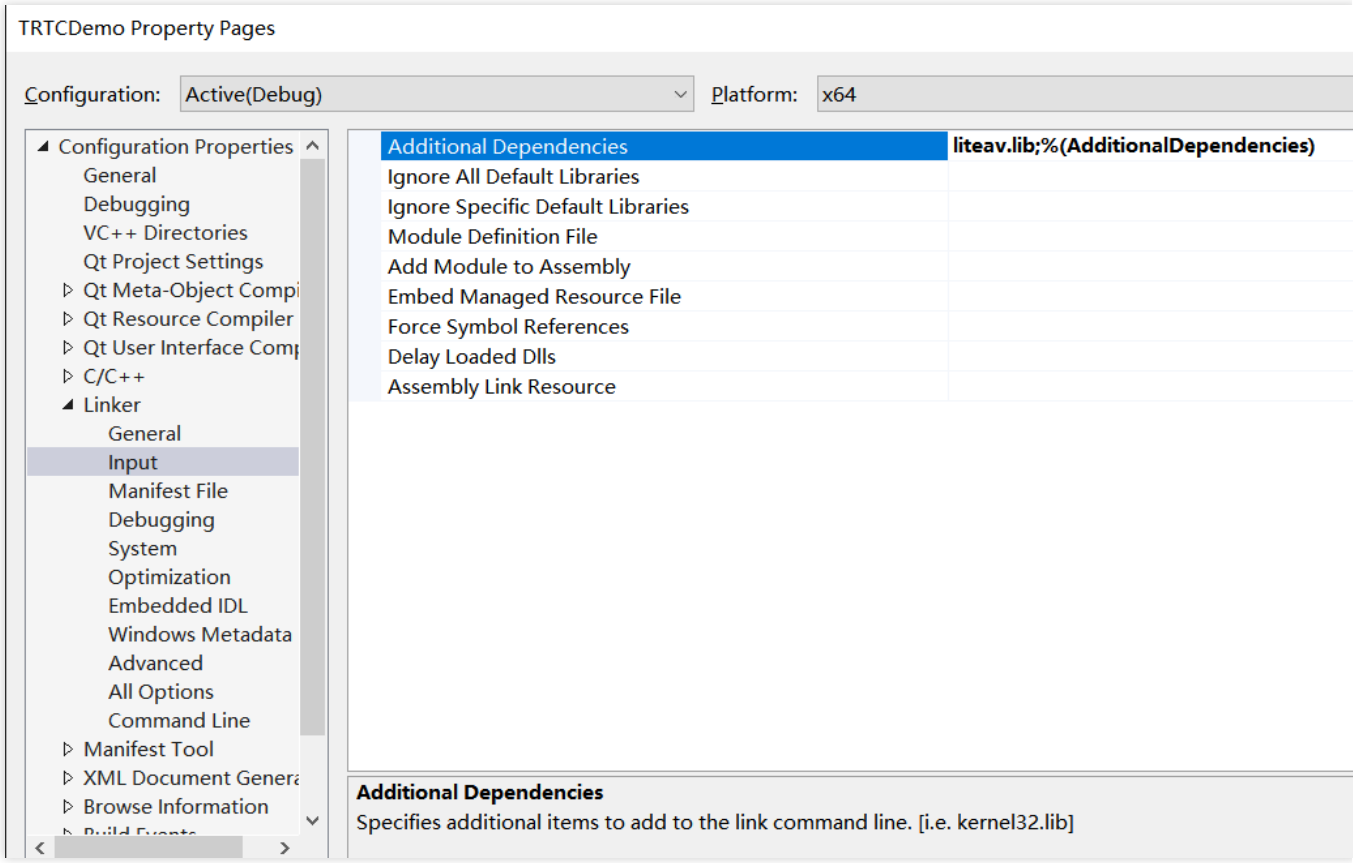
说明：

如果为32位，则需要将 SDK 库目录设为 `$(ProjectDir)SDK\\CPlusPlus\\Win32\\lib` 。



3. **添加库文件：**

在**链接器 > 输入 > 附加依赖项**，添加 SDK 库文件 `liteav.lib`，如下图所示：

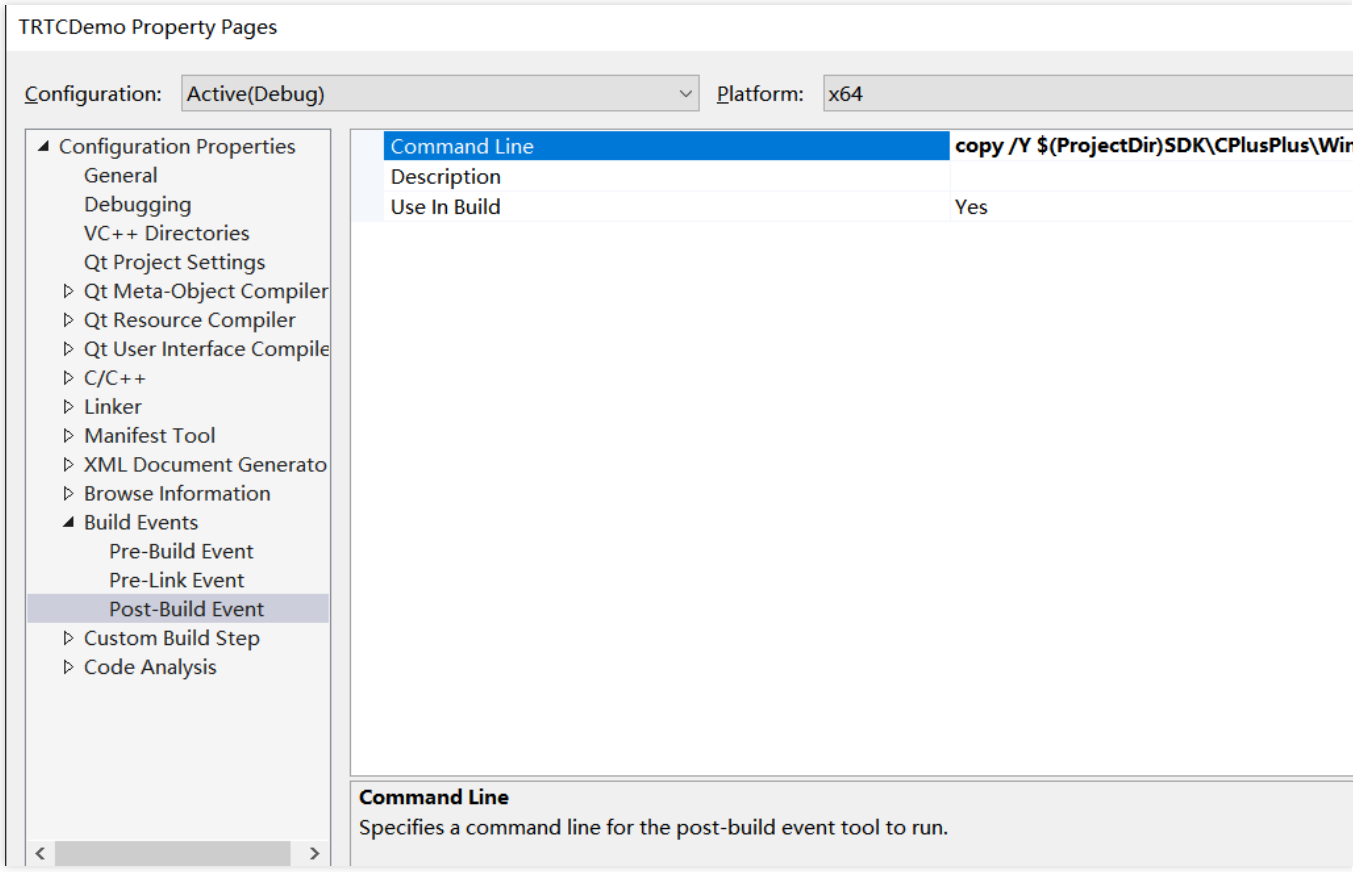


4. **添加 copy 命令：**

在**生成事件 > 后期生成事件 > 命令行**，添加拷贝命令 `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win64\\lib\\*.dll $(OutDir)`，能够在编译完成后，自动将 SDK 的 .dll 文件拷贝到程序的运行目录下，如下图所示：

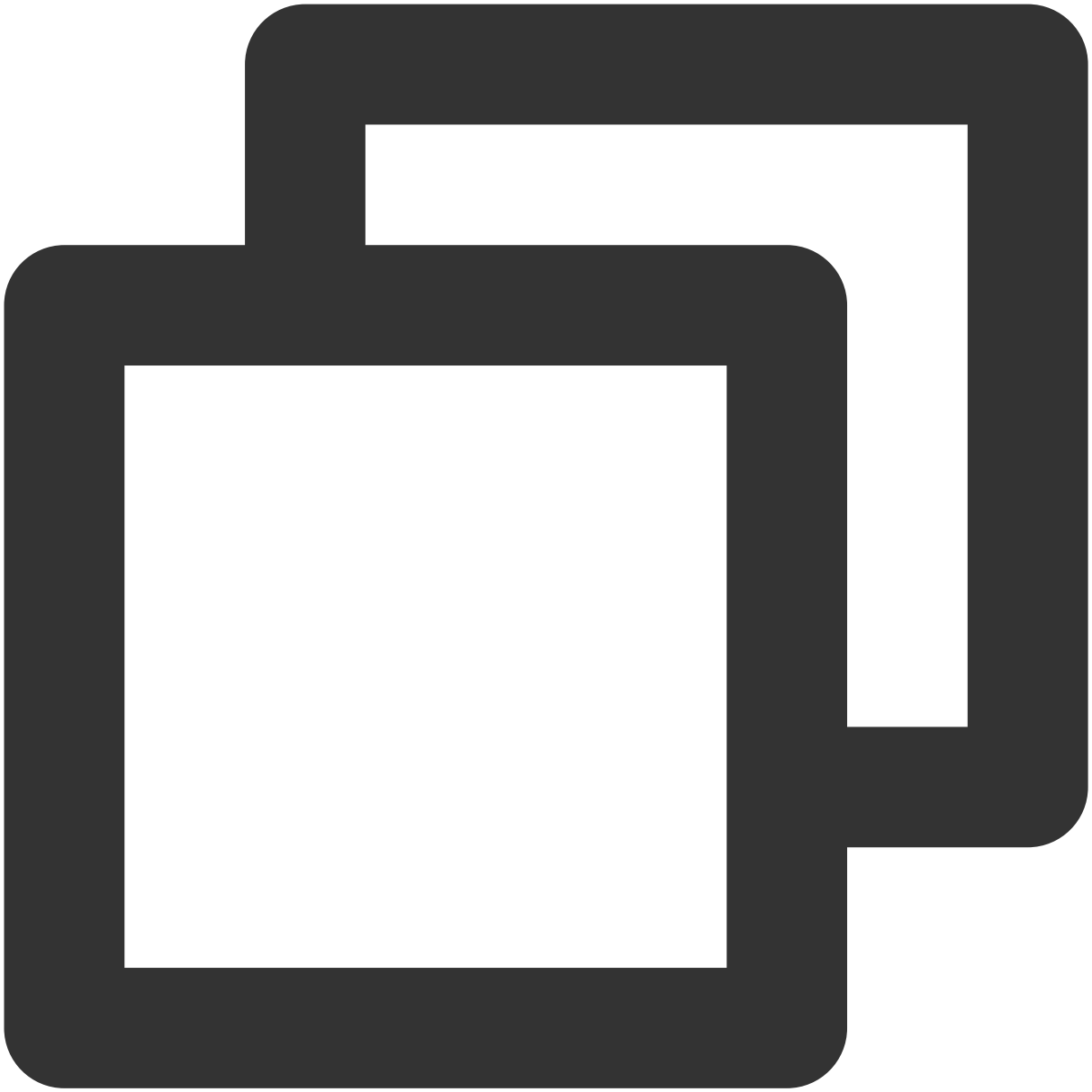
说明：

如果为32位，则添加拷贝命令为 `copy /Y $(ProjectDir)SDK\\CPlusPlus\\Win32\\lib\\*.dll $(OutDir)` 。



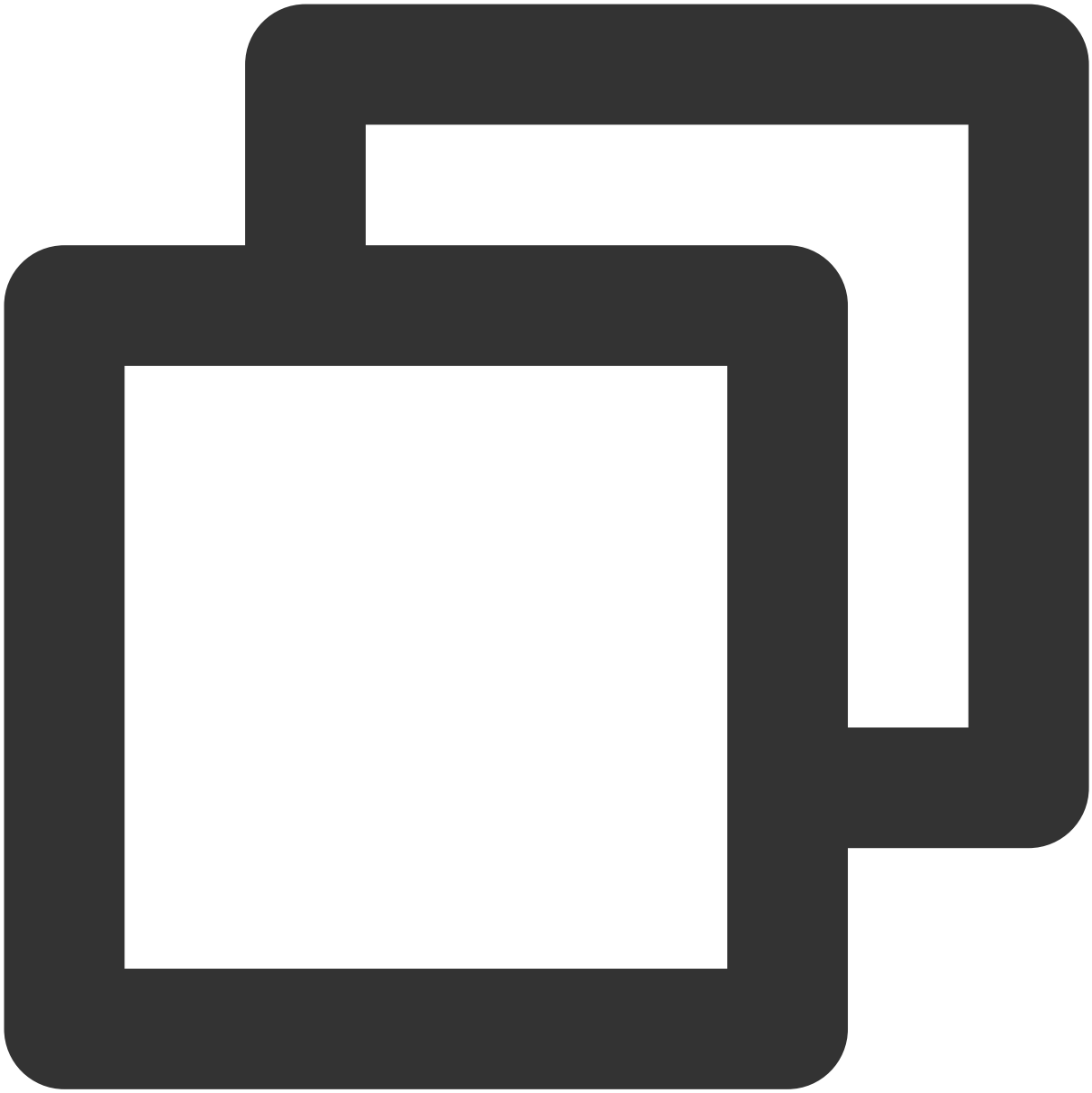
步骤5：打印 SDK 版本号

1. 在 `TRTCDemo.cpp` 文件顶部增加头文件引入，代码如下：



```
#include "ITRTCCloud.h"
#include <QLabel>
```

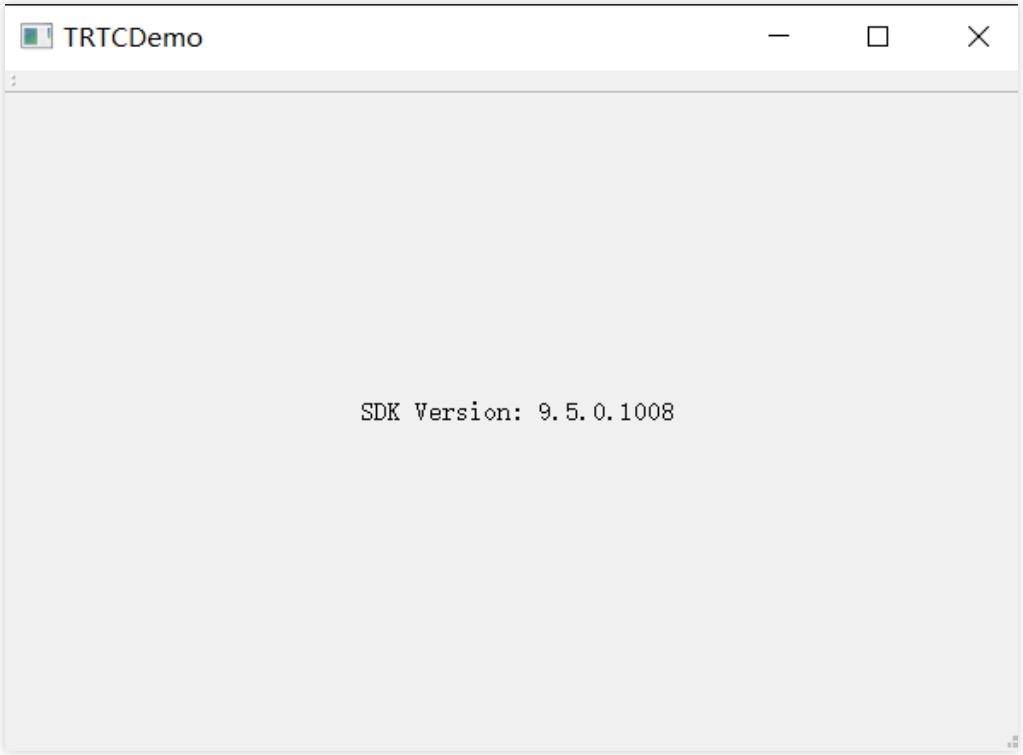
2. 在 `TRTCDemo.cpp` 文件的 `TRTCDemo::TRTCDemo` 构造函数中，添加下面的测试代码：



```
ITRTCCloud * pTRTCCloud = getTRTCShareInstance();
std::string version(pTRTCCloud->getSDKVersion());

QString sdk_version = QString("SDK Version: %1").arg(version.c_str());
QLabel* label_text = new QLabel(this);
label_text->setAlignment(Qt::AlignCenter);
label_text->resize(this->width(), this->height());
label_text->setText(sdk_version);
```

3. 按键盘 F5 运行，打印 SDK 的版本号，如下图所示：



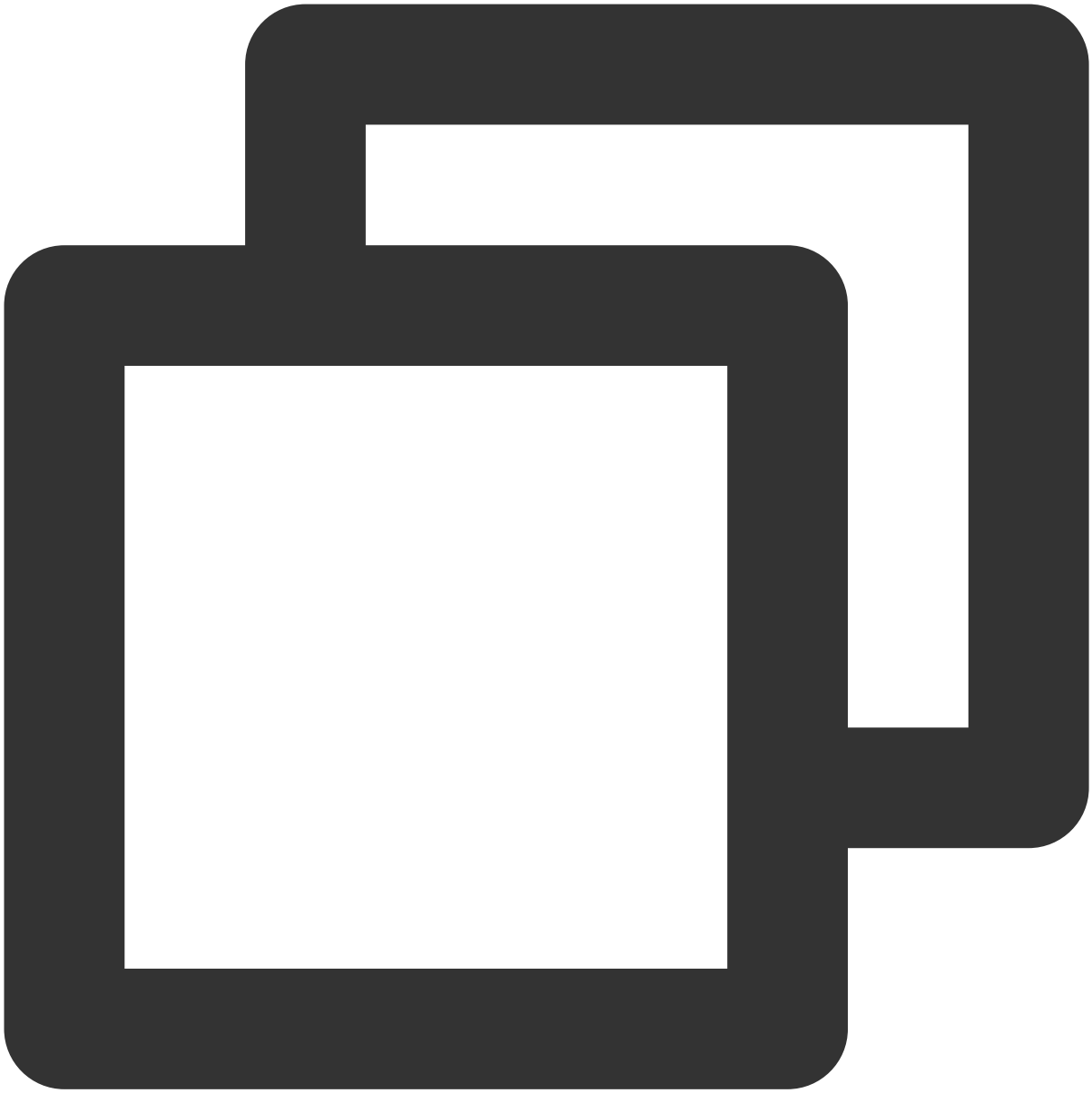
Mac 端集成

开发环境要求

操作系统：Mac10.10及以上版本。
开发环境：Qt Creator 4.10.3及以上版本，推荐使用 Qt Creator 4.13.3及以上。
开发框架：Based on Qt 5.10及以上。

操作步骤

本节以创建一个简单的 QTTest 项目为例，介绍如何在 Qt Creator 工程中集成 C++ 跨平台 SDK。
1. **下载 C++ 跨平台 SDK。**
1.1 下载 SDK，解压并打开文件。
1.2 在您的 QTTest 同级目录下新建一个空的 SDK 文件夹，将 SDK 中的 `TXLiteAVSDKTRTCMacx.x.x/SDK/TXLiteAVSDKTRTC_Mac.framework` 拷贝到与您 QTTest 工程目录同级目录的 SDK 文件夹下。
2. **配置 QTTest.pro。**
打开 QTTest 工程目录，使用一任意文本编辑器打开 `QTTest.pro` 文件，然后添加 SDK 相关引用：



```
INCLUDEPATH += $$PWD/.
DEPENDPATH += $$PWD/.

LIBS += "-F$$PWD/base/util/mac/usersig"
LIBS += "-F$$PWD/../SDK"
LIBS += -framework TXLiteAVSDK_TRTC_Mac
LIBS += -framework Accelerate
LIBS += -framework AudioUnit

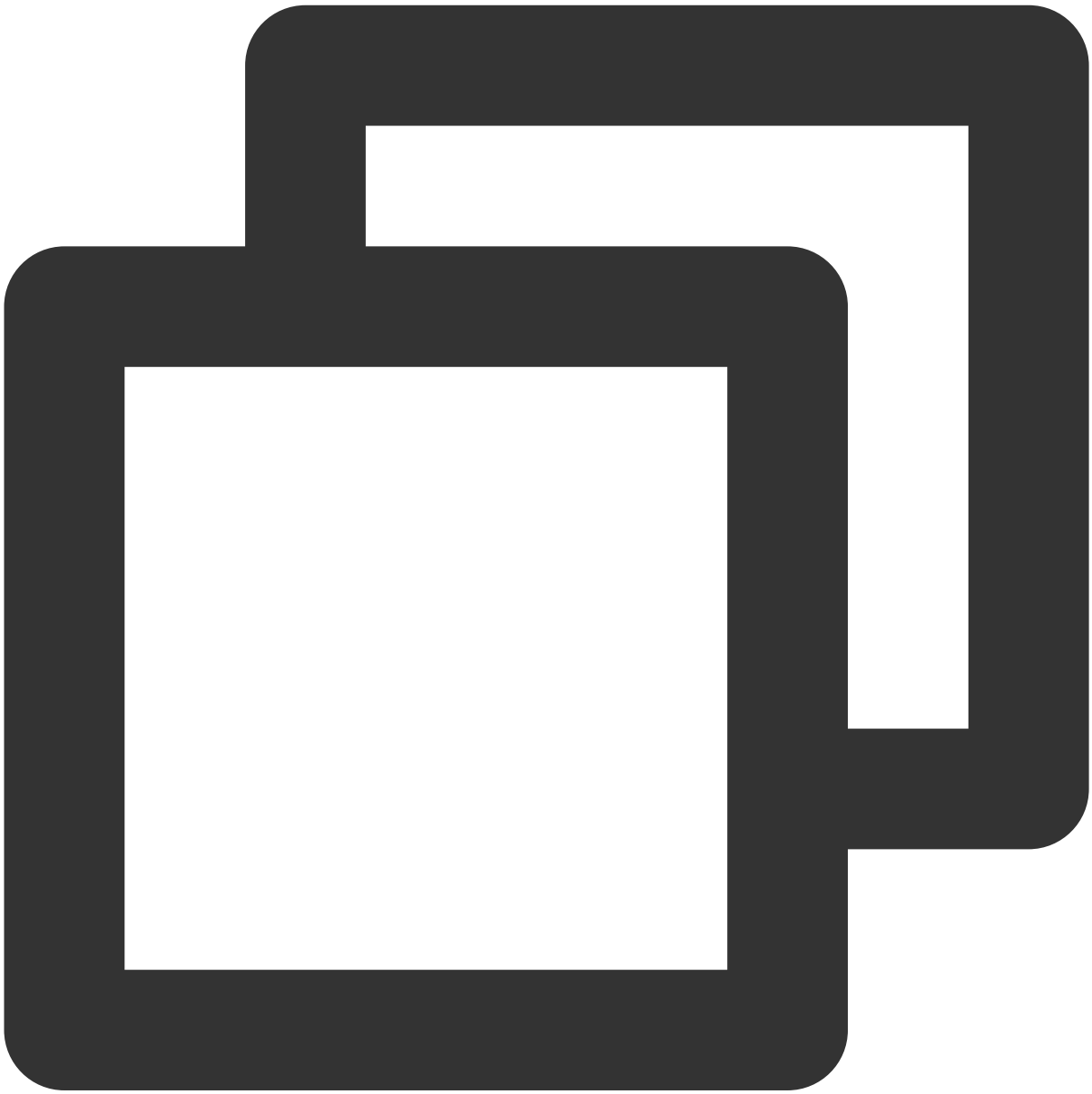
INCLUDEPATH += $$PWD/../SDK/TXLiteAVSDK_TRTC_Mac.framework/Headers/cpp_interface
```

版权所有：腾讯云计算（北京）有限责任公司

```
INCLUDEPATH += $$PWD/base/util/mac/usersig/include
DEPENDPATH += $$PWD/base/util/mac/usersig/include
```

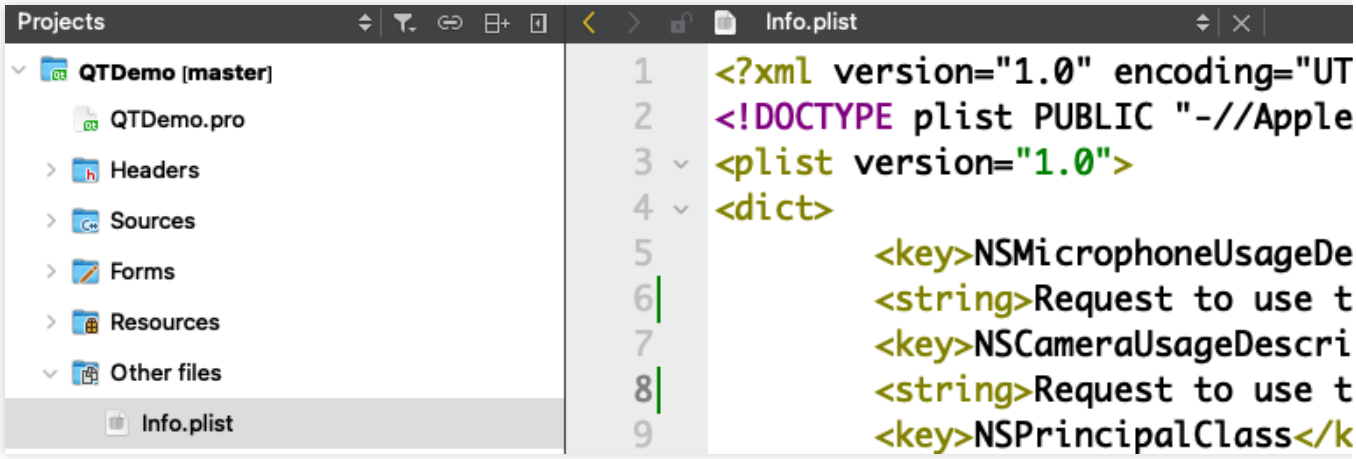
3. **授权摄像头和麦克风使用权限。**

因为 SDK 会使用您的摄像头和麦克风，所以您需要在对应的 `Info.plist` 添加对应的权限申请说明：



```
NSMicrophoneUsageDescription : 申请使用麦克风
NSCameraUsageDescription : 申请使用摄像头
```

如下图所示：



4. 引用 TRTC SDK。

您可以通过头文件 `#include "ITRTCCloud.h"` 直接引用。

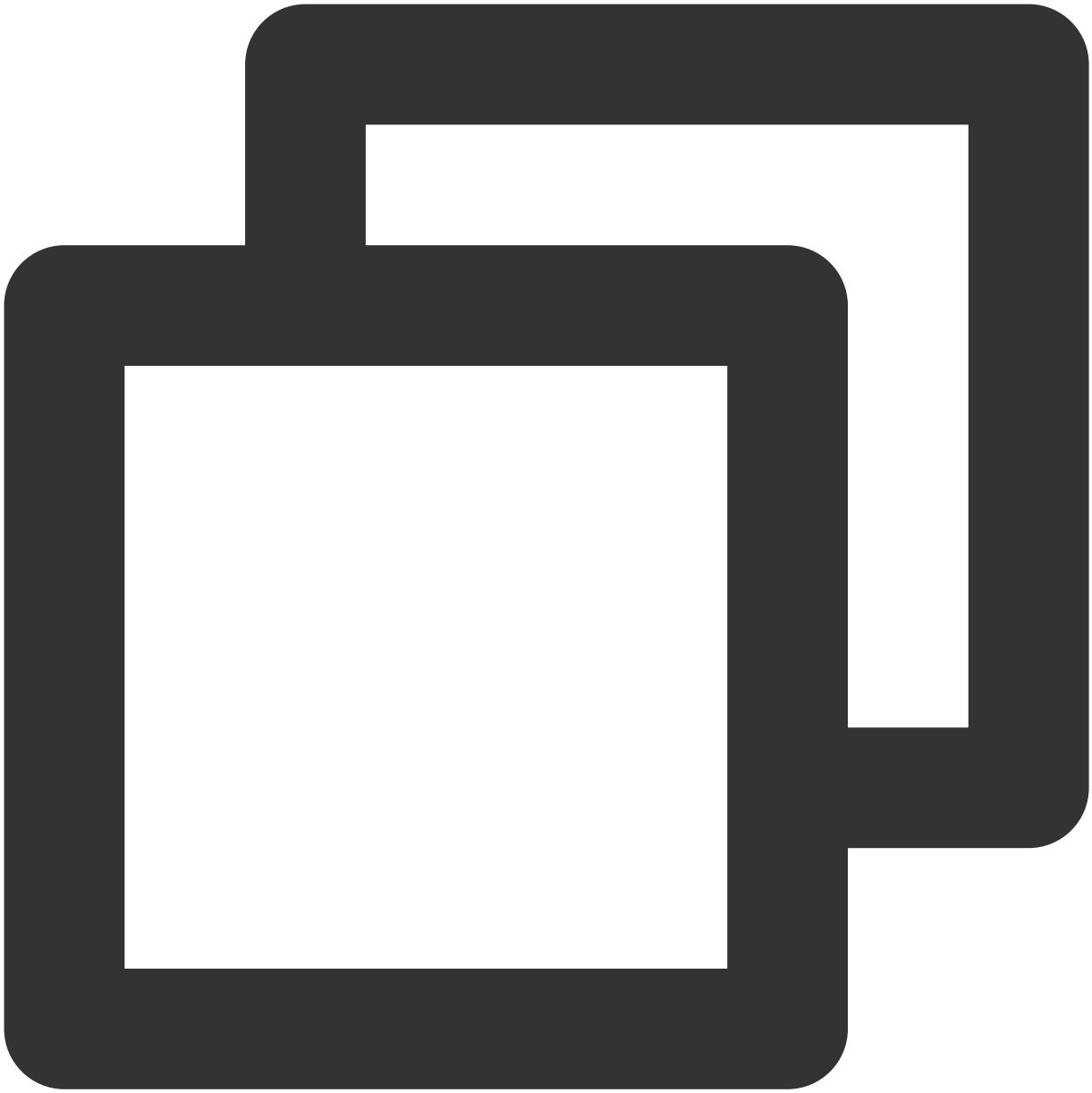
使用命名空间：C++ 全平台接口的方法、类型等均定义在 trtc 命名空间中，为了让代码更加简洁，建议您直接使用 trtc 命名空间。

说明：

至此您的集成工作已经完成，可以编译运行您的项目了。关于更多跨平台 SDK 的 API 使用 Demo，请下载 QTDemo 参考。

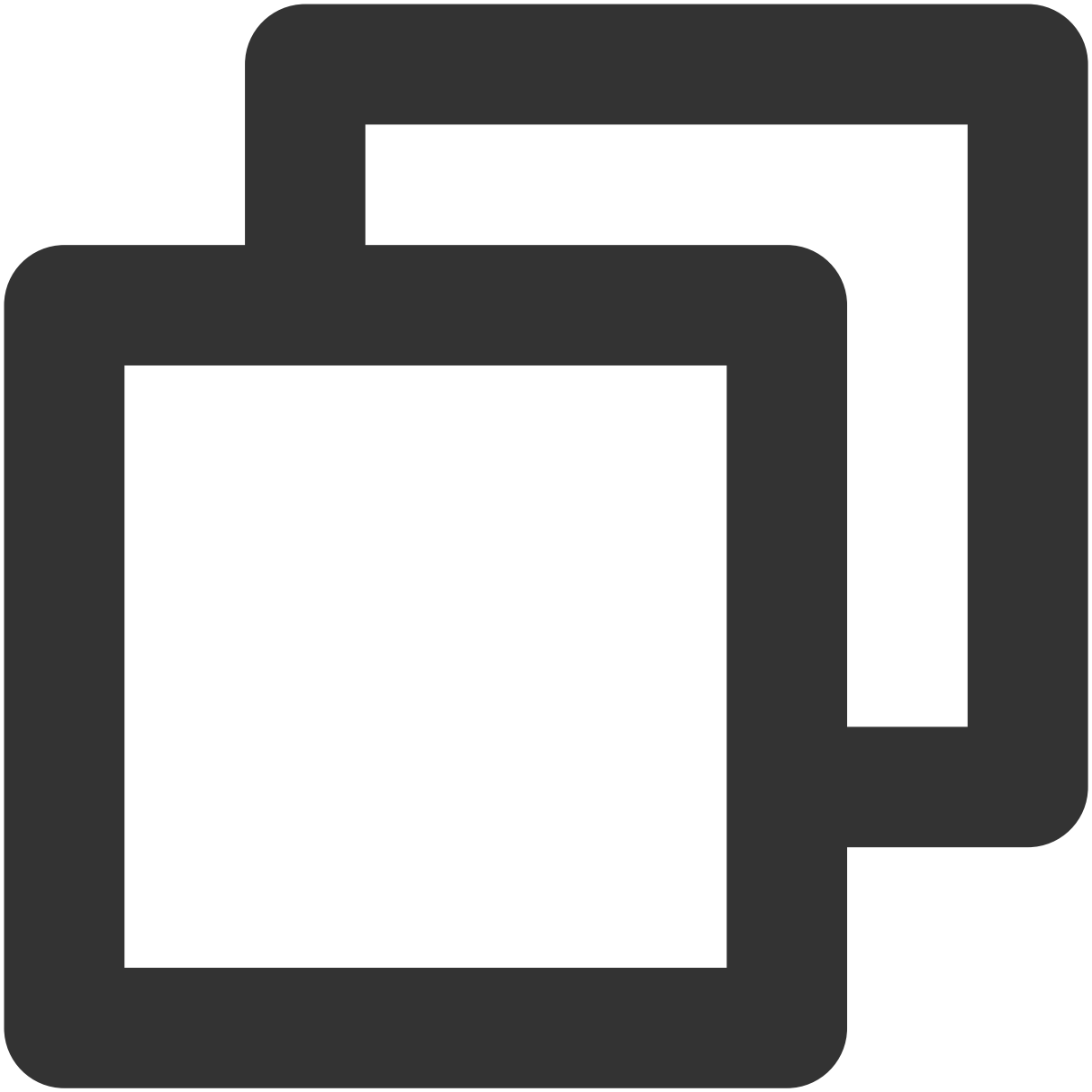
常见问题

若出现以下错误，请按照前面的工程配置，检查 SDK 头文件的目录是否正确添加。



```
fatal error C1083: 无法打开包括文件: “TRTCCloud.h”: No such file or directory
```

若出现以下错误，请按照前面的工程配置，检查 SDK 库目录和库文件是否正确添加。



```
error LNK2019: 无法解析的外部符号 "__declspec(dllimport) public: static class TXString
```

Unity

最近更新时间：2022-09-01 12:31:35

本文主要介绍如何快速地将腾讯云 TRTC SDK（Unity）集成到您的项目中，只要按照如下步骤进行配置，就可以完成 SDK 的集成工作。

环境要求

- Unity 建议版本： 2020.2.1f1c1。
- 目前支持 Android、iOS、Windows、Mac（Mac 还在内测中）平台。
- 需要包含 `Android Build Support` 、 `iOS Build Support` 、 `Winodows Build Support` 和 `MacOs Build Support` 模块。

- 其中 iOS 端开发还需要：
  - Xcode 11.0及以上版本。
  - 请确保您的项目已设置有效的开发者签名。

集成 SDK

1. 下载 SDK 及配套的 Demo 源码。
2. 解压后，把项目中的 `TRTCUnitySDK/Assets/TRTCSDK/SDK` 文件夹拷贝到您项目中的 Assets 目录下。

常见问题

Android 提示网络权限问题？

请将项目中 `/Assets/Plugins/AndroidManifest.xml` 文件放到同级目录下。

Android 没有音视频的权限？

Android 端的麦克风、摄像头权限要手动申请，具体方法请参见以下代码：

```
#if PLATFORM_ANDROID
if (!Permission.HasUserAuthorizedPermission(Permission.Microphone))
{
Permission.RequestUserPermission(Permission.Microphone);
}
```

版权所有：腾讯云计算（北京）有限责任公司

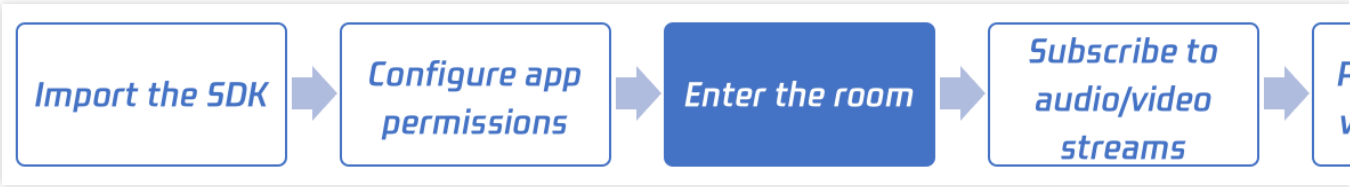
```
if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
{
Permission.RequestUserPermission(Permission.Camera);
}
#endif
```

02. 进入房间

Android&iOS&Windows&Mac

最近更新时间：2024-01-24 16:06:51

本文档主要介绍如何进入 TRTC 房间中，只有在进入音视频房间后，用户才能订阅房间中其他用户的音视频流，或者向房间中的其他用户发布自己的音视频流。



调用指引

步骤1：导入 SDK 并设置 App 权限

请参考文档 导入 SDK 到项目中 完成 SDK 的导入工作。

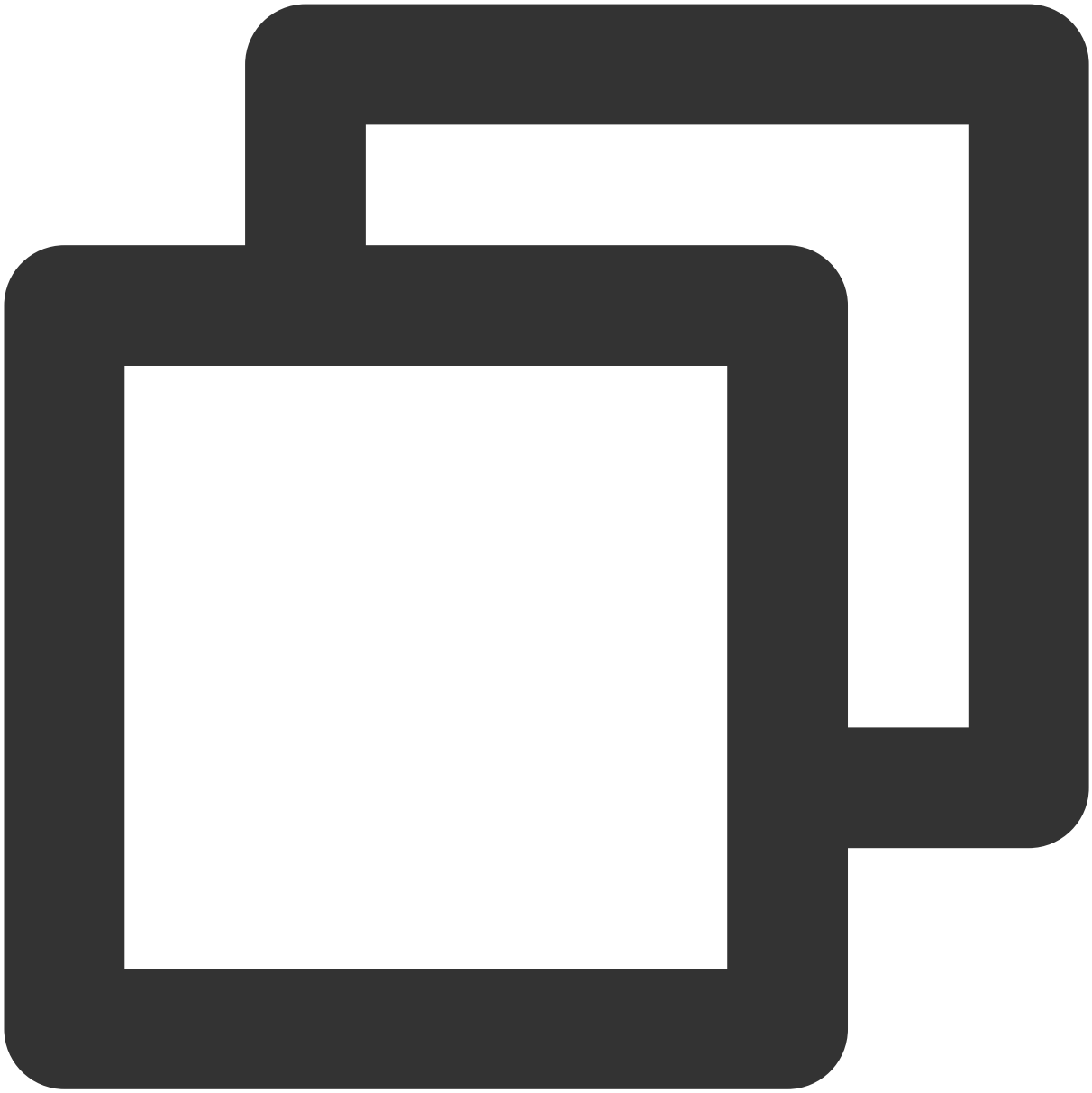
步骤2：创建 SDK 实例并设置事件监听器

调用各平台的初始化接口创建 TRTC 的对象实例。

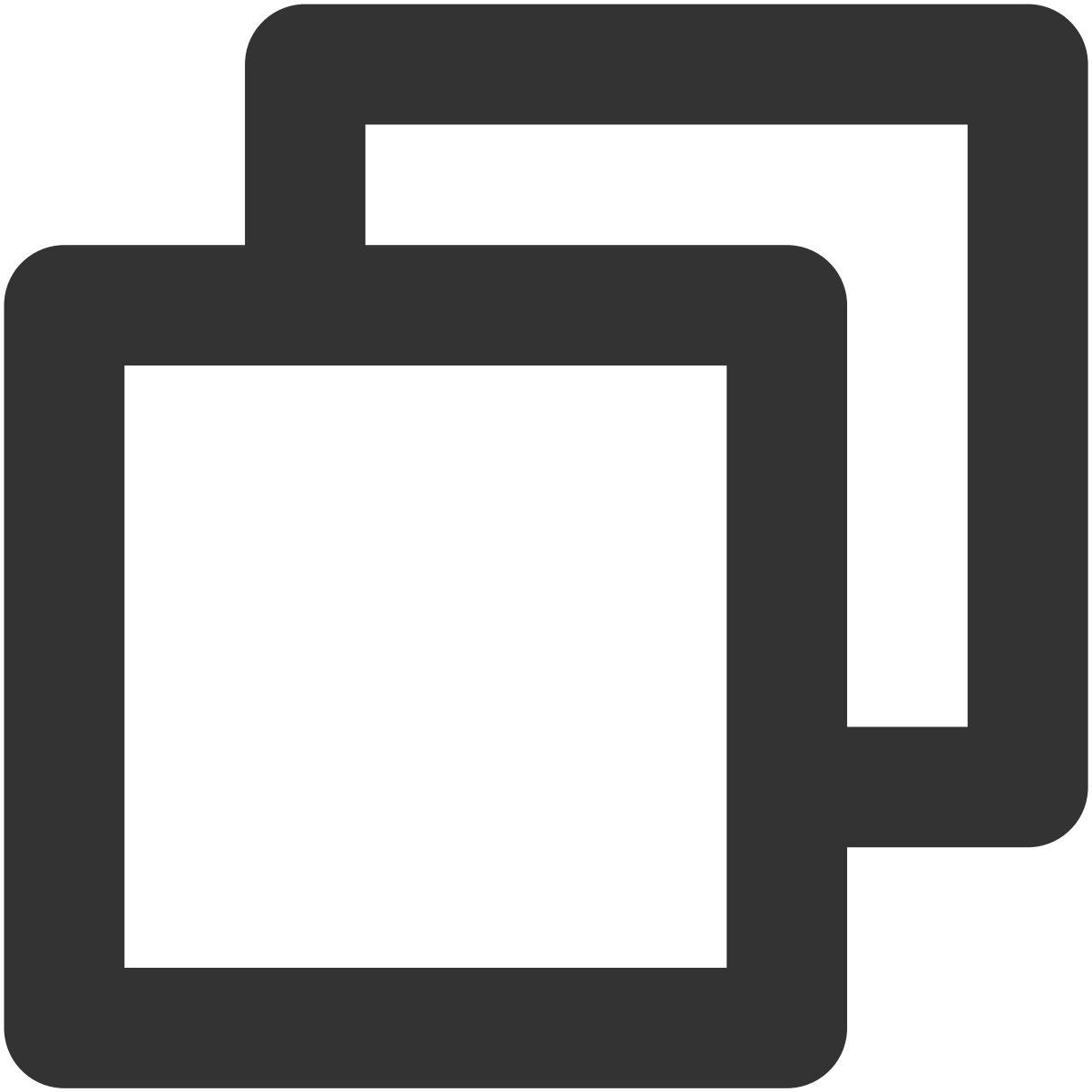
Android

iOS&Mac

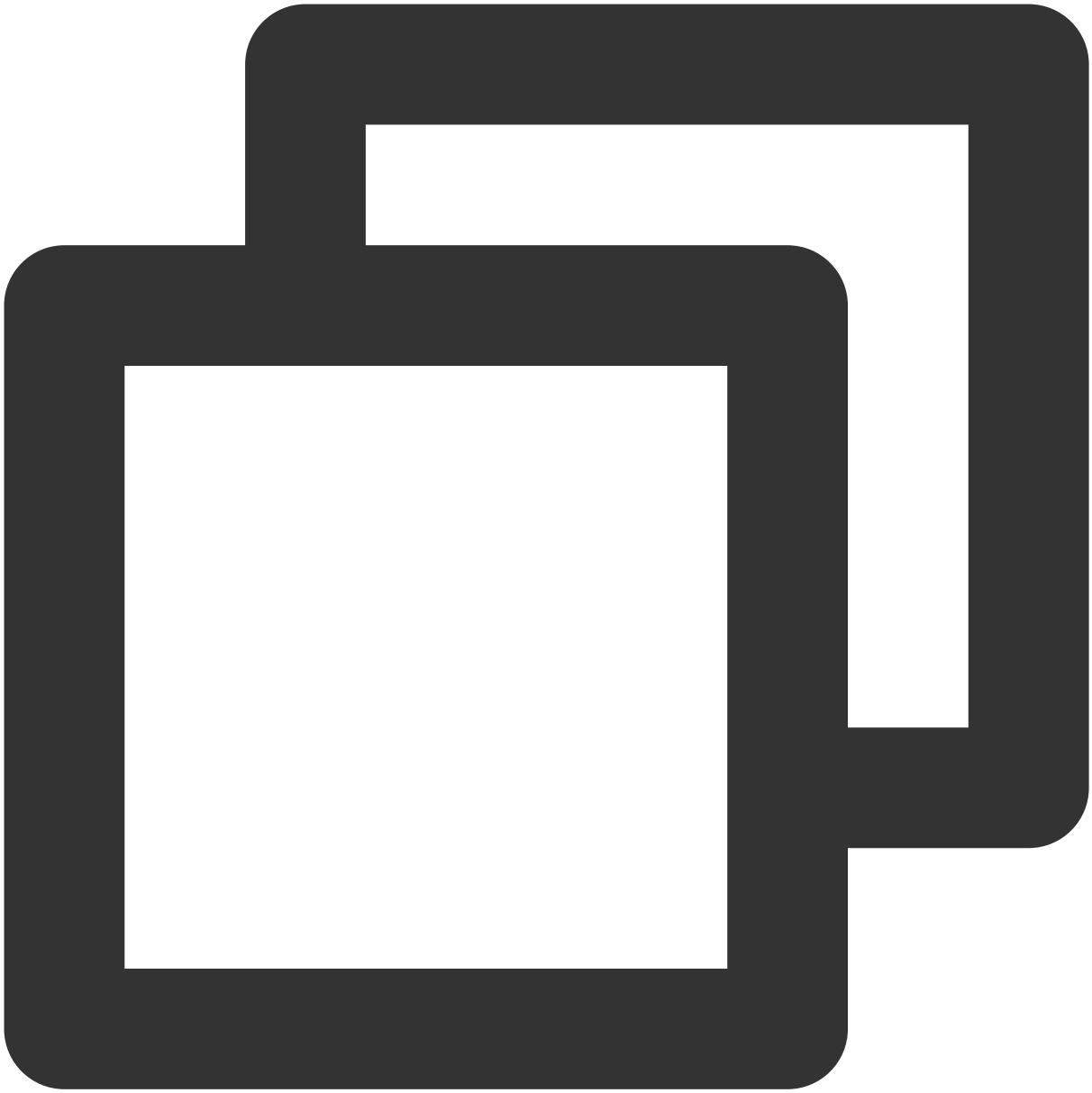
Windows



```
// 创建 SDK 实例（单例模式）并设置事件监听器
// Create trtc instance(singleton)  and set up event listeners
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
mCloud.setListener(this);
```



```
// 创建 SDK 实例（单例模式）并设置事件监听器
// Create trtc instance(singleton)  and set up event listeners
_trtcCloud = [TRTCCloud sharedInstance];
_trtcCloud.delegate = self;
```

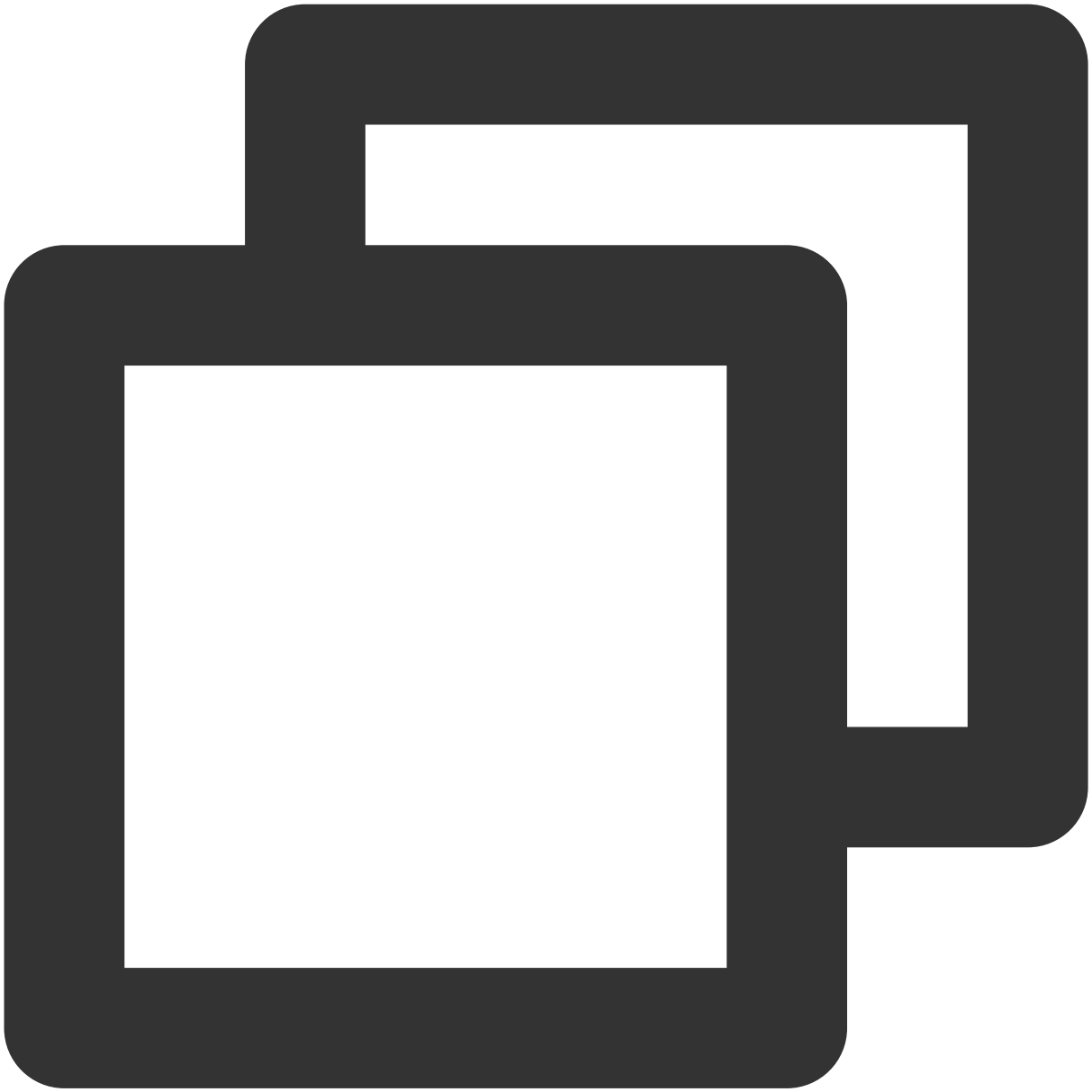


```
// 创建 SDK 实例（单例模式）并设置事件监听器
// Create trtc instance(singleton)  and set up event listeners
trtc_cloud_ = getTRTCShareInstance();
trtc_cloud_->addCallback(this);
```

步骤3：监听 SDK 的事件

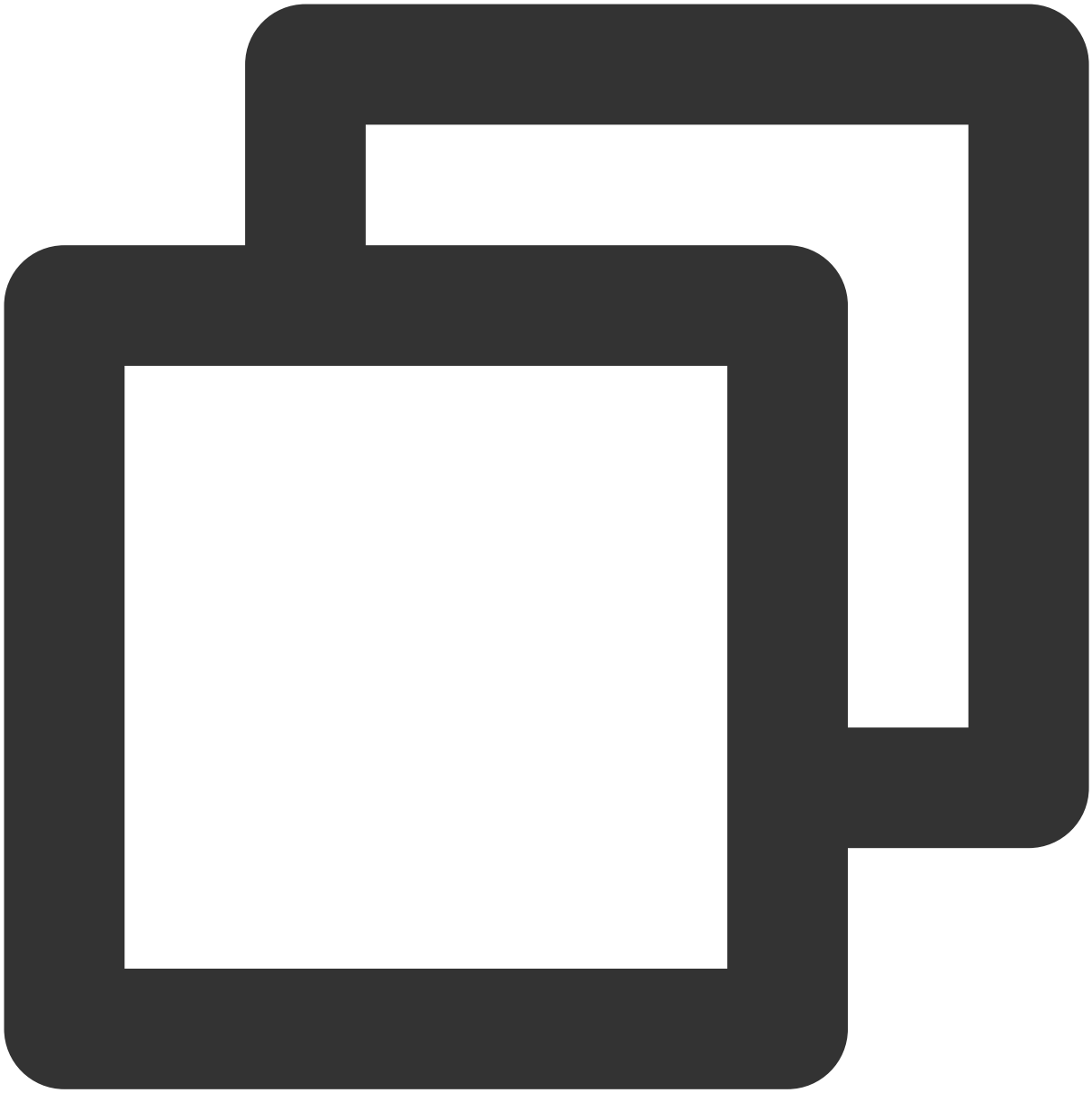
通过设置事件回调接口，您可以监听 SDK 在运行期间所发生的错误信息、警告信息、流量统计信息、网络质量信息以及各种音视频事件。

Android
iOS&Mac
Windows



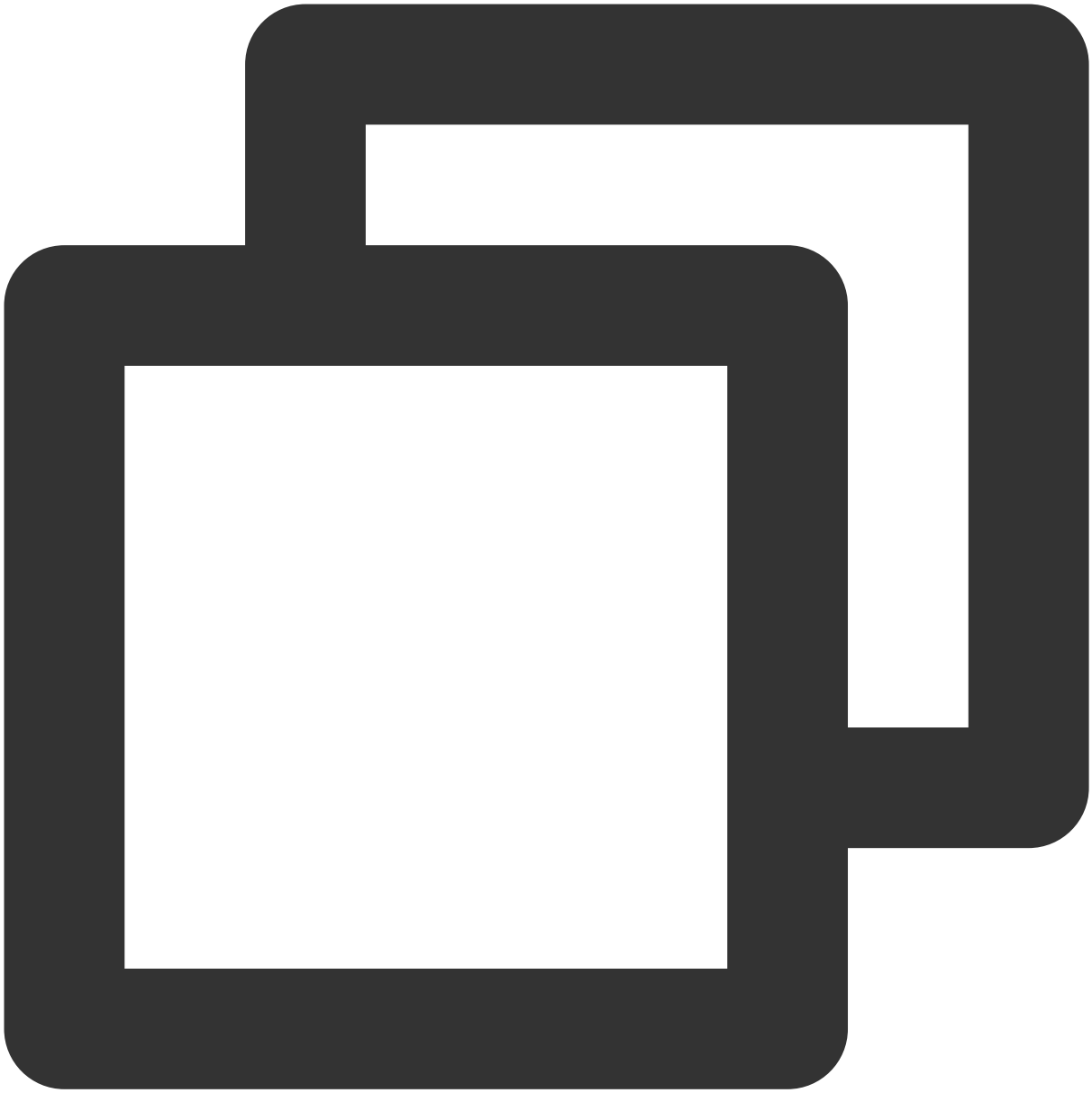
我们可以让自己的类继承 **TRTCCloudListener**，并重载 onError 函数，最后将 this 指针通过 **setListe**

<dx-code-holder data-codeindex="0"></dx-code-holder>



我们可以让自己的类继承 **TRTCCloudDelegate**，并重载 onError 函数，最后将 this 指针通过 TRTCCCl

```
<dx-code-holder data-codeindex="1"></dx-code-holder>
```



我们可以让自己的类继承 **ITRTCCloudCallback**，并重载 onError 函数，最后将 this 指针通过 **addCal**
<dx-code-holder data-codeindex="2"></dx-code-holder>

步骤4：准备进房参数 TRTCParams

在调用 enterRoom 接口时需要填写两个关键参数，即 `TRTCParams` 和 `TRTCAppScene`，接下来进行详细介绍：

参数一：TRTCAppScene

该参数用于指定您的应用场景，即**在线直播**还是**实时通话**：

实时通话：

包含 `TRTCAppSceneVideoCall` 和 `TRTCAppSceneAudioCall` 两个可选项，分别是视频通话和语音通话，该模式适合 1对1 的音视频通话，或者参会人数在 300 人以内的在线会议。

在线直播：

包含 `TRTCAppSceneLIVE` 和 `TRTCAppSceneVoiceChatRoom` 两个可选项，分别是视频直播和语音直播，该模式适合十万人以内的在线直播场景，但需要您在接下来介绍的 TRTCParams 参数中指定 **角色(role)** 这个字段，也就是将房间中的用户区分为 **主播(anchor)** 和 **观众(audience)** 两种不同的角色。

参数二：TRTCParams

TRTCParams 由很多的字段构成，但通常您只需要关心如下几个字段的填写：

参数名称	字段含义	补充说明	数据类型	填写示例
SDKAppID	应用 ID	您可以在 实时音视频控制台 中找到这个 SDKAppID，如果没有就单击“创建应用”按钮创建一个新的应用。	数字	1400000123
userId	用户 ID	即用户名，只允许包含大小写英文字母（a-z、A-Z）、数字（0-9）及下划线和连词符。注意 TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。	字符串	“denny” 或者 “123321”
userSig	进房鉴权票据	您可以使用 SDKAppID 和 userId 计算出 userSig，计算方法请参见 如何计算及使用 UserSig 。	字符串	eJyrVareCeYrSy1SsII...
roomId	房间号	数字类型的房间号。注意如果您想使用字符串类型的房间号，请使用 **strRoomId** 字段，而不要使用 roomId 字段，因为 strRoomId 和 roomId 不可以混用。	数字	29834
strRoomId	房间号	字符串类型的房间号。注意 strRoomId 和 roomId 不可以混用，“123” 和 123 在 TRTC 后台服务看来并不是同一个房间。	字符串	"29834"
role	角色	分为“主播”和“观众”两种角色，只有当 TRTCAppScene 被指定为 `TRTCAppSceneLIVE` 或 `TRTCAppSceneVoiceChatRoom` 这两种直播场景时才需要指定该字段。	枚举值	TRTCRoleAnchor 或 TRTCRoleAudience

注意：

TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。
每个端在应用场景 appScene 上必须要进行统一，否则会出现一些不可预料的问题。

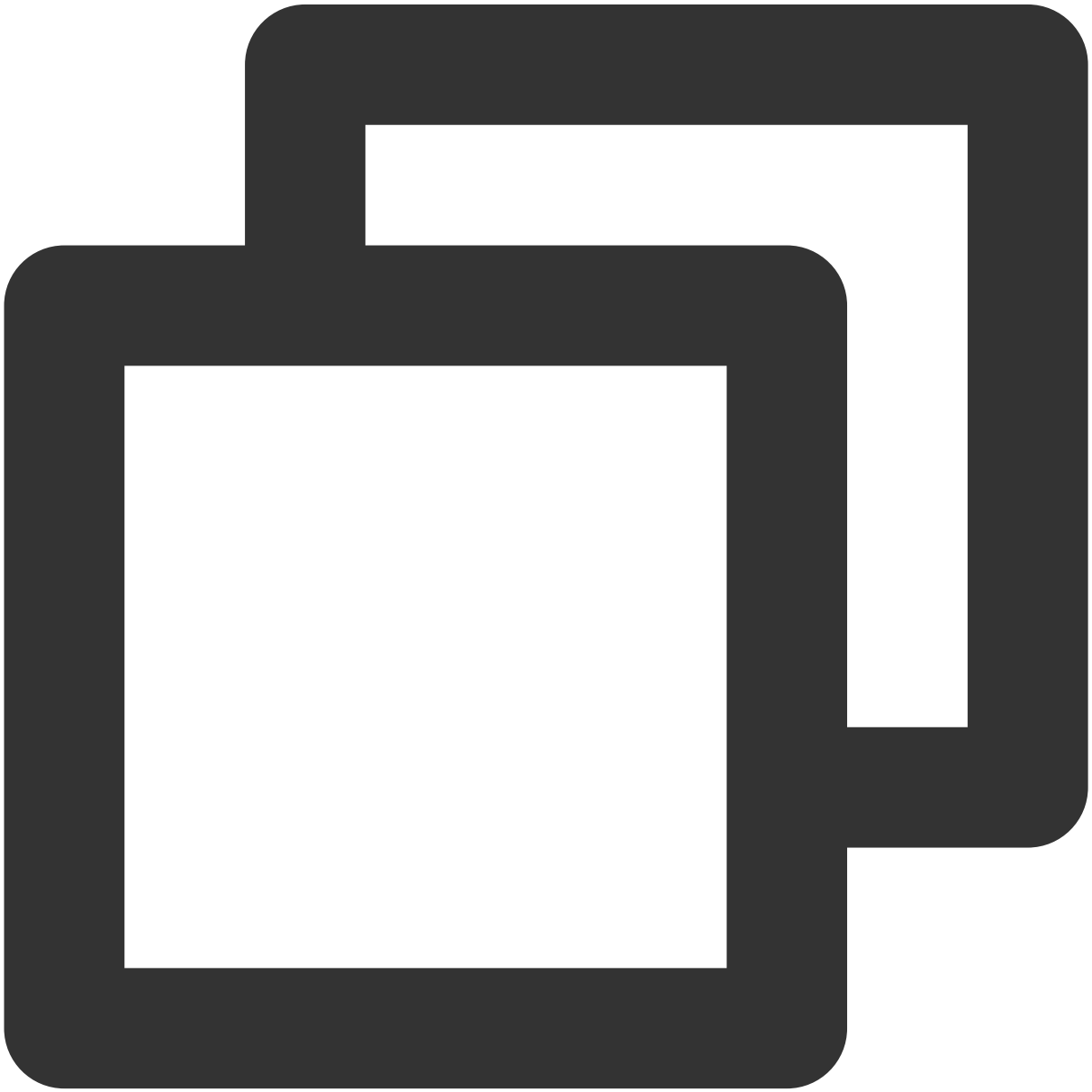
步骤5：进入房间（enterRoom）

在准备好步骤4中两个参数（TRTCAppScene 和 TRTCParams）后，就可以调用 enterRoom 接口函数进入房间了。

Android

iOS&Mac

Windows

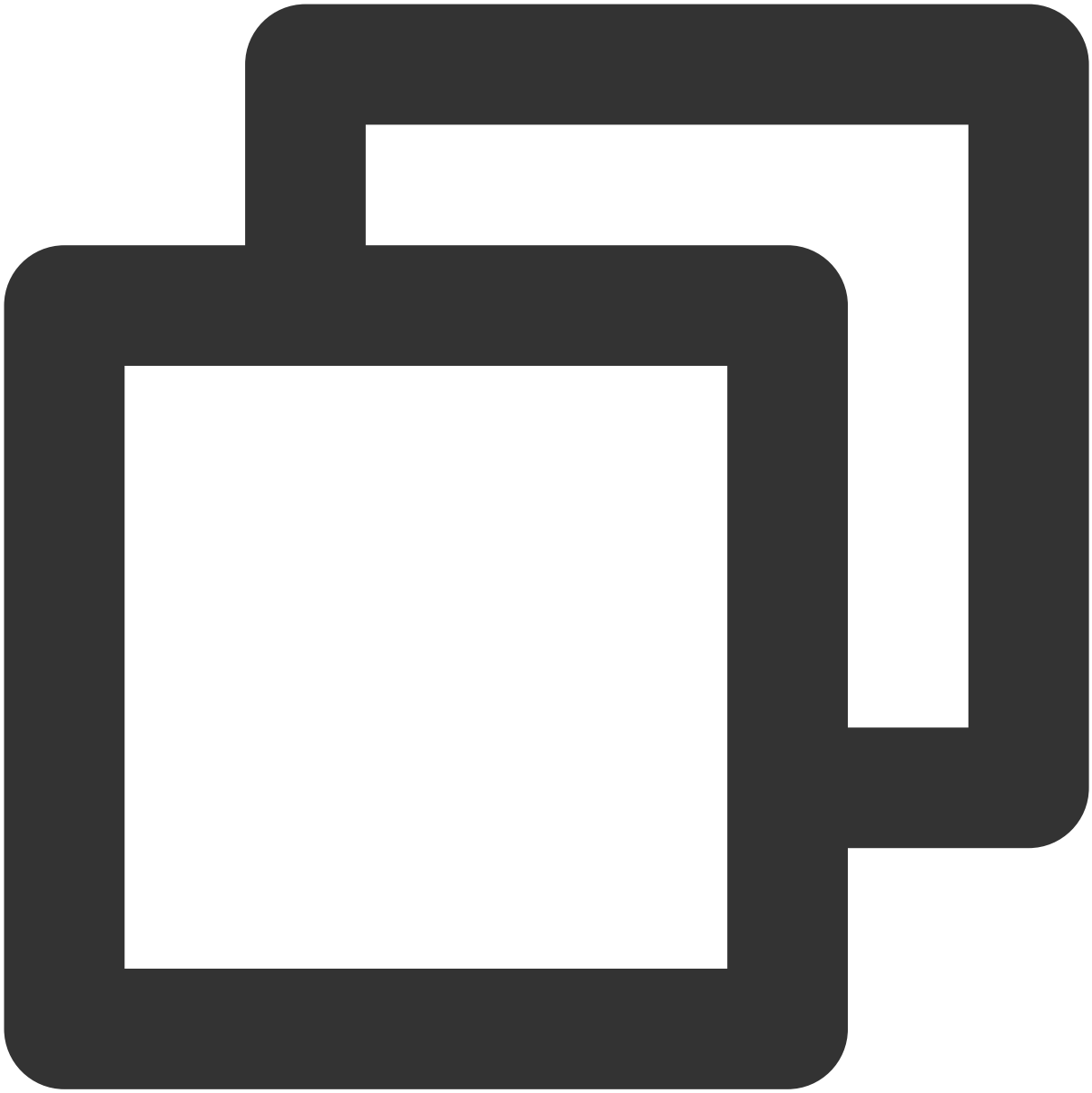


```
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
```

```
mCloud.setListener(mTRTCCloudListener);

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams param = new TRTCCloudDef.TRTCParams();
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.userId = "denny";       // Please replace with your own userid
params.roomId = 123321;        // Please replace with your own room number
params.userSig = "xxx";        // Please replace with your own userSig
params.role = TRTCCloudDef.TRTCRoleAnchor;

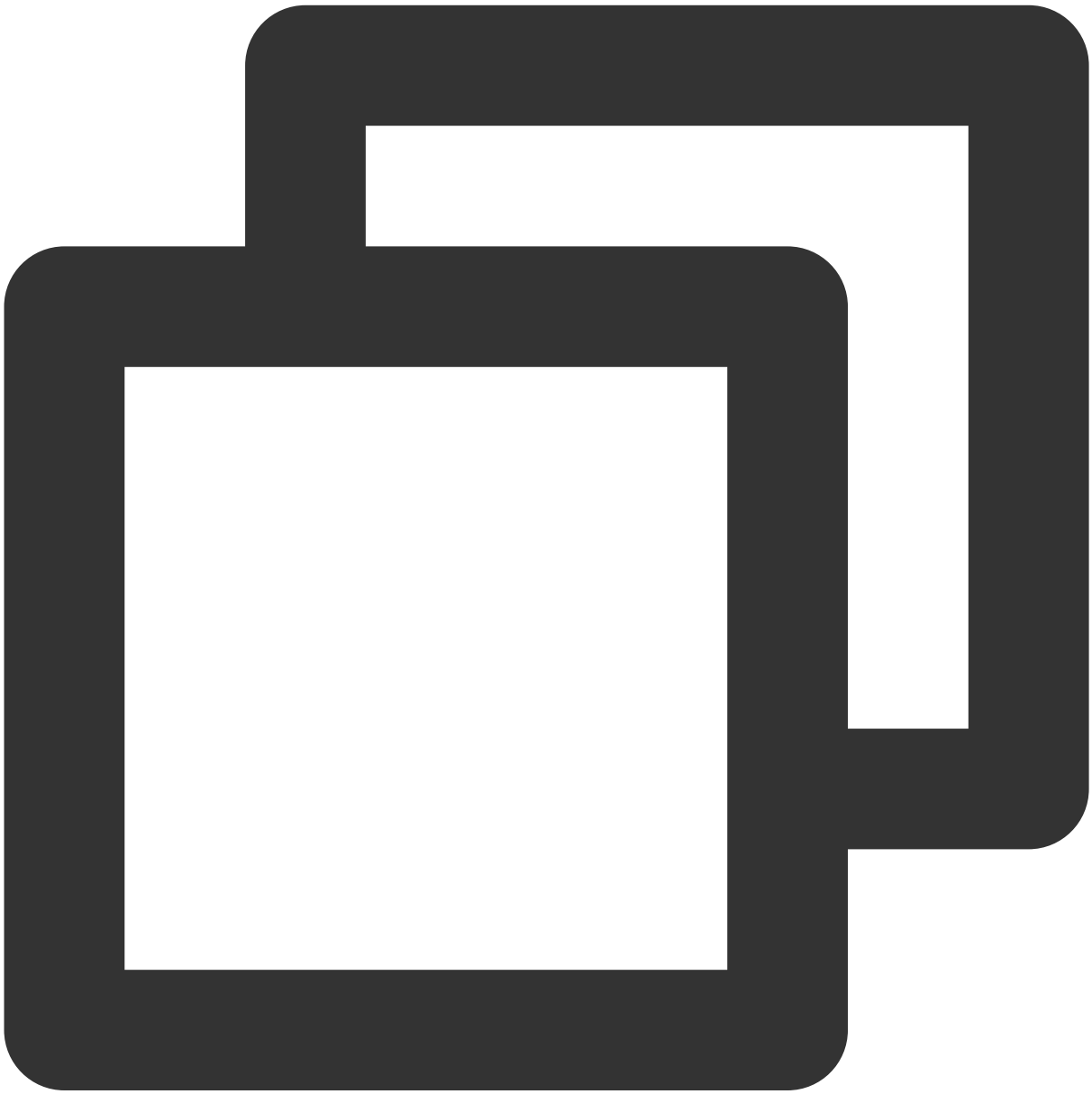
// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
mCloud.enterRoom(param, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```



```
self.trtcCloud = [TRTCCloud sharedInstance];
self.trtcCloud.delegate = self;

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
TRTCParams *params = [[TRTCParams alloc] init];
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.roomId = 123321; // Please replace with your own room number
params.userId = @"denny";   // Please replace with your own userid
params.userSig = @"xxx"; // Please replace with your own userSig
params.role = TRTCRoleAnchor;
```

```
// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
[self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
```



```
trtc_cloud_ = getTRTCShareInstance();
trtc_cloud_->addCallback(this);

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
```

```
liteav::TRTCParams params;
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.userId = "denny";       // Please replace with your own userid
params.roomId = 123321;        // Please replace with your own room number
params.userSig = "xxx";        // Please replace with your own userSig
params.role = liteav::TRTCRoleAnchor;

// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
trtc_cloud_->enterRoom(params, liteav::TRTCAppSceneLIVE);
```

事件回调

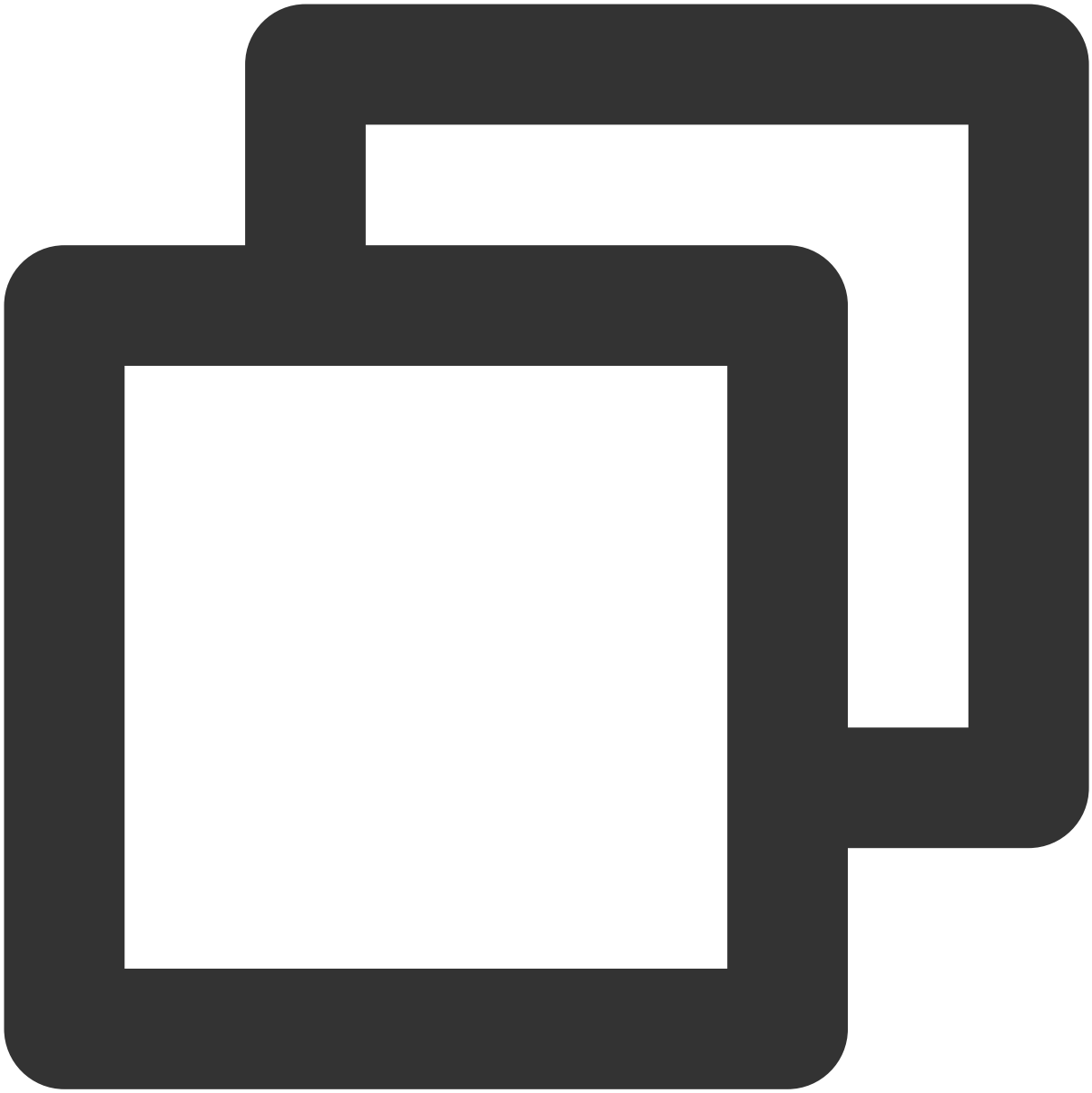
如果进入房间成功，SDK 会回调 onEnterRoom(result) 事件，其中 result 会是一个大于 0 的数值，代表加入房间所消耗的时间，单位为毫秒（ms）。

如果进入房间失败，SDK 同样会回调 onEnterRoom(result) 事件，但参数 `result` 会是一个负数，其数值为进房失败的错误码。

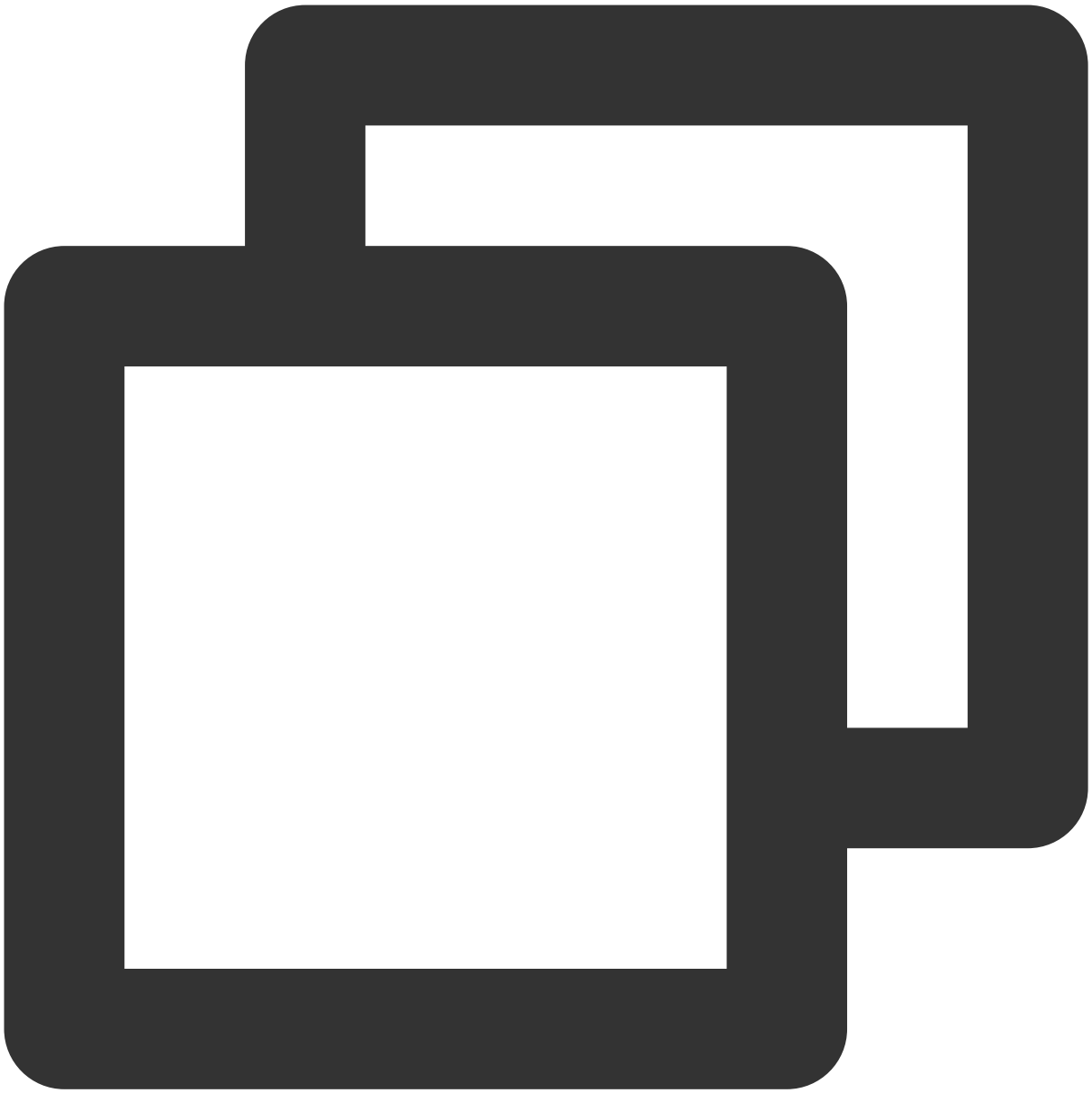
Android

iOS&Mac

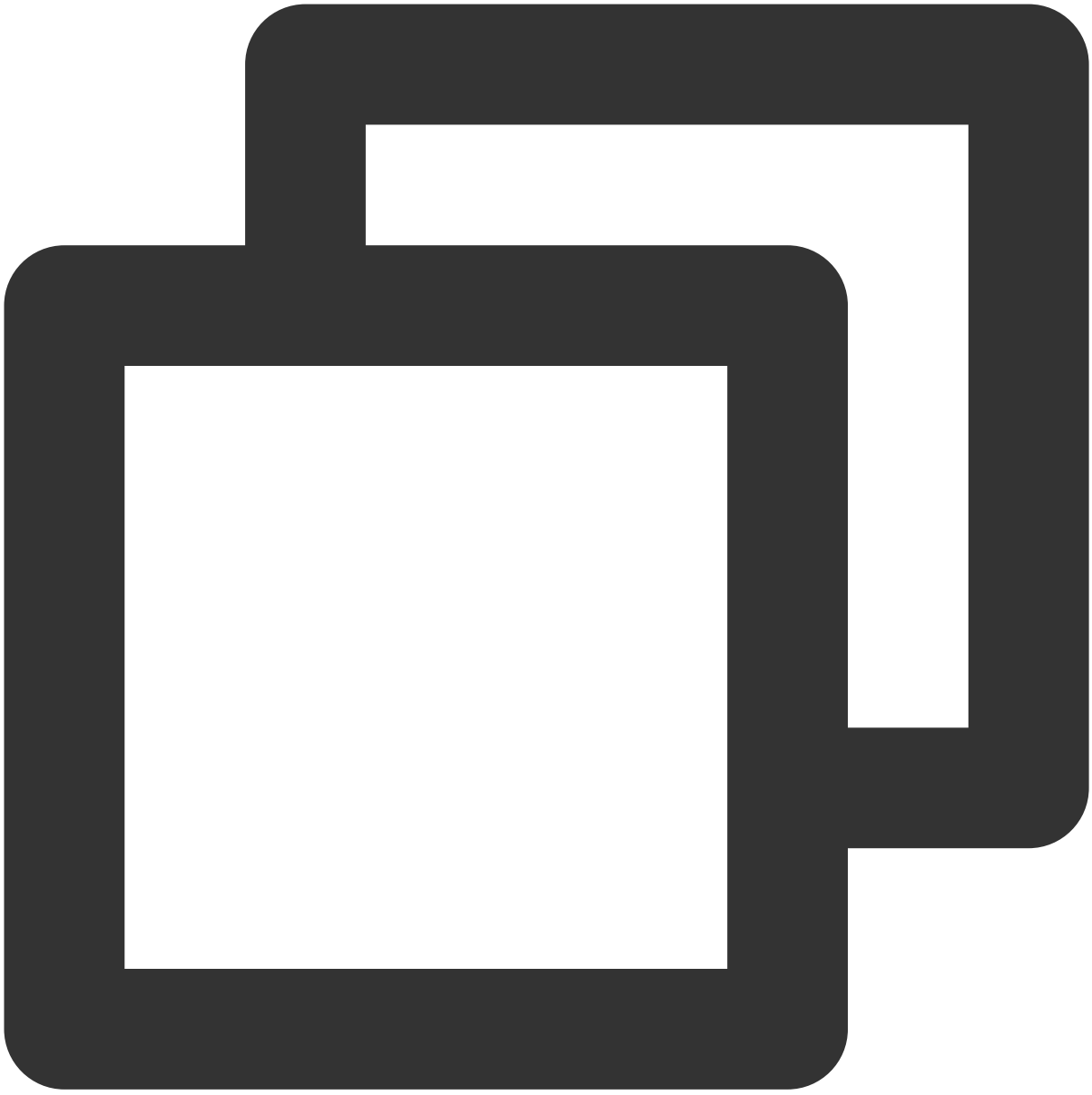
Windows



```
// 监听 SDK 的 onEnterRoom 事件并获知是否成功进入房间
// Listen to the onEnterRoom event of the SDK and learn whether the room is success
@Override
public void onEnterRoom(long     result) {
    if (result > 0) {
        Log.d(TAG, "Enter room succeed");
    } else {
        Log.d(TAG, "Enter room failed");
    }
}
```



```
// 监听 SDK 的 onEnterRoom 事件并获知是否成功进入房间
// Listen to the onEnterRoom event of the SDK and learn whether the room is success
- (void)onEnterRoom:(NSInteger)result {
    if (result > 0) {
        [self toastTip:@"Enter room succeed!"];
    } else {
        [self toastTip:@"Enter room failed!"];
    }
}
```



```
// 监听 SDK 的 onEnterRoom 事件并获知是否成功进入房间
// override to the onEnterRoom event of the SDK and learn whether the room is succe
void onEnterRoom(int result) {
    if (result > 0) {
        printf("Enter room succeed");
    } else {
        printf("Enter room failed");
    }
}
```

Web

最近更新时间：2024-06-28 15:03:32

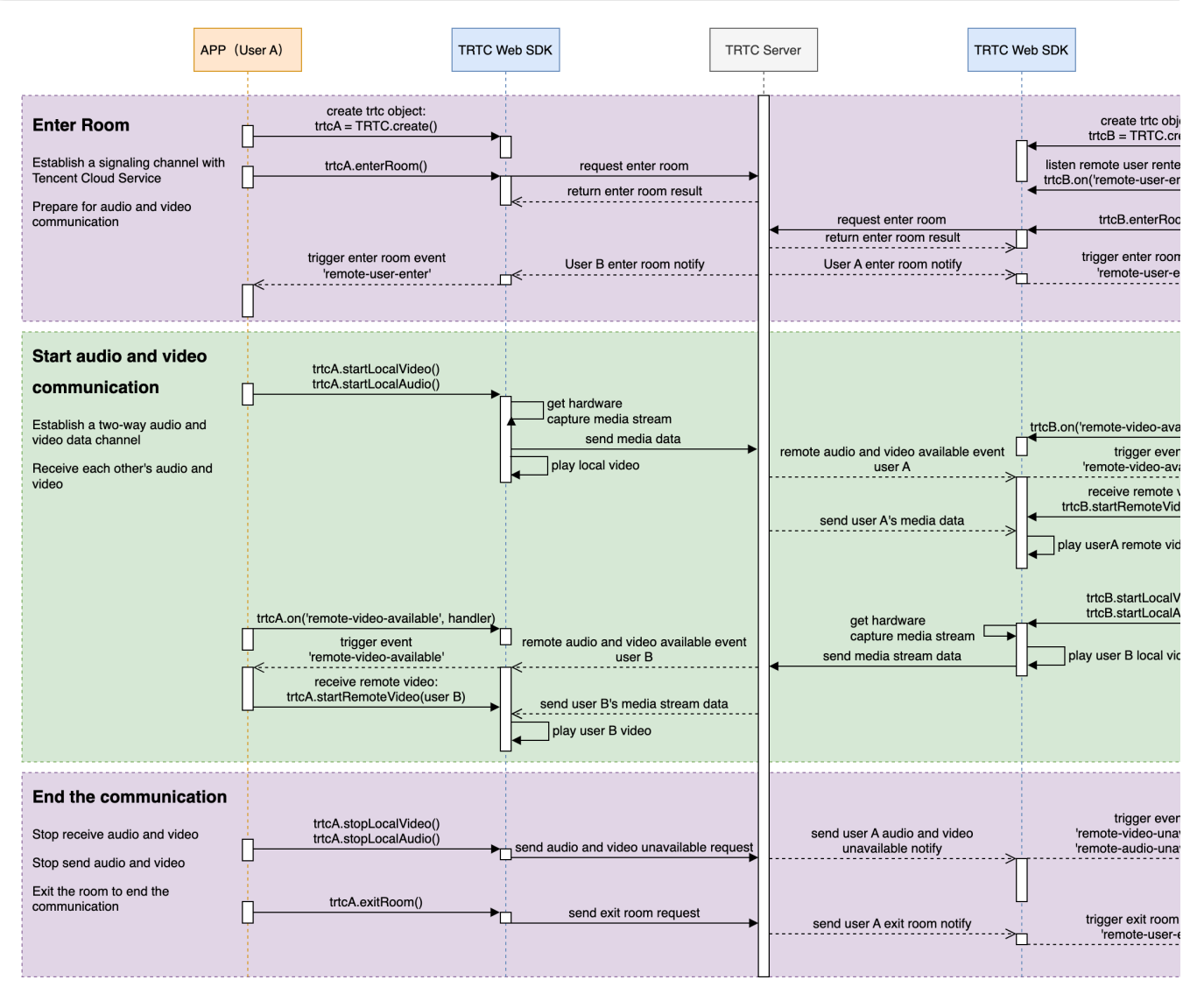
本文档主要介绍如何进入 TRTC 房间中，只有在进入音视频房间后，用户才能订阅房间中其他用户的音视频流，或者向房间中的其他用户发布自己的音视频流。



实现音视频通话基本逻辑

1. 调用 `TRTC.create()` 方法创建 `trtc` 对象。
2. 调用 `trtc.enterRoom()` 方法进入房间，进入后其他用户会收到 TRTC.EVENT.REMOTE_USER_ENTER 进房事件。
3. 在进入房间后，可以开启摄像头和麦克风并发布到房间。
调用 `TRTC.startLocalVideo()` 开启摄像头并发布到房间。
调用 `TRTC.startLocalAudio()` 开启麦克风并发布到房间。
4. 当一个远端用户发布了音视频后，SDK 默认情况下会自动播放远端音频。您需要通过如下方式来播放远端视频：
在进房前监听 TRTC.EVENT.REMOTE_VIDEO_AVAILABLE 事件， 就能收到所有远端用户的发布视频事件。
在事件回调函数中，调用 `trtc.startRemoteVideo()` 方法播放远端视频。

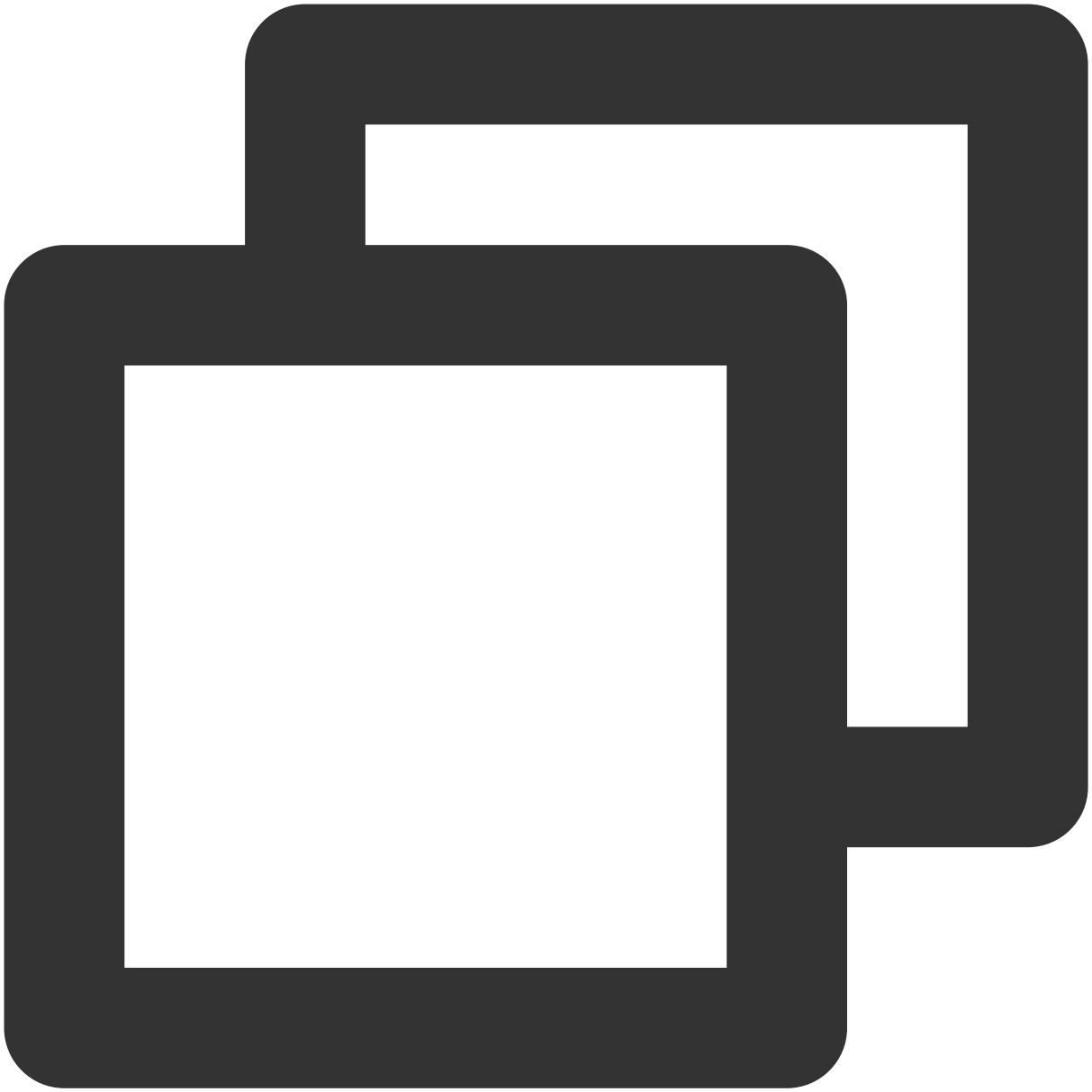
版权所有：腾讯云计算（北京）有限责任公司



步骤 1：创建 TRTC 对象

TRTC 类的实例代表一个本地客户端。TRTC 的对象方法提供了加入通话房间、预览本地摄像头、发布本地摄像头和麦克风、播放远端音视频等功能。

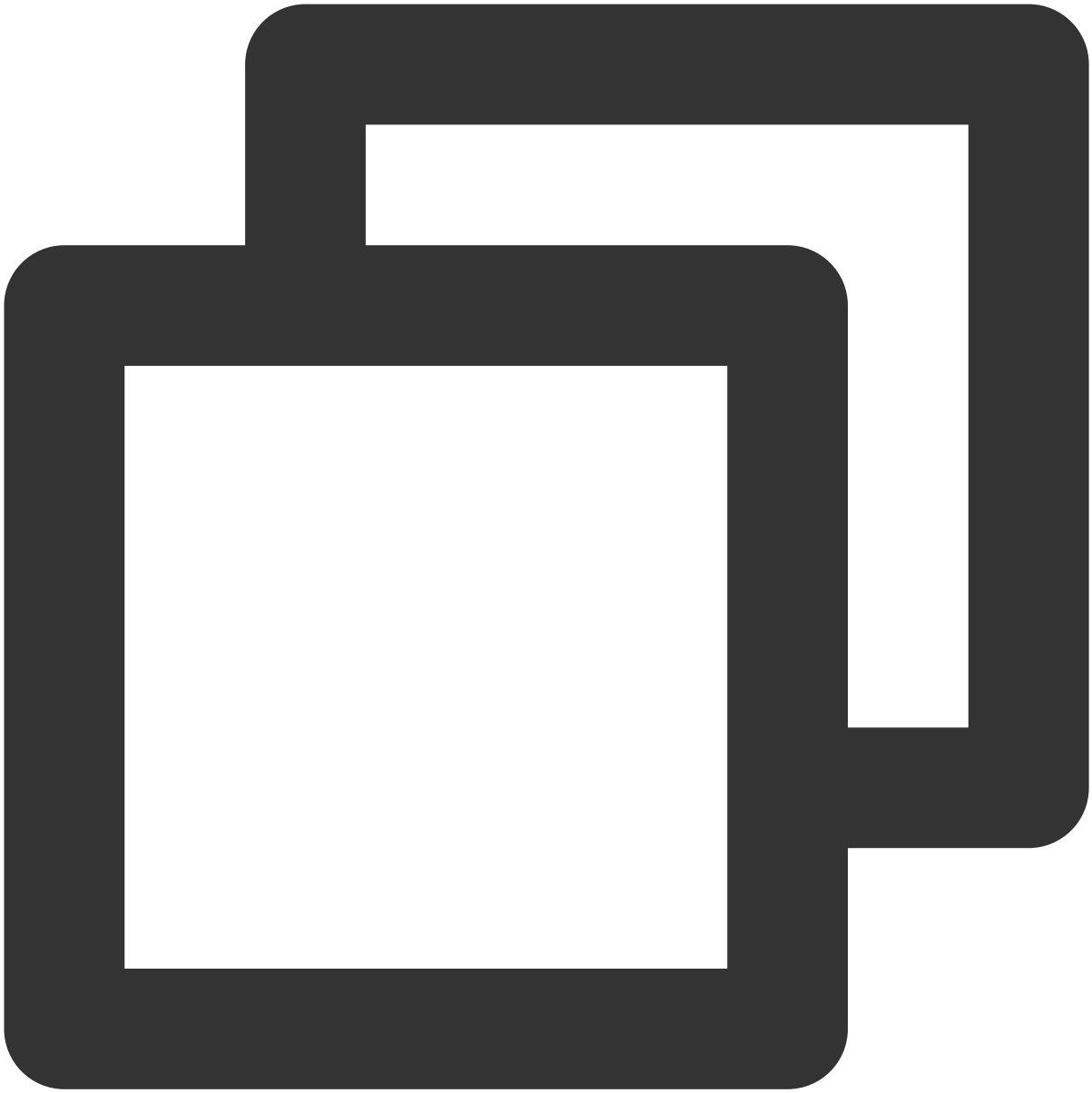
调用 `TRTC.create()` 方法创建 `TRTC` 对象。



```
const trtc = TRTC.create();
```

注意：

若您使用 Vue3 框架，此处还需要使用 markRaw 将 trtc 实例标记，避免 Vue 将 trtc 转为 Proxy 代理对象，出现一些不符合预期的行为。



```
import { markRaw } from 'vue';
const trtc = markRaw(TRTC.create());
```

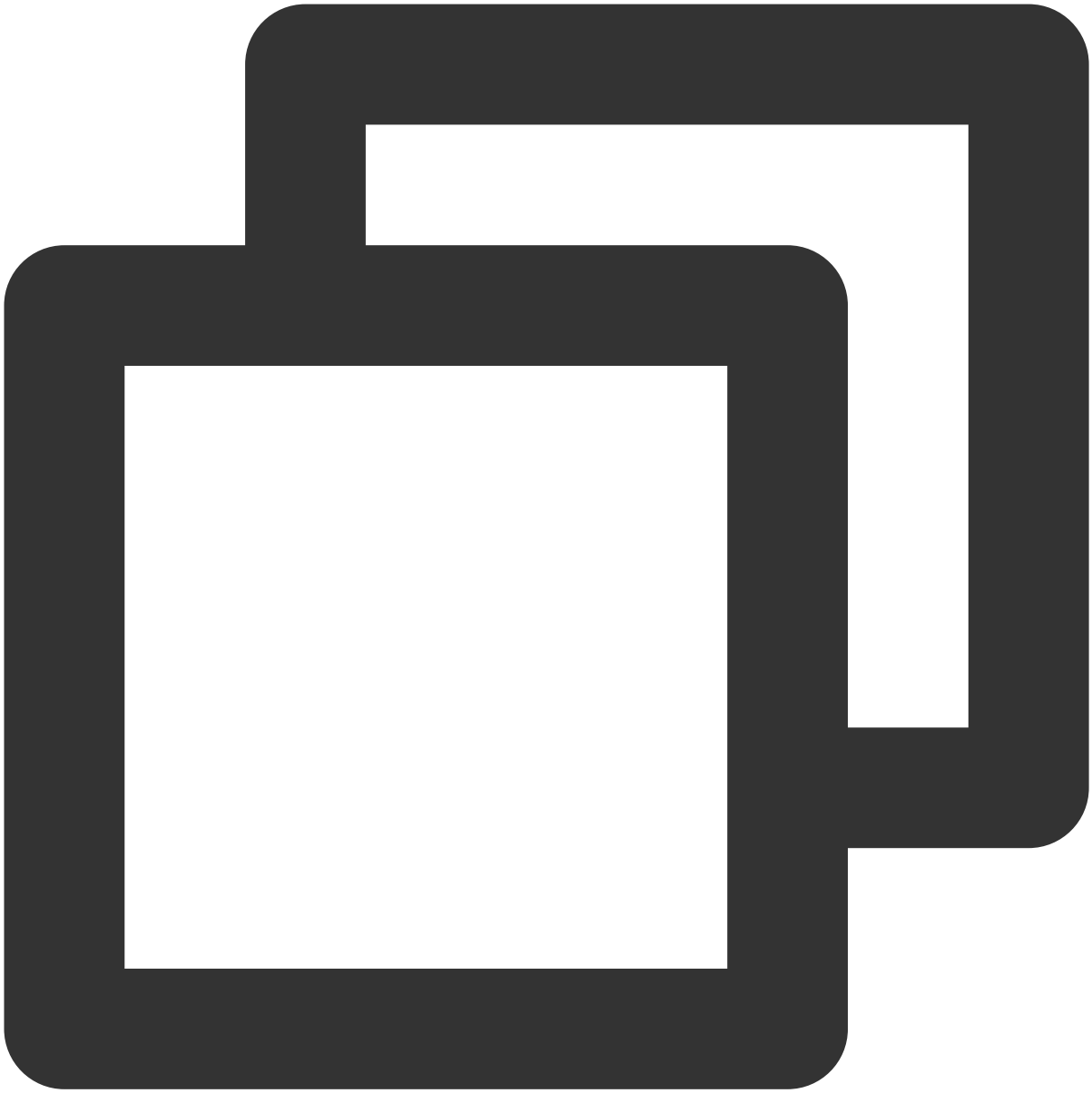
步骤 2：进入房间

调用 `trtc.enterRoom()` 方法进入房间。通常在 `开始通话` 按钮的点击回调里进行调用。关键参数：

参数名称	字段含	补充说明	数据类	填写示例

	义		型	
sdkAppId	应用 ID	您可以在实时音视频控制台中找到您的 sdkAppId，如果没有，则单击“创建应用”按钮创建一个新的应用。	number	1400000123
userId	用户 ID	即用户名，只允许包含大小写英文字母（a-z、A-Z）、数字（0-9）及下划线和连词符。注意 TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。	string	“mike” 或者 “123321”
userSig	进房鉴权票据	您可以使用 sdkAppId 和 userId 计算出 userSig，计算方法请参见 如何计算及使用 UserSig 。	string	eJyrVareCeYrSy1SsII...
roomId	房间号	数字类型的房间号。 注意：如果您想使用字符串类型的房间号，请使用 **strRoomId** 字段，而不要使用 roomId 字段，因为 strRoomId 和 roomId 不可以混用。	number	29834
strRoomId	房间号	字符串类型的房间号。**注意：strRoomId 和 roomId 不可以混用，数字类型房间 123 和字符类型房间 "123" 是不同的房间，不互通。**	string	"29834"
scene	应用场景	有如下两种模式： - `rtc` ：实时通话模式 - `live` ：在线直播模式	string	'rtc' 或者 'live'
role	角色	只在 `live` 场景下才需要填写该字段，有如下两种角色： `anchor` ：“主播”角色，可以推流和拉流。 `audience` ：“观众”角色，只能拉流观看，无法推流。	string	'anchor' 或者 'audience'

更详细的参数说明参考接口文档 `trtc.enterRoom()` 。

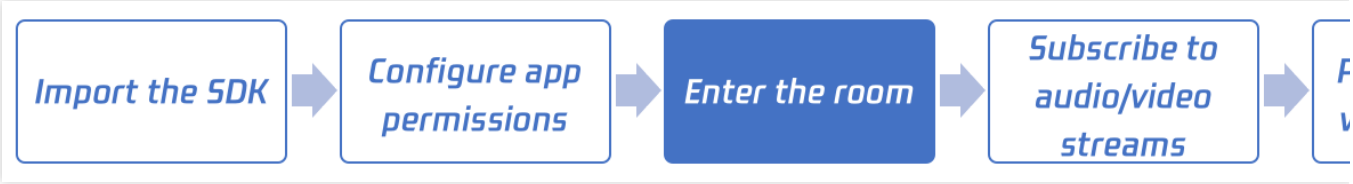


```
try {
  await trtc.enterRoom({ roomId: 8888, scene:'rtc', sdkAppId, userId, userSig });
  console.log('进房成功');
} catch (error) {
  console.error('进房失败 ' + error);
}
```

Electron

最近更新时间：2024-01-24 16:05:42

本文档主要介绍如何进入 TRTC 房间中，只有在进入音视频房间后，用户才能订阅房间中其他用户的音视频流，或者向房间中的其他用户发布自己的音视频流。

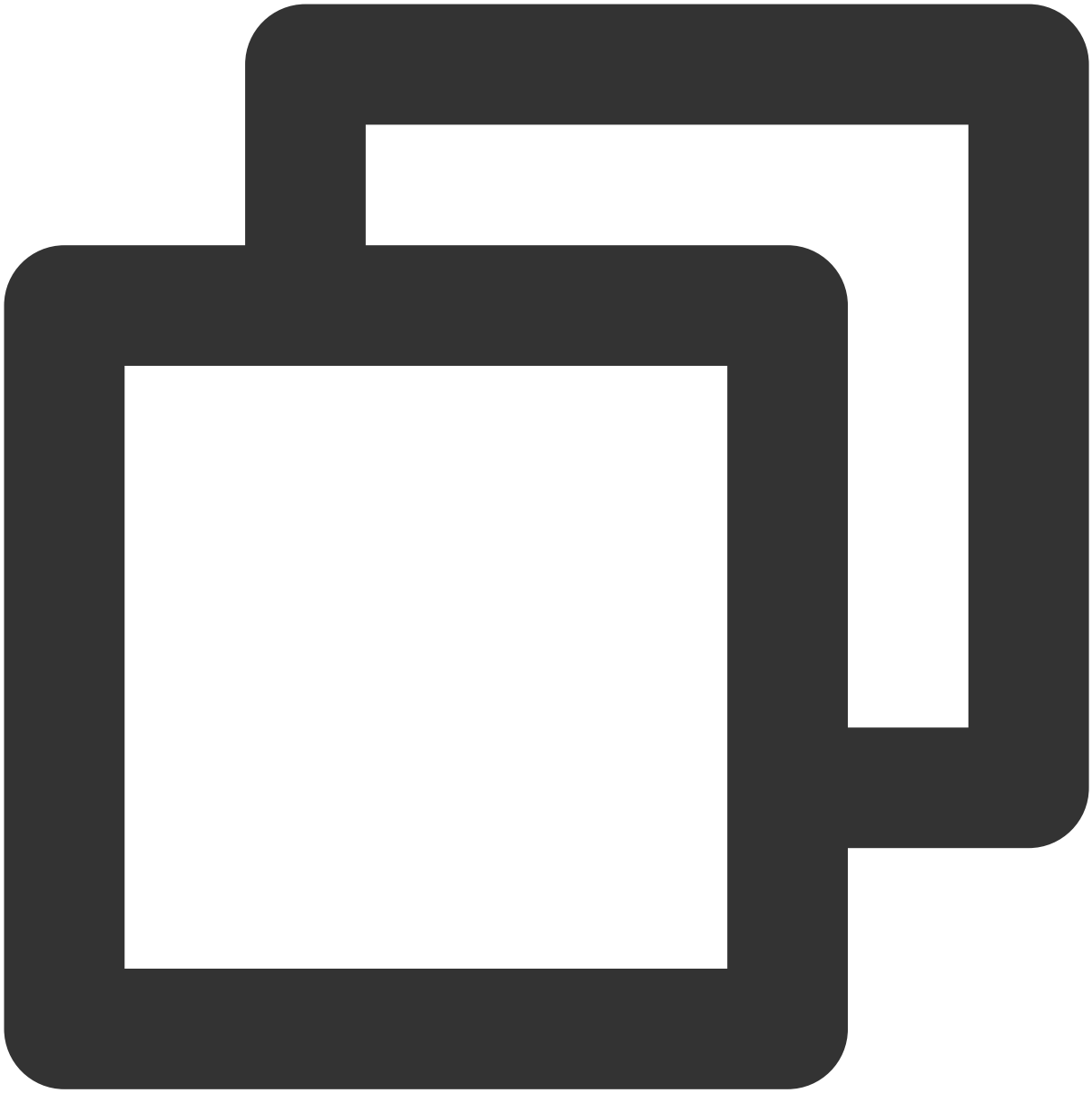


调用指引

步骤1：导入 SDK

请参考文档 导入 SDK 到项目中 完成 SDK 的导入工作。

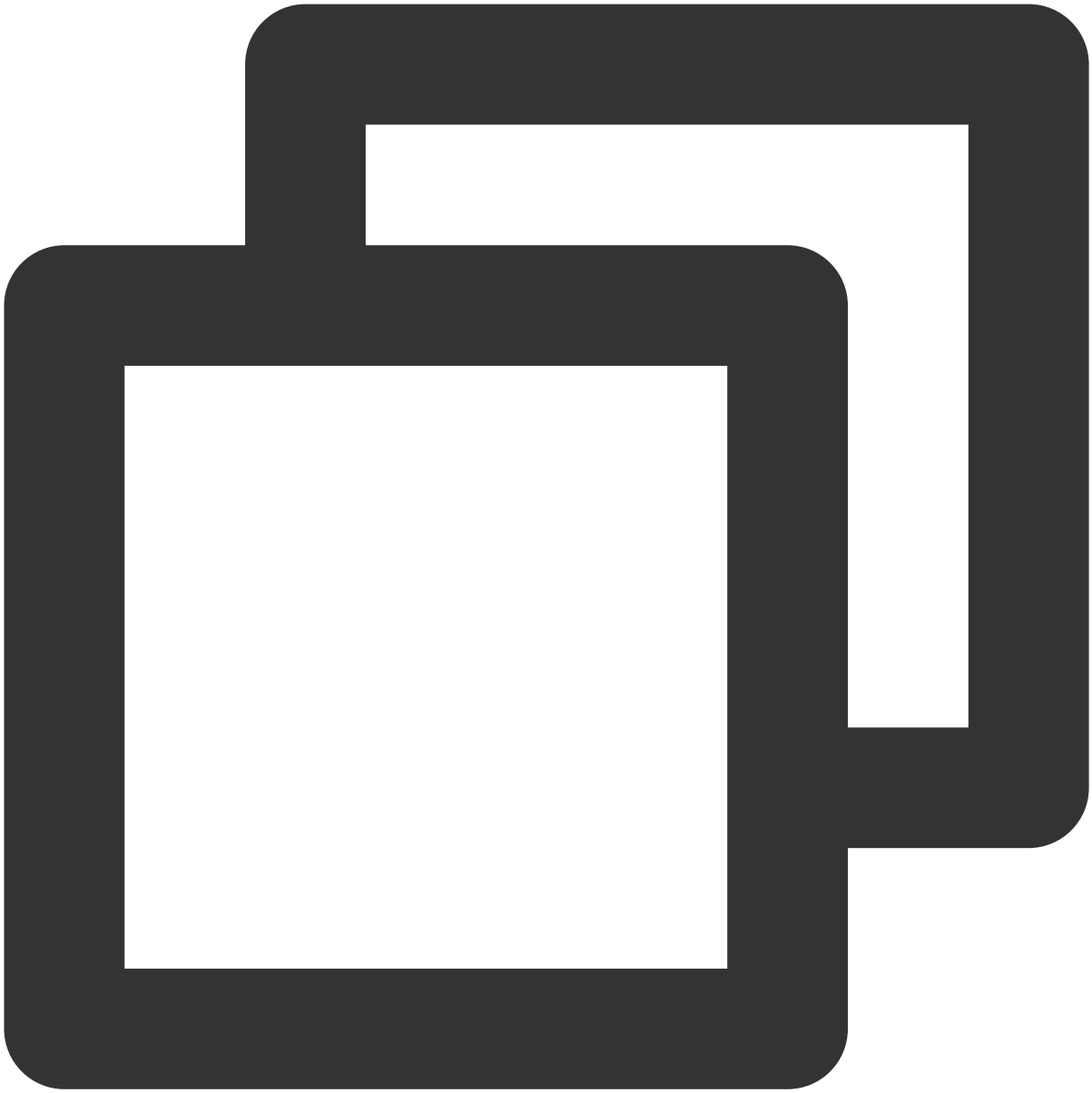
步骤2：创建 SDK 实例



```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
```

步骤3：监听 SDK 的事件

通过设置事件回调接口，您可以监听 SDK 在运行期间所发生的错误信息、警告信息、流量统计信息、网络质量信息以及各种音视频事件。



```
function onError(errCode, errMsg) {
  // errorCode 可参考 https://cloud.tencent.com/document/product/647/32257#.E9.94.99
  console.log(errCode, errMsg);
}

function onWarning(warningCode, warningMsg) {
  // warningCode 可参考 https://cloud.tencent.com/document/product/647/32257#.E8.AD.
  console.log(warningCode, warningMsg);
}

rtcCloud.on('onError', onError);
```

```
rtcCloud.on('onWarning', onWarning);
```

步骤4：准备进房参数 TRTCParams

在调用 enterRoom 接口时需要填写两个关键参数，即 `TRTCParams` 和 `TRTCAppScene`，接下来进行详细介绍：

参数一：TRTCAppScene

该参数用于指定您的应用场景，即**在线直播**还是**实时通话**：
实时通话：包含 `TRTCAppSceneVideoCall` 和 `TRTCAppSceneAudioCall` 两个可选项，分别是视频通话和语音通话，该模式适合 1对1 的音视频通话，或者参会人数在 300 人以内的在线会议。
在线直播：包含 `TRTCAppSceneLIVE` 和 `TRTCAppSceneVoiceChatRoom` 两个可选项，分别是视频直播和语音直播，该模式适合十万人以内的在线直播场景，但需要您在接下来介绍的 TRTCParams 参数中指定 **角色(role)** 这个字段，也就是将房间中的用户区分为 **主播(anchor)** 和 **观众(audience)** 两种不同的角色。

参数二：TRTCParams

TRTCParams 由很多的字段构成，但通常您只需要关心如下几个字段的填写：

参数名称	字段含义	补充说明	数据类型	填写示例
SDKAppID	应用 ID	您可以在 实时音视频控制台中找到这个 SDKAppID，如果没有就单击“创建应用”按钮创建一个新的应用。	数字	1400000123
userId	用户 ID	即用户名，只允许包含大小写英文字母（a-z、A-Z）、数字（0-9）及下划线和连词符。注意 TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。	字符串	“denny” 或者 “123321”
userSig	进房鉴权票据	您可以使用 SDKAppID 和 userId 计算出 userSig，计算方法请参见 如何计算及使用 UserSig 。	字符串	eJyrVareCeYrSy1SsII...
roomId	房间号	数字类型的房间号。注意如果您想使用字符串类型的房间号，请使用 **strRoomId** 字段，而不要使用 roomId 字段，因为 strRoomId 和 roomId 不可以混用。	数字	29834
strRoomId	房间号	字符串类型的房间号。注意 strRoomId 和 roomId 不可以混用，“123” 和 123 在 TRTC 后台服务看来并不是同一个房间。	字符串	"29834"
role	角色	分为“主播”和“观众”两种角色，只有当 TRTCAppScene 被指定为	枚举值	TRTCRoleAnchor 或 TRTCRoleAudience

		`TRTCAppSceneLIVE` 或 `TRTCAppSceneVoiceChatRoom` 这两种直播场景时才需要指定该字段。		

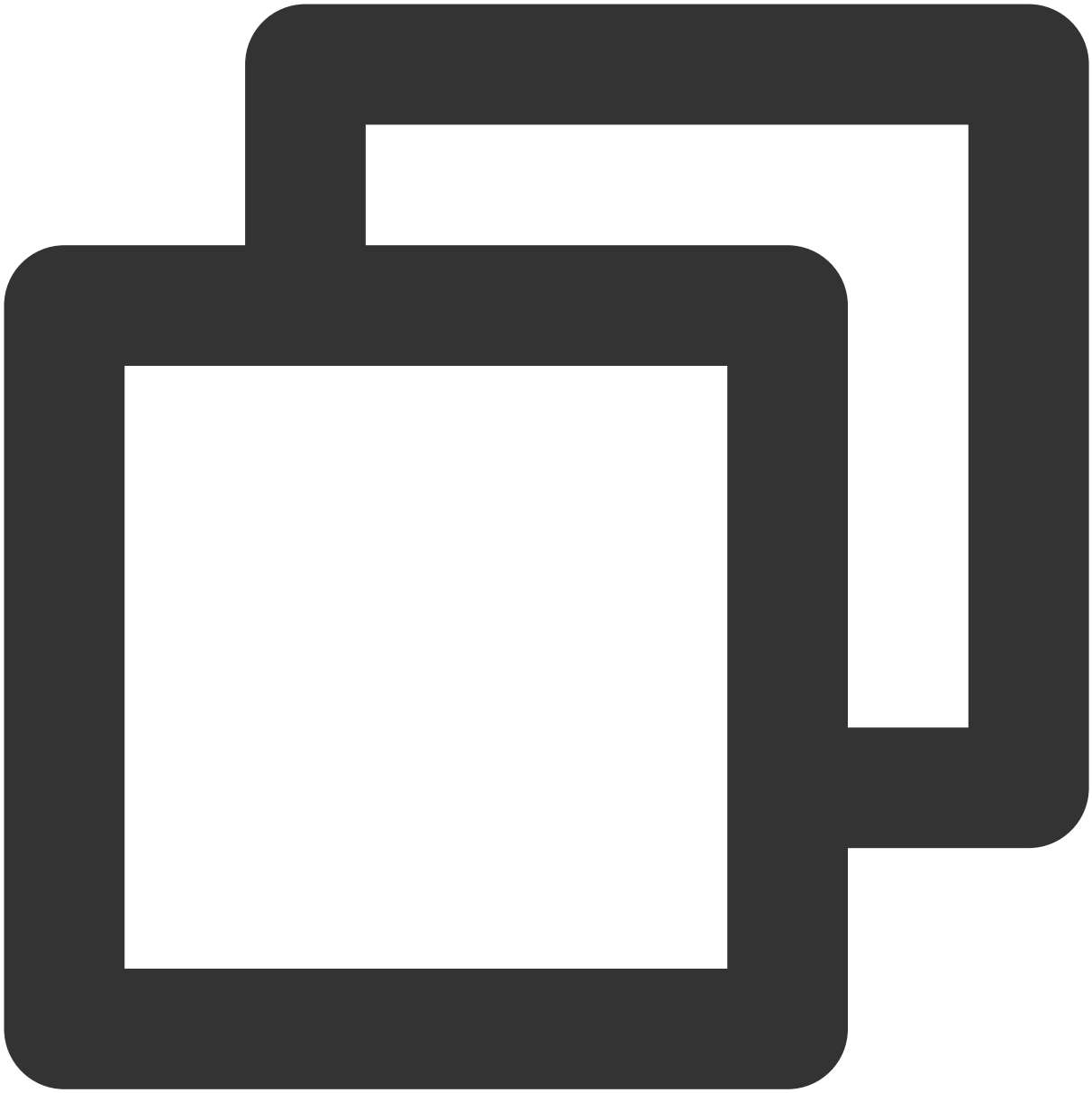
注意：

TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。

每个端在应用场景 appScene 上必须要进行统一，否则会出现一些不可预料的问题。

步骤5：进入房间(enterRoom)

在准备好 步骤4 中两个参数（TRTCAppScene 和 TRTCParams）后，就可以调用 enterRoom 接口函数进入房间了。



```
import { TRTCParams, TRTCRoleType, TRTCAppScene } from 'trtc-electron-sdk';

const param = new TRTCParams();
param.sdkAppId = 1400000123;
param.userId = "denny";
param.roomId = 123321;
param.userSig = "xxx";
param.role = TRTCRoleType.TRTCRoleAnchor;

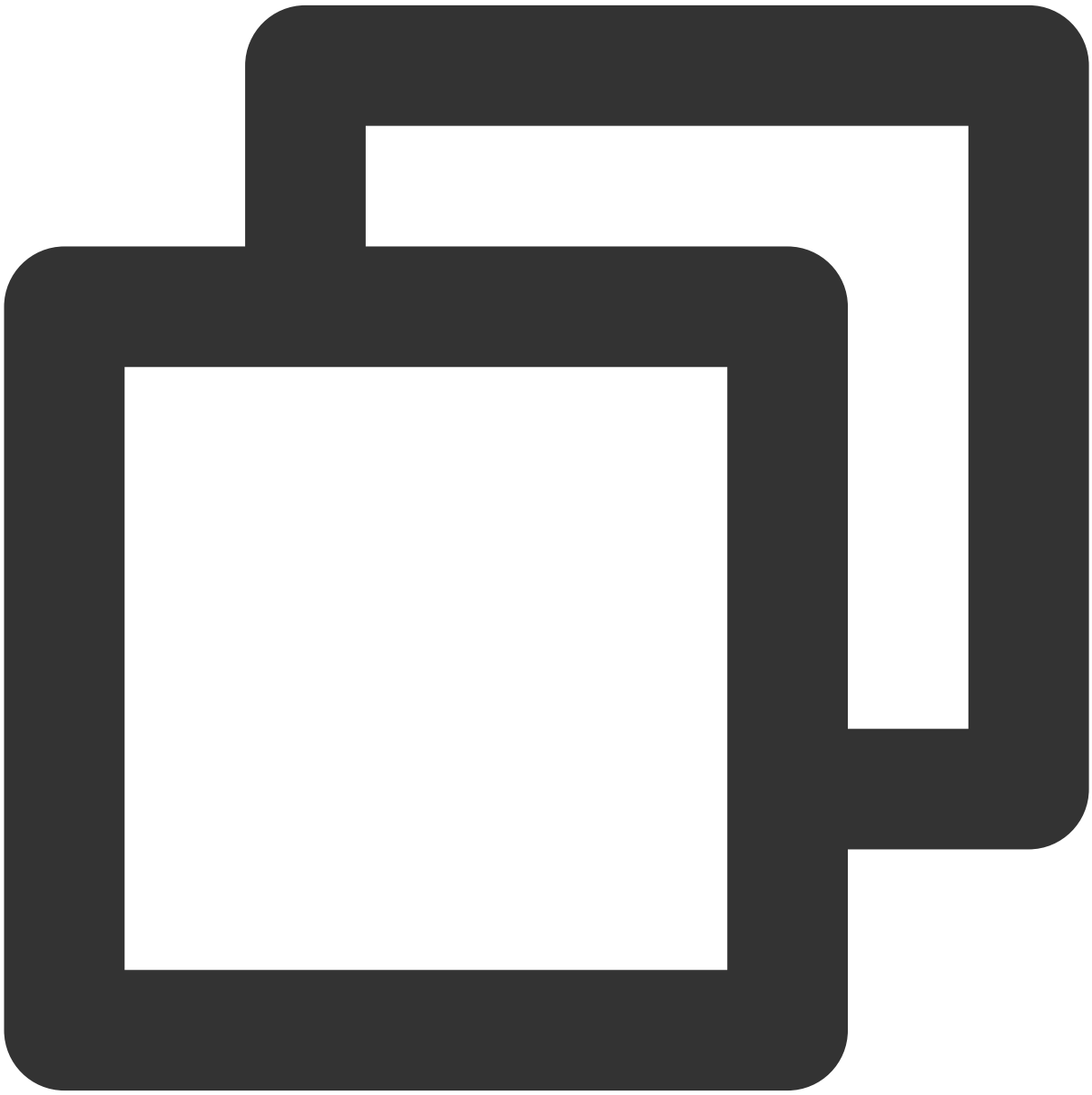
// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
rtcCloud.enterRoom(param, TRTCAppScene.TRTCAppSceneLIVE);
```

版权所有：腾讯云计算（北京）有限责任公司

事件回调：

如果进入房间成功，SDK 会回调 onEnterRoom(result) 事件，其中 result 会是一个大于 0 的数值，代表加入房间所消耗的时间，单位为毫秒（ms）。

如果进入房间失败，SDK 同样会回调 onEnterRoom(result) 事件，但参数 `result` 会是一个负数，其数值为进房失败的错误码。



```
function onEnterRoom(result) {
  // onEnterRoom 参见 https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electr
  if (result > 0) {
    console.log('Enter room succeed');
```

```
  } else {
    // 参见进房错误码 https://www.tencentcloud.com/document/product/647/35124
    console.log('Enter room failed');
  }

}

rtcCloud.on('onEnterRoom', onEnterRoom);
```

Flutter

最近更新时间：2024-02-02 18:43:03

本文档主要介绍如何进入 TRTC 房间中，只有在进入音视频房间后，用户才能订阅房间中其他用户的音视频流，或者向房间中的其他用户发布自己的音视频流。

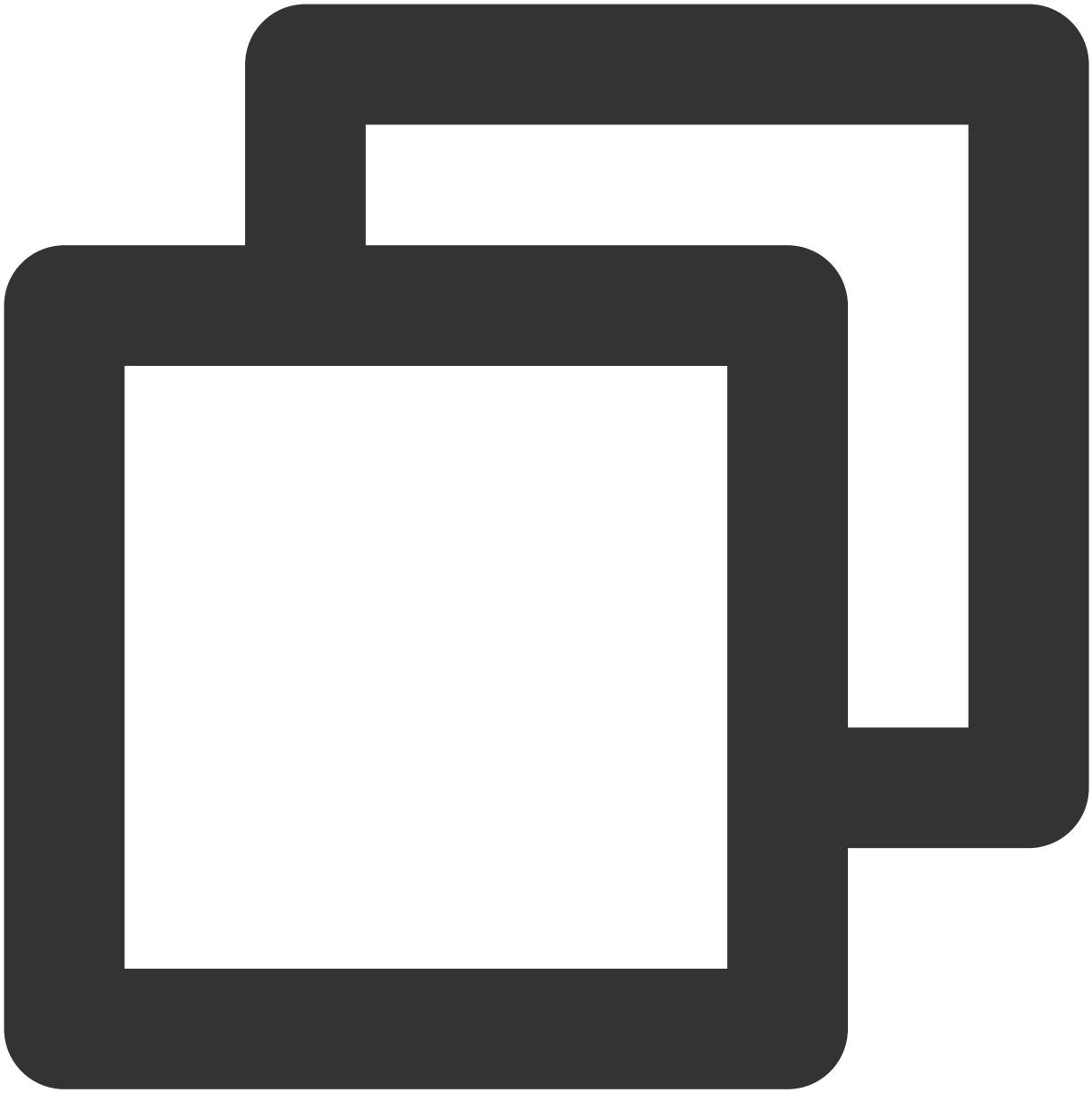
调用指引

步骤1：导入 SDK 并设置 App 权限

请参考文档 导入 SDK 到项目中 完成 SDK 的导入工作。

步骤2：创建 SDK 实例并设置事件监听器

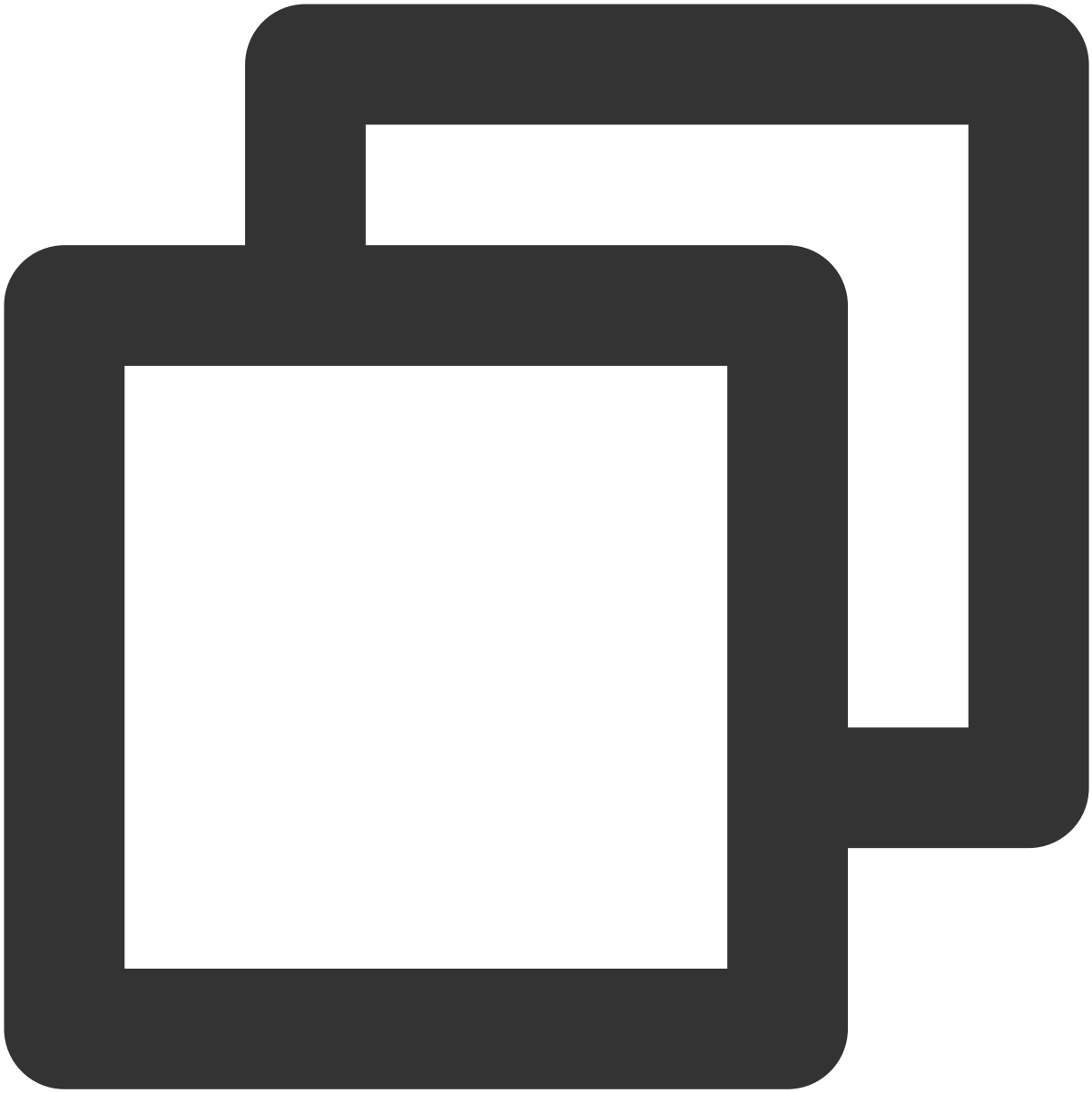
调用各平台的初始化接口创建 TRTC 的对象实例。



```
// 创建 TRTCCloud 单例
trtcCloud = (await TRTCCloud.sharedInstance())!;
// 注册 TRTC 事件回调
trtcCloud.registerListener(onRtcListener);
```

步骤3：监听 SDK 的事件

通过设置事件回调接口，您可以监听 SDK 在运行期间所发生的错误信息、警告信息、流量统计信息、网络质量信息以及各种音视频事件。



```
// 我们需要定义一个方法用来处理事件的回调，在方法中根据收到的事件的type来进行相应的处理。
// 以onError为例
trtcCloud.registerListener(onRtcListener);

onRtcListener(type, param) async {
  if (type == TRTCCloudListener.onError) {
    if (param['errCode'] == -1308) {
      MeetingTool.toast('Failed to start screen recording', context);
    } else {
      showErrordDialog(param['errMsg']);
    }
```

```
    }
}
```

步骤4：准备进房参数 TRTCParams

在调用 enterRoom 接口时需要填写两个关键参数，即 `TRTCParams` 和 应用场景，接下来进行详细介绍：

参数一：scene

该参数用于指定您的应用场景，即**视频通话、互动视频直播、音频通话**还是**互动音频直播**：

场景类型	场景介绍
TRTC_APP_SCENE_VIDEOCALL	在视频通话场景中，支持 720p 和 1080p 高清图像质量。单个房间可容纳最多300名同时在线用户，最多50人同时发言。
TRTC_APP_SCENE_LIVE	在互动视频直播场景中，麦克风可以顺畅地打开/关闭，无需等待切换，主播延迟低至300毫秒以下。支持数十万并发观众用户的直播，播放延迟降至1000毫秒。 **注意**：在此场景下，您必须使用 TRTCParams 中的 role 字段指定当前用户的角色。
TRTC_APP_SCENE_AUDIOCALL	在音频通话场景中，支持48 kHz双声道音频通话。单个房间可容纳最多300名同时在线用户，最多50人同时发言。
TRTC_APP_SCENE_VOICE_CHATROOM	在互动音频直播场景中，麦克风可以顺畅地打开/关闭，无需等待切换，主播延迟低至300毫秒以下。支持数十万并发观众用户的直播，播放延迟降至1000毫秒。 **注意**：在此场景下，您必须使用 TRTCParams 中的 role 字段指定当前用户的角色。

参数二：TRTCParams

TRTCParams 由很多的字段构成，但通常您只需要关心如下几个字段的填写：

参数名称	字段含义	补充说明	数据类型	填写示例
sdkAppId	应用 ID	您可以在 实时音视频控制台 中找到这个 SDKAppID，如果没有就单击“创建应用”按钮创建一个新的应用。	数字	1400000123
userId	用户 ID	即用户名，只允许包含大小写英文字母（a-z、A-Z）、数字（0-9）及下划线和连词符。注意 TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。	字符串	“denny” 或者 “123321”
userSig	进房鉴	您可以使用 SDKAppID 和 userId 计算出	字符串	eJyrVareCeYrSy1SsII...

	权票据	userSig，计算方法请参见 如何计算及使用 UserSig 。		
roomId	房间号	数字类型的房间号。注意如果您想使用字符串类型的房间号，请使用 **strRoomId** 字段，而不要使用 roomId 字段，因为 strRoomId 和 roomId 不可以混用。	数字	29834
strRoomId	房间号	字符串类型的房间号。注意 strRoomId 和 roomId 不可以混用，“123” 和 123 在 TRTC 后台服务看来并不是同一个房间。	字符串	"29834"
role	角色	分为“主播”和“观众”两种角色，只有当 TRTCAppScene 被指定为 `TRTCAppSceneLIVE` 或 `TRTCAppSceneVoiceChatRoom` 这两种直播场景时才需要指定该字段。	枚举值	TRTCRoleAnchor 或 TRTCRoleAudience

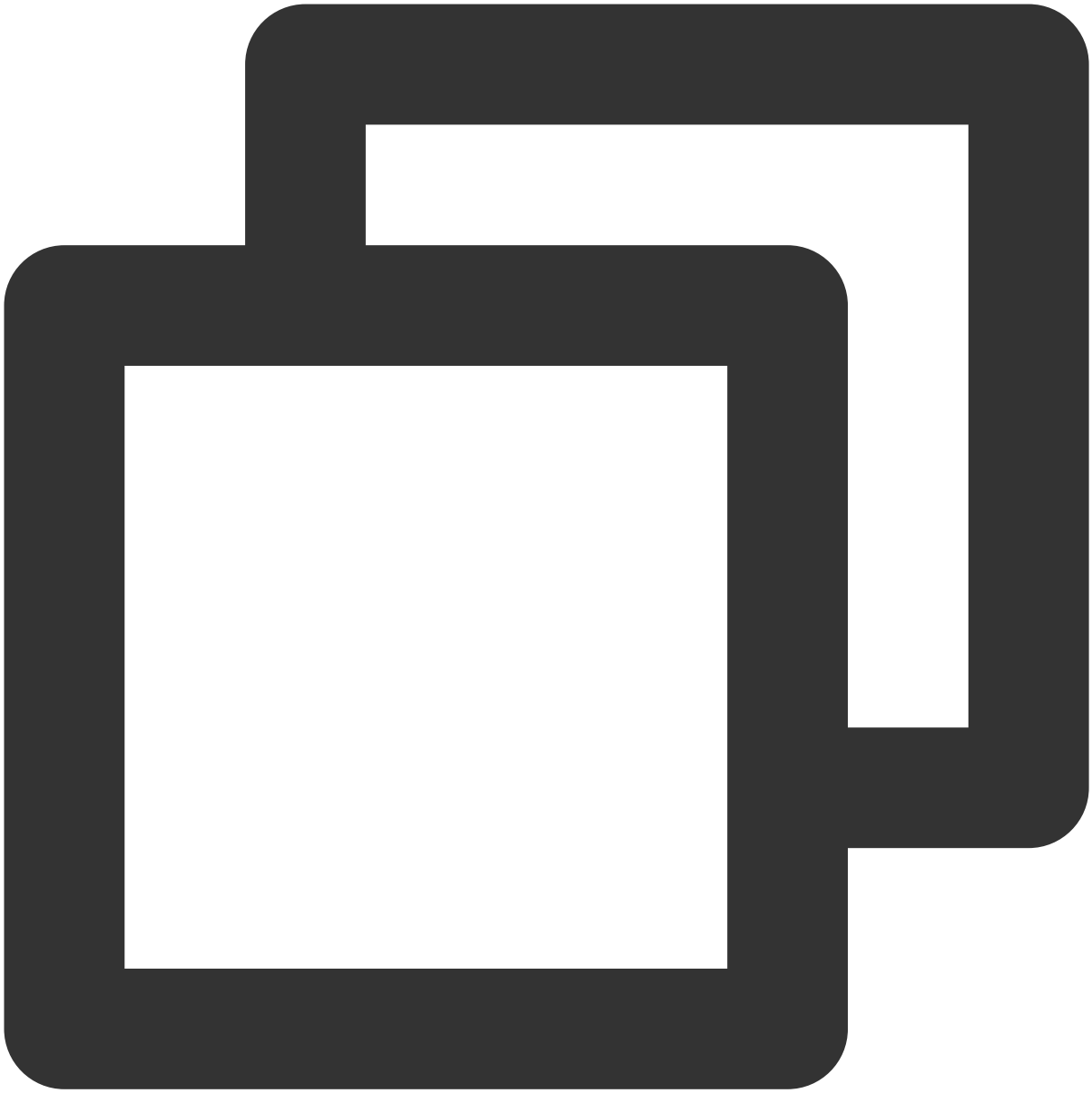
注意：

TRTC 不支持同一个 userId 在两台不同的设备上同时进入房间，否则会相互干扰。

每个端在应用场景 appScene 上必须要进行统一，否则会出现一些不可预料的问题。

步骤5：进入房间（enterRoom）

在准备好步骤4中两个参数（应用场景 和 TRTCParams）后，就可以调用 enterRoom 接口函数进入房间了。



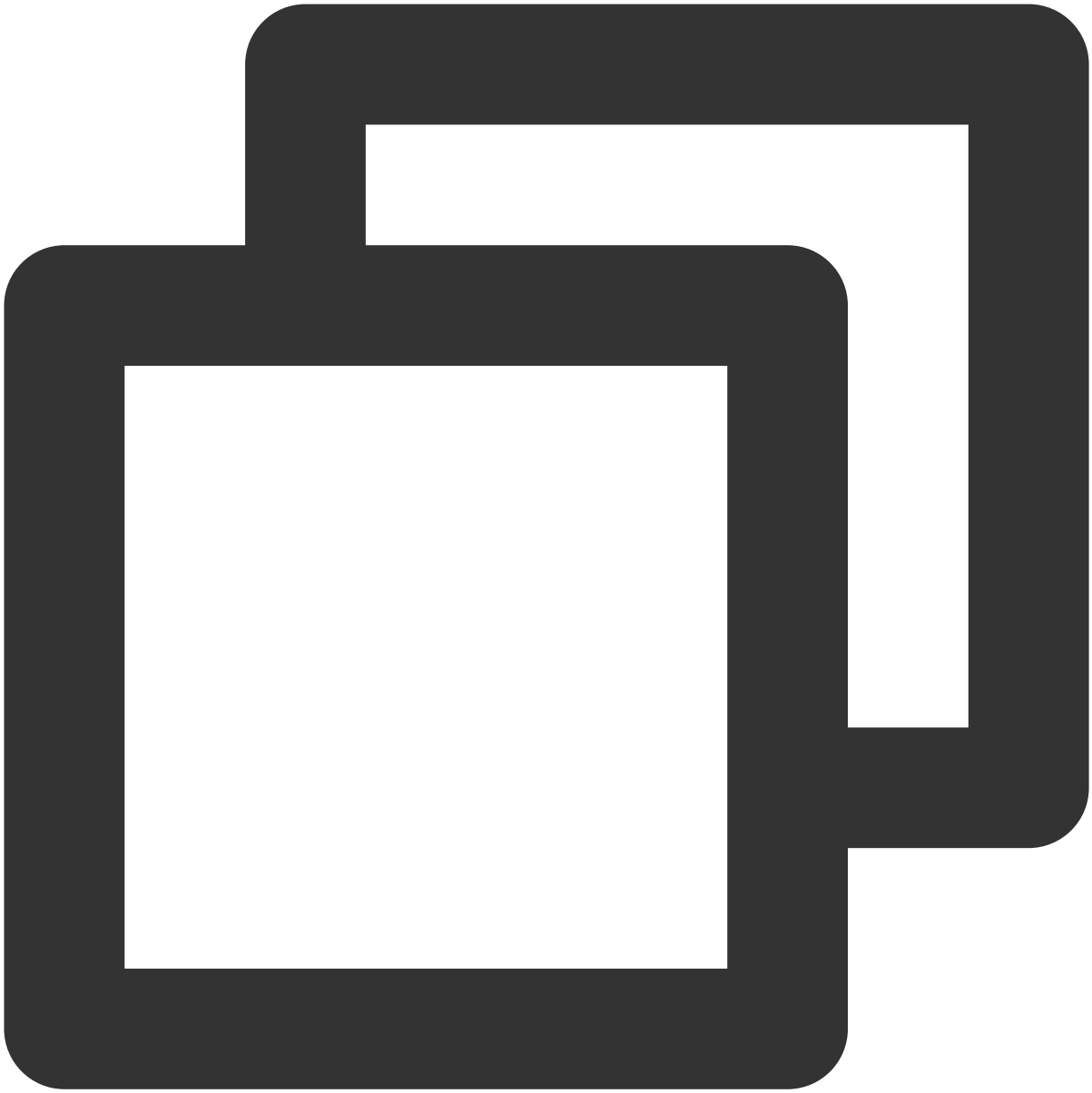
```
enterRoom() async {
  try {
    userInfo['userSig'] =
        await GenerateTestUserSig.genTestSig(userInfo['userId']);
    meetModel.setUserInfo(userInfo);
  } catch (err) {
    userInfo['userSig'] = '';
    print(err);
  }
  // 如果您的场景为“互动视频直播”，请将场景设置为 TRTC_APP_SCENE_LIVE，并在TRTCParams中为rol
  await trtcCloud.enterRoom(
```

```
        TRTCParams(
            sdkAppId: GenerateTestUserSig.sdkAppId,
            userId: userInfo['userId'],
            userSig: userInfo['userSig'],
            role: TRTCCloudDef.TRTCRoleAnchor,
            roomId: meetId!),
        TRTCCloudDef.TRTC_APP_SCENE_LIVE);
  }
```

事件回调

如果进入房间成功，SDK 会回调 onEnterRoom(result) 事件，其中 result 会是一个大于 0 的数值，代表加入房间所消耗的时间，单位为毫秒（ms）。

如果进入房间失败，SDK 同样会回调 onEnterRoom(result) 事件，但参数 `result` 会是一个负数，其数值为进房失败的错误码。

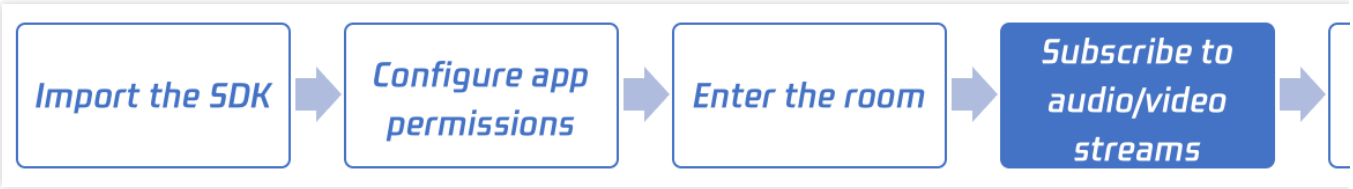


```
//监听 SDK 的 onEnterRoom 事件并获知是否成功进入房间
onRtcListener(type, param) async {
  if (type == TRTCCloudListener.onEnterRoom) {
    if (param > 0) {
      MeetingTool.toast('Enter room success', context);
    }
  }
}
```

03. 订阅音视频流 Android&iOS&Windows&Mac

最近更新时间：2024-06-07 11:20:49

本文档主要介绍如何订阅房间中其他用户的音视频流，也就是如何播放其他用户的音频和视频。为了方便起见，我们在接下来的文档中，会将“房间中的其他用户”统称为“远端用户”。



调用指引

步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。

步骤2：设定订阅模式（非必须）

您可以通过调用 TRTCCloud 中的 **setDefaultStreamRecvMode** 接口设置订阅模式，TRTC 提供了两种订阅模式：
自动订阅：SDK 会自动播放远端用户的声音，无需您进行额外操作，这是 SDK 的默认行为。
手动订阅：SDK 不会自动拉取和播放远端用户的声音，需要您手动调用 **muteRemoteAudio(userId, false)** 来触发声音的播放。

注意：

需要您注意的是，如果您不调用 setDefaultStreamRecvMode 也是没有关系的，SDK 的默认行为是自动订阅。但如果您希望设置为手动订阅，请务必注意 setDefaultStreamRecvMode 只有在 enterRoom 之前调用才有效果。

步骤3：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间，只有在成功进入房间之后才能订阅其他用户的音视频流。

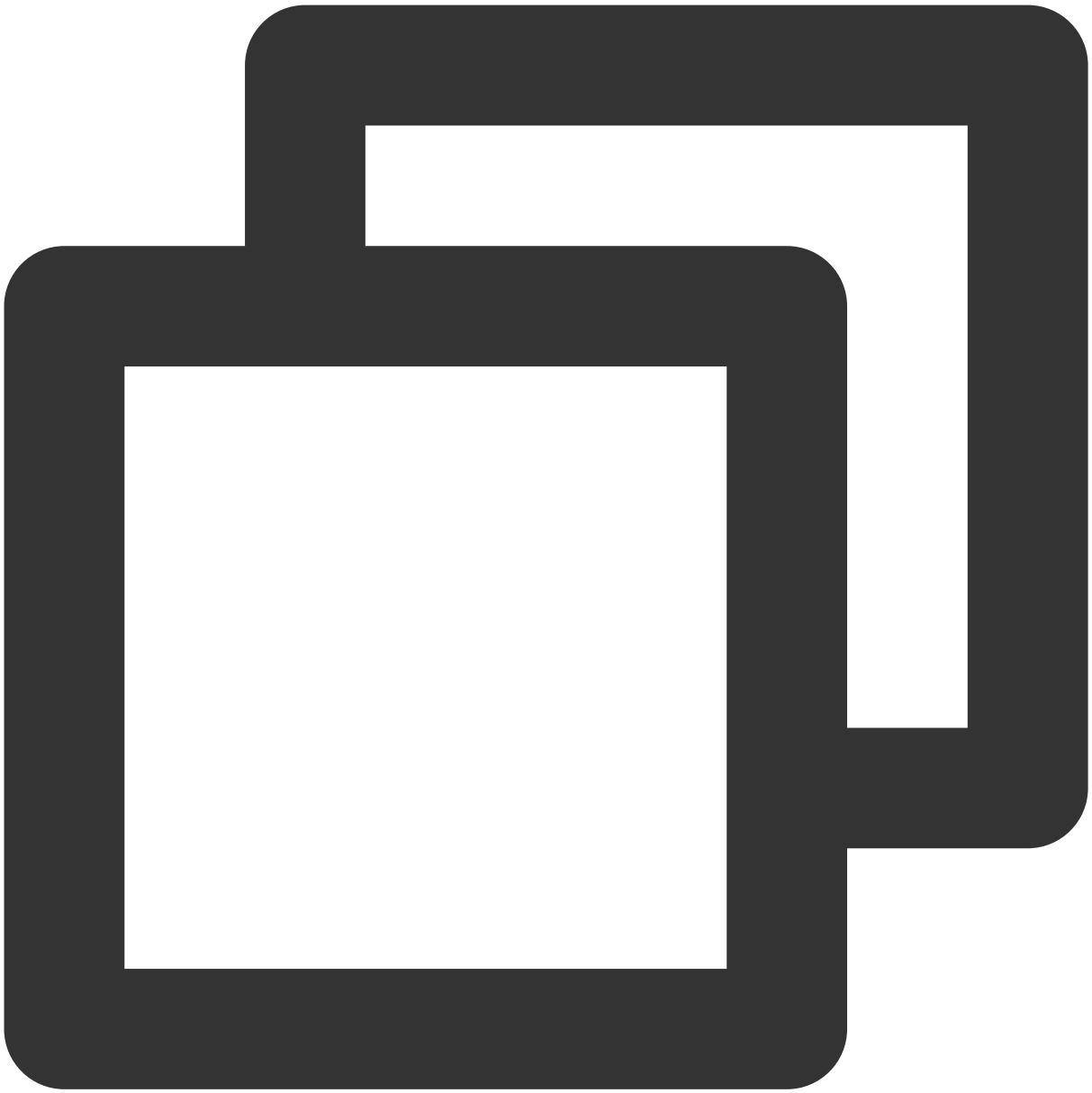
步骤4：音频流的播放

您可以通过调用接口 muteRemoteAudio("denny"，true) 来静音远端用户 denny 的声音，之后也可以通过调用接口 muteRemoteAudio("denny"，false) 来解除对他的静音。

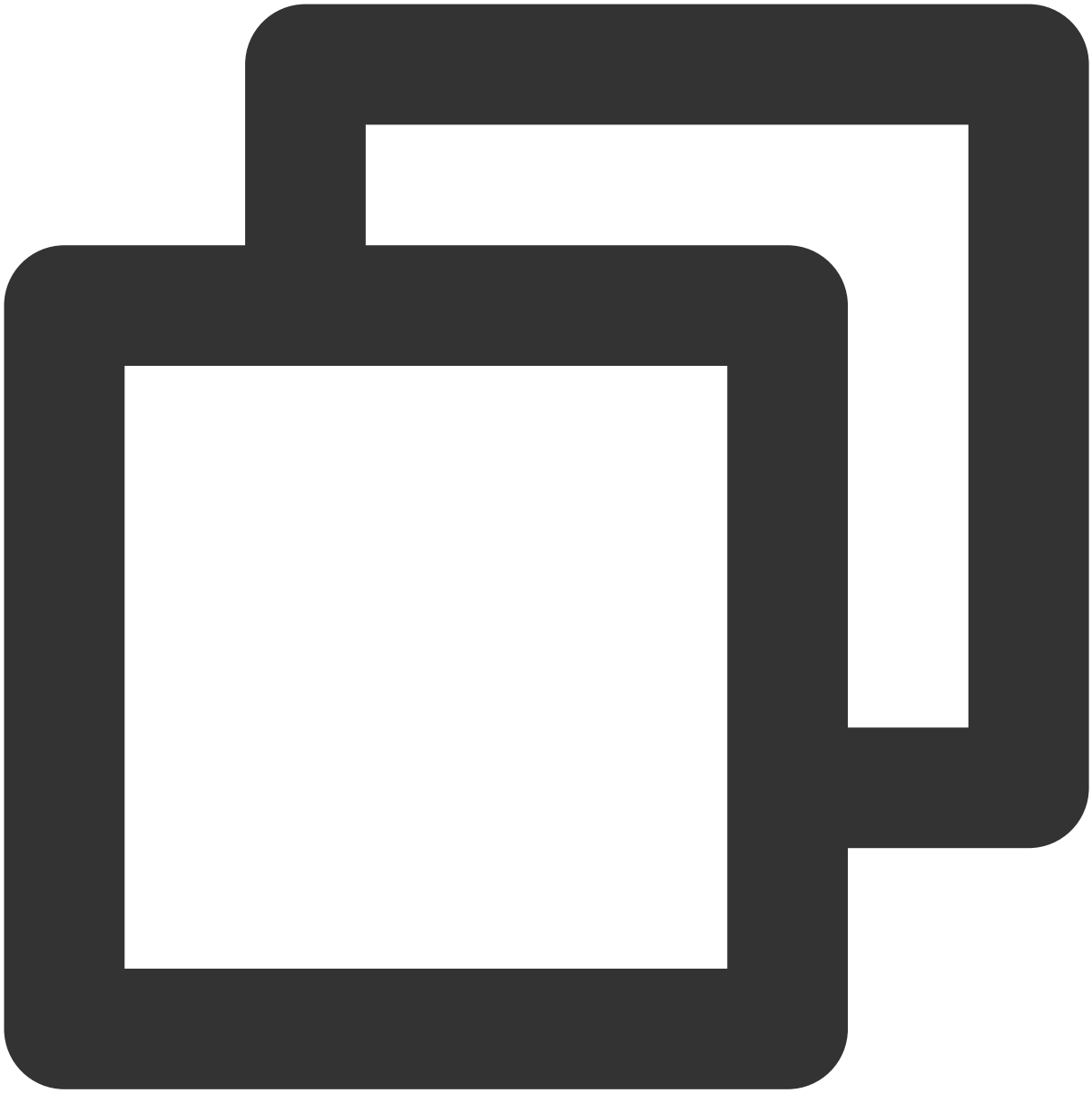
Android

iOS&Mac

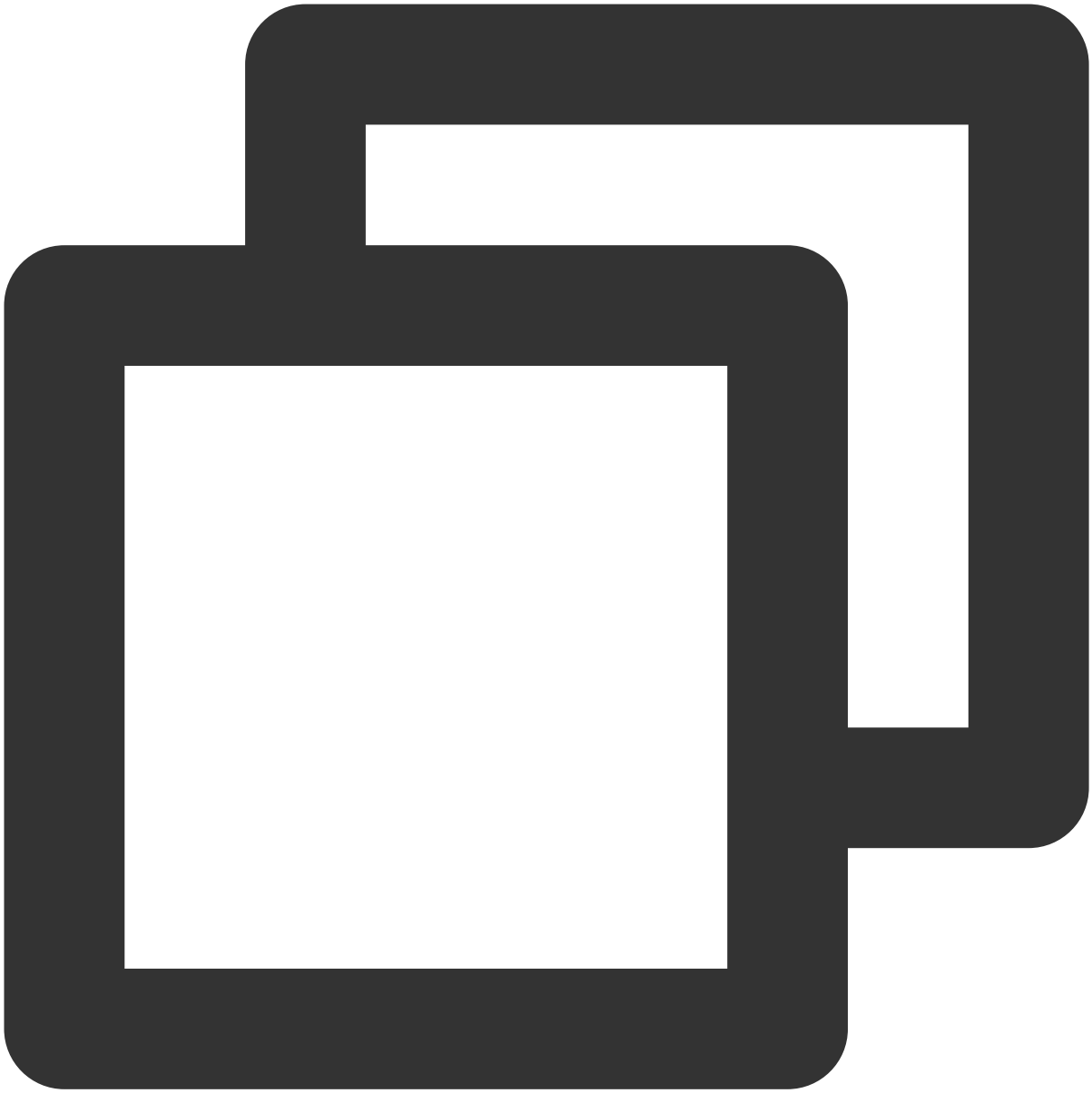
Windows



```
// Mute user with id denny
mCloud.muteRemoteAudio("denny", true);
// Unmute user with id denny
mCloud.muteRemoteAudio("denny", false);
```



```
self.trtcCloud = [TRTCCloud sharedInstance];
// Mute user with id denny
[self.trtcCloud muteRemoteAudio:@"denny" mute:YES];
// Unmute user with id denny
[self.trtcCloud muteRemoteAudio:@"denny" mute:YES];
```



```
// Mute user with id denny
trtc_cloud_->muteRemoteAudio("denny", true);
// Unmute user with id denny
trtc_cloud_->muteRemoteAudio("denny", false);
```

步骤5：视频流的播放

1. 开始和停止播放(startRemoteView + stopRemoteView)

您可以通过调用接口 startRemoteView 来播放远端用户的视频画面，但前提是您需要传递给 SDK 一个 view 对象，用来作为承载该用户的视频画面的渲染控件。

startRemoteView 的第一个参数是远端用户的 userId，第二个参数是远端用户的流类型，第三个参数则是您需要传递的 view 对象。其中第二个参数 streamType(流类型) 有三个可选的值，分别是：

TRTCVideoStreamTypeBig : 用户的主路画面，一般用来传输用户摄像头的画面。

TRTCVideoStreamTypeSub : 用户的辅路画面，一般用来传输用户屏幕分享的画面。

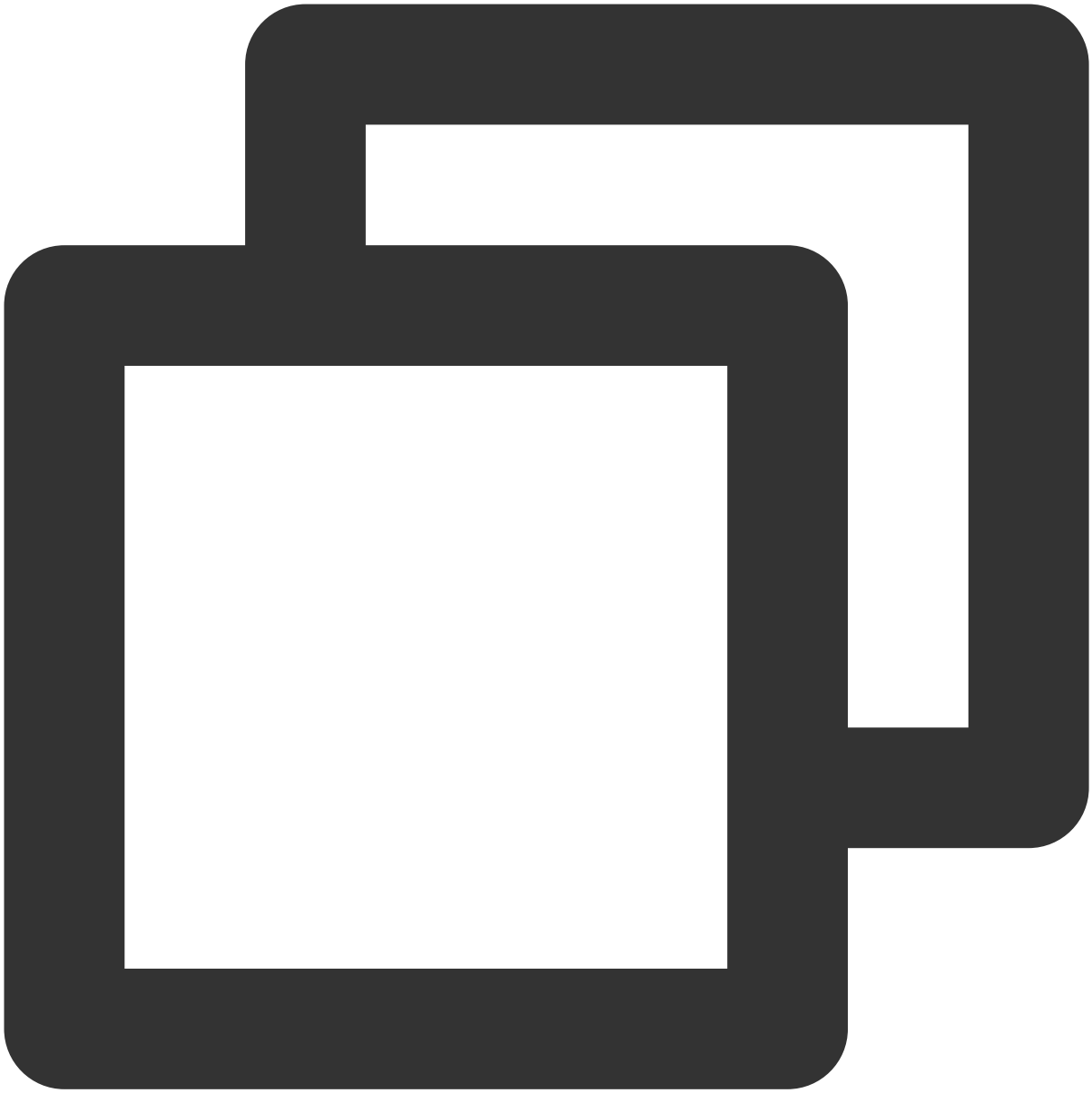
TRTCVideoStreamTypeSmall : 用户的低清小画面，这是相对于主路画面而言的，只有当远端用户开启了“双路编码（enableEncSmallVideoStream）”之后，我们才能播放他/她的低清画面。并且同时只能在主路画面和低清小画面中二选一。

通过调用 stopRemoteView 接口，您可以停止播放某一个远端用户的视频，也可以通过 stopAllRemoteView 接口停止播放所有远端用户的视频。

Android

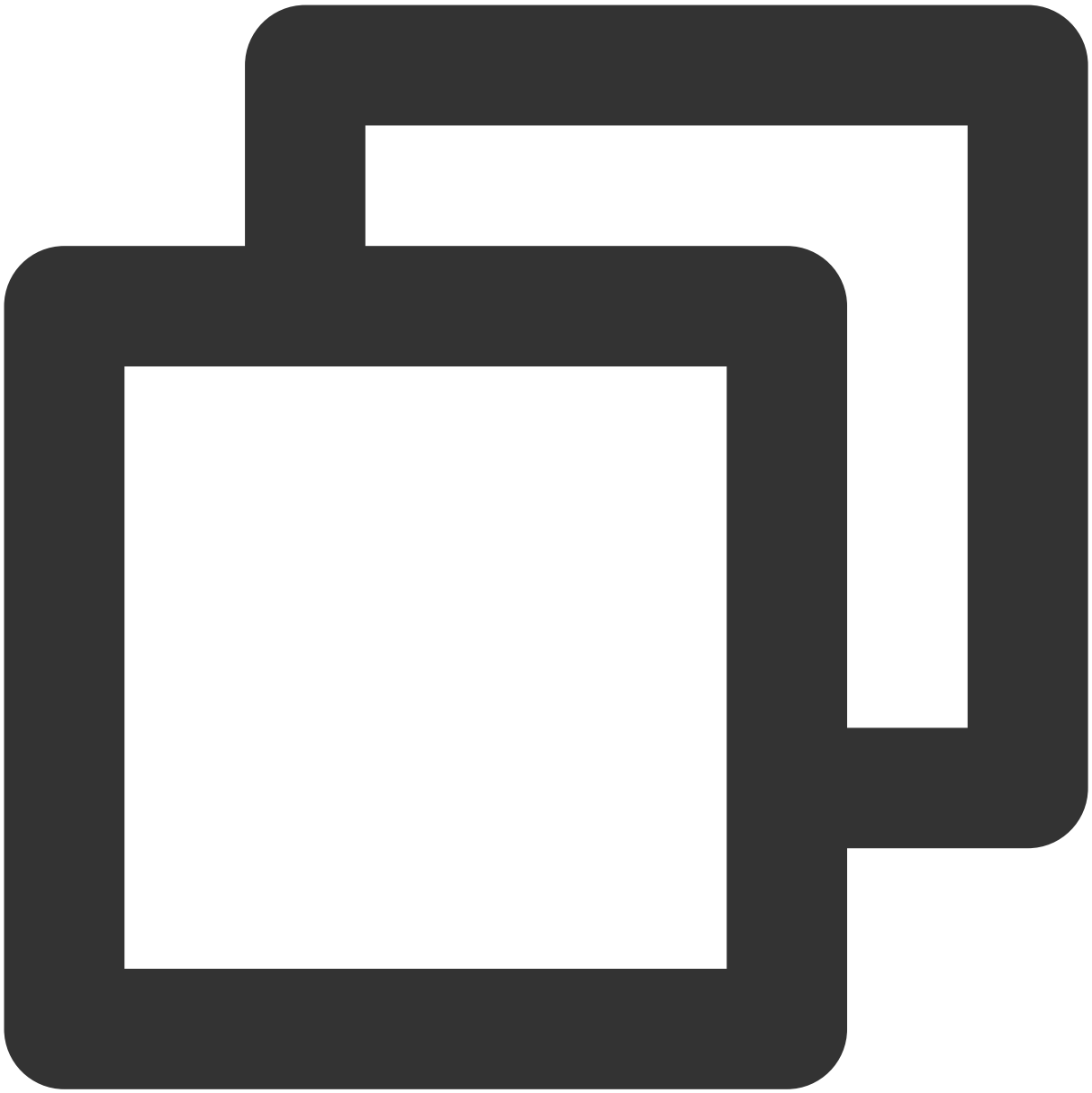
iOS&Mac

Windows



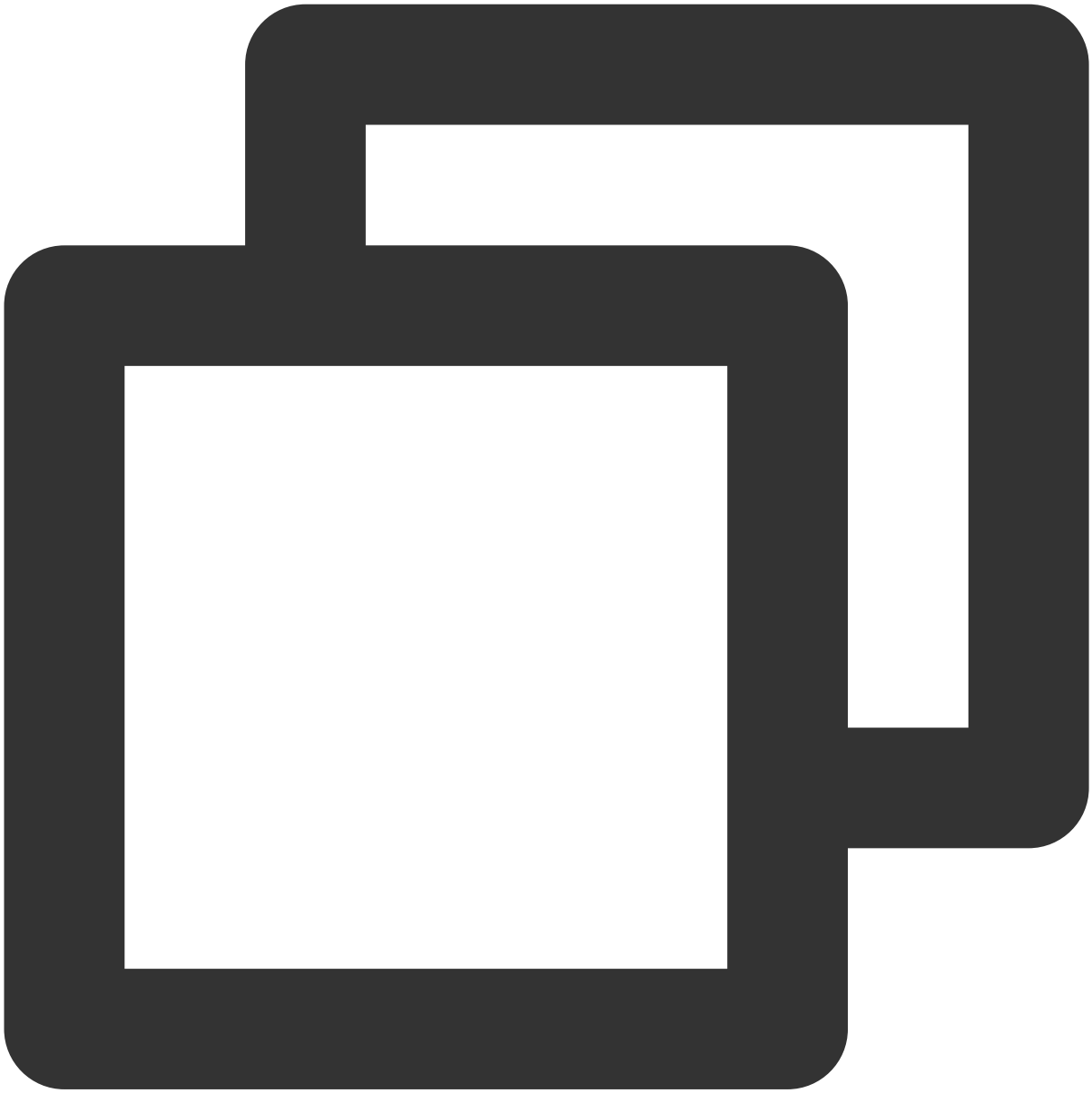
```
// 播放 denny 的摄像头画面（我们称之为“主路画面”）
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraView)
// 播放 denny 的屏幕分享画面（我们称之为“辅路画面”）
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SUB, screenView)
// 播放 denny 的低分辨率画面（主路画面和低清画面只能二选一）
mCloud.startRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SMALL, cameraVie
// 停止播放 denny 的摄像头画面
mCloud.stopRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraView)
// 停止播放所有视频画面
mCloud.stopAllRemoteView();
```

版权所有：腾讯云计算（北京）有限责任公司



```
self.trtcCloud = [TRTCCloud sharedInstance];
// 播放 denny 的摄像头画面（我们称之为“主路画面”）
[self.trtcCloud startRemoteView:@"denny" streamType:TRTCVideoStreamTypeBig view:cam
// 播放 denny 的屏幕分享画面（我们称之为“辅路画面”）
[self.trtcCloud startRemoteView:@"denny" streamType:TRTCVideoStreamTypeSub view:scr
// 播放 denny 的低分辨率画面（主路画面和低清画面只能二选一）
[self.trtcCloud startRemoteView:@"denny" streamType:TRTCVideoStreamTypeSmall view:c
// 停止播放 denny 的摄像头画面
[self.trtcCloud stopRemoteView:@"denny" streamType:TRTCVideoStreamTypeBig view:came
// 停止播放所有视频画面
```

```
[self.trtcCloud stopAllRemoteView];
```



```
// 播放 denny 的摄像头画面（我们称之为“主路画面”）
trtc_cloud_->startRemoteView("denny", liteav::TRTCVideoStreamTypeBig, (liteav::TXVi
// 播放 denny 的屏幕分享画面（我们称之为“辅路画面”）
trtc_cloud_->startRemoteView("denny", liteav::TRTCVideoStreamTypeSub, (liteav::TXVi
// 播放 denny 的低分辨率画面（主路画面和低清画面只能二选一）
trtc_cloud_->startRemoteView("denny", liteav::TRTCVideoStreamTypeSmall, (liteav::TX
// 停止播放 denny 的摄像头画面
trtc_cloud_->stopRemoteView("denny", liteav::TRTCVideoStreamTypeBig);
```

```
// 停止播放所有视频画面
trtc_cloud_->stopAllRemoteView();
```

2. 设置播放参数（updateRemoteView + setRemoteRenderParams）

通过调用 updateRemoteView 接口，您可以在播放中变更 view 对象，这在切换视频渲染控件时非常有用。

通过 setRemoteRenderParams 您可以设置画面的填充模式、旋转角度和镜像模式。

填充模式：分为填充或者适应，两种模式下画面都能保持原始的宽高比例，区别在于有无黑边。

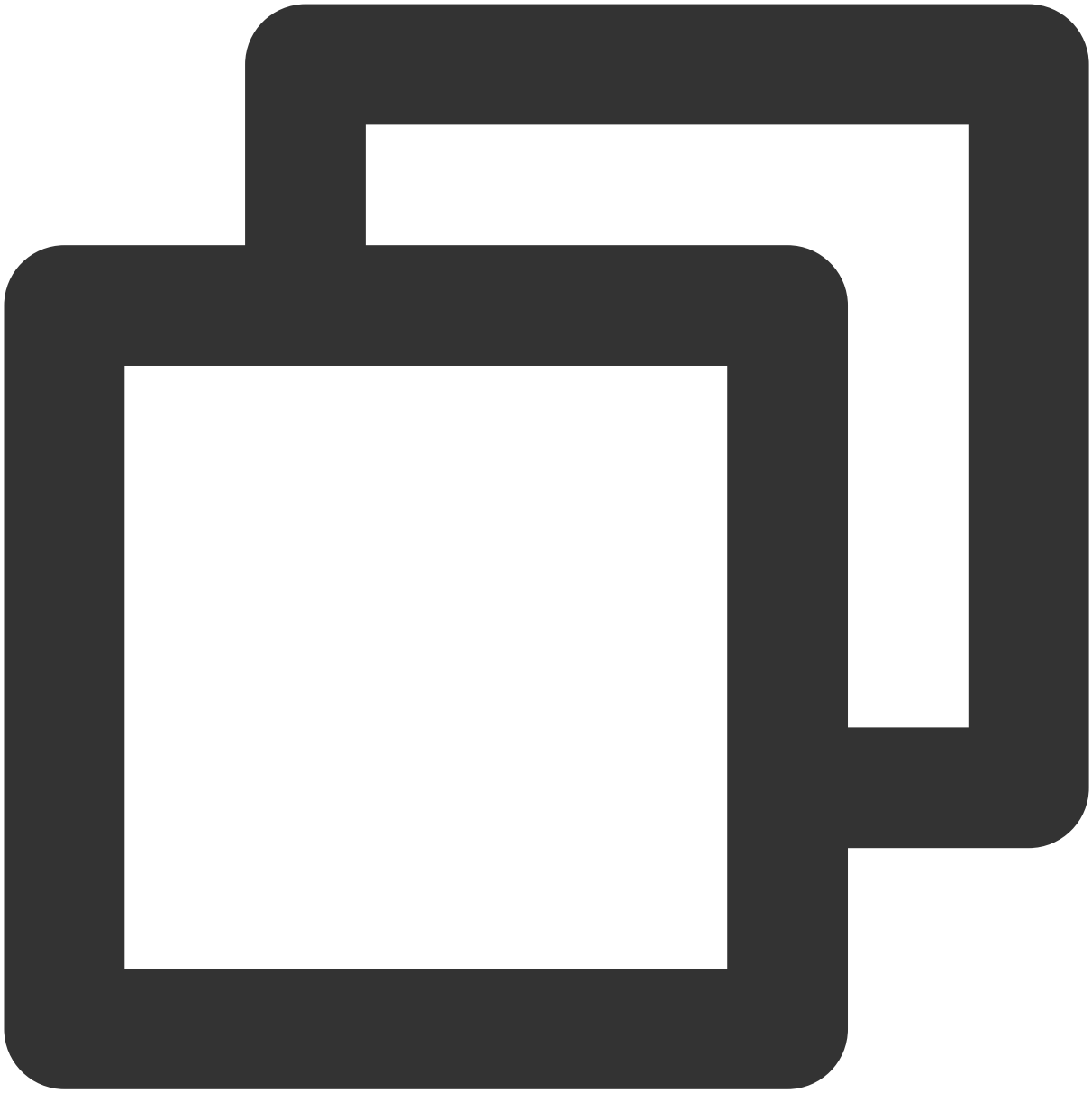
旋转角度：可以设置 0度、90度、180度 和 270度等四个旋转角度。

镜像模式：即画面的左右镜像模式。

Android

iOS&Mac

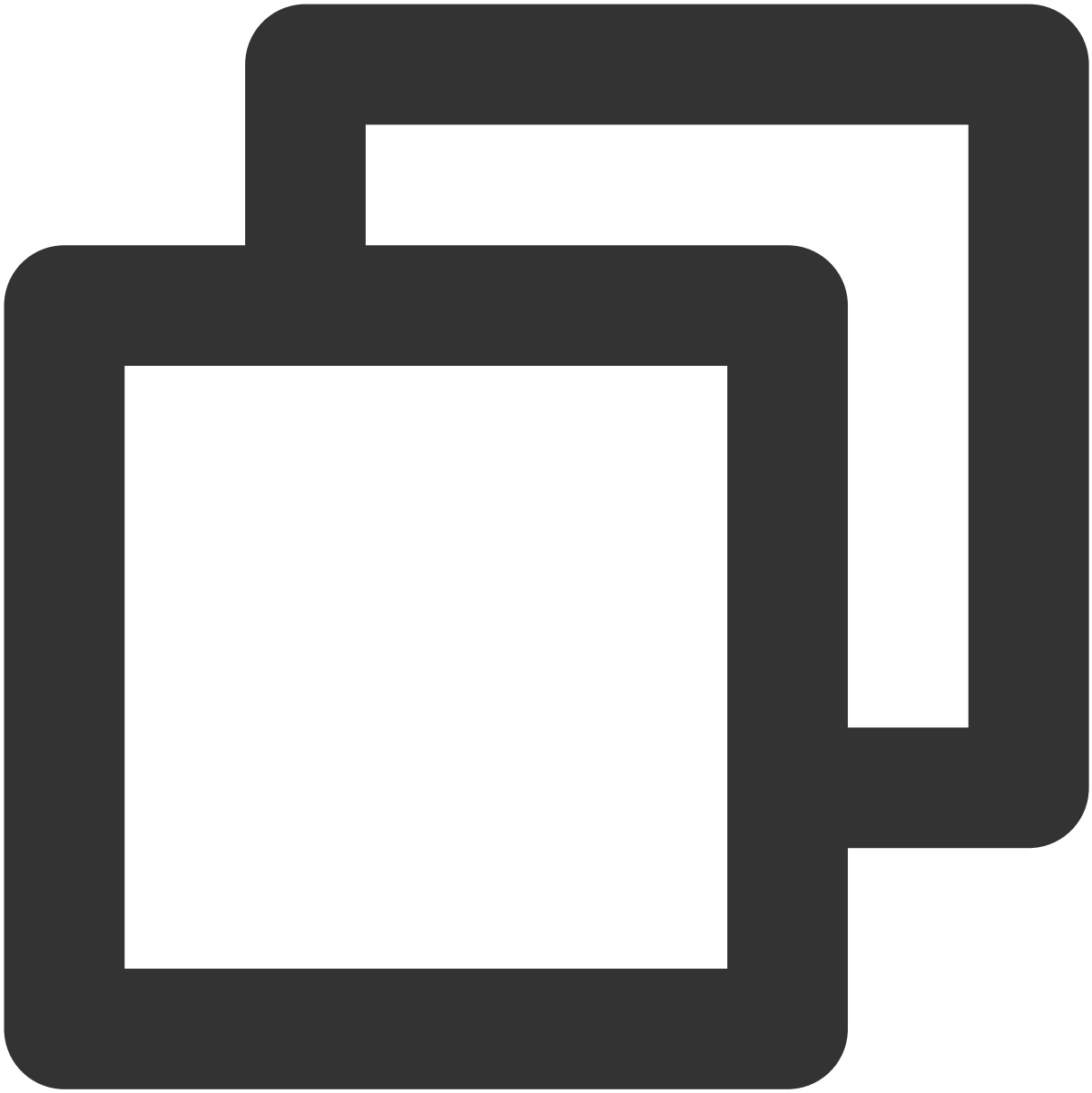
Windows



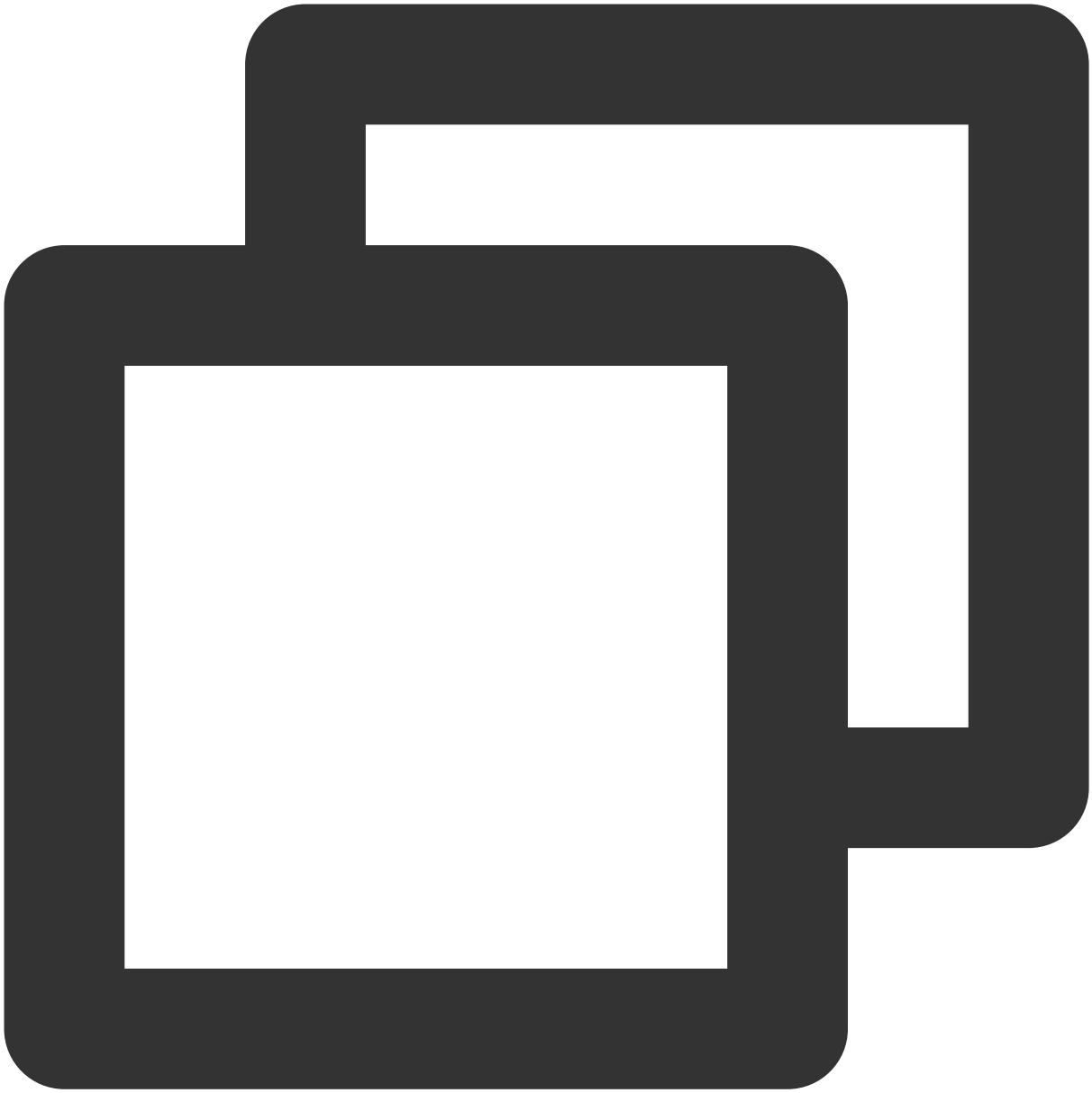
```
// 将 denny 的主路画面切换到一个悬浮的小窗口中（假如该迷你小窗口为 miniFloatingView）
mCloud.updateRemoteView("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, miniFloati

// 将远端用户 denny 的主路画面设置为填充模式，并开启左右镜像模式
TRTCCloudDef.TRTCRenderParams param = new TRTCCloudDef.TRTCRenderParams();
param.fillMode    = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;
param.mirrorType    = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_DISABLE;
mCloud.setRemoteRenderParams("denny", TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, para
```

版权所有：腾讯云计算（北京）有限责任公司



```
self.trtcCloud = [TRTCCloud sharedInstance];
// 将 denny 的主路画面切换到一个悬浮的小窗口中（假如该迷你小窗口为 miniFloatingView）
[self.trtcCloud updateRemoteView:miniFloatingView streamType:TRTCVideoStreamTypeBig
// 将远端用户 denny 的主路画面设置为填充模式，并开启左右镜像模式
TRTCRenderParams *param = [[TRTCRenderParams alloc] init];
param.fillMode      = TRTCVideoFillMode_Fill;
param.mirrorType    = TRTCVideoMirrorTypeDisable;
[self.trtcCloud setRemoteRenderParams:@"denny" streamType:TRTCVideoStreamTypeBig pa
```



```
// 将 denny 的主路画面切换到另外一个窗口中（假如新窗口句柄为 newView）
trtc_cloud_->updateRemoteView("denny", liteav::TRTCVideoStreamTypeBig, (liteav::TXV

// 将远端用户 denny 的主路画面设置为填充模式，并开启左右镜像模式
liteav::TRTCRenderParams param;
param.fillMode = TRTCVideoFillMode_Fill;
param.mirrorType = TRTCVideoMirrorType_Enable;
trtc_cloud_->setRemoteRenderParams("denny", TRTCVideoStreamTypeBig, param);
```

版权所有：腾讯云计算（北京）有限责任公司

步骤6：感知房间中远端用户的音视频状态

在 步骤4 和步骤5中，您可以控制对远端用户的声音和视频的播放，但是如果没有足够的信息，您并不知道：
当前房间里有哪些用户？
他们是否开启了摄像头和麦克风？
为了解决这个问题，您需要监听来自 SDK 的几个事件回调：

音频状态变化通知（onUserAudioAvailable）

当远端用户开启或关闭麦克风时，您可以通过监听 onUserAudioAvailable(userId，boolean) 来感知到这个状态的变化。

视频状态变化通知（onUserVideoAvailable）

当远端用户开启或者关闭摄像头画面时，您可以通过监听 onUserVideoAvailable(userId，boolean) 来感知到这个状态的变化。
当远端用户开启或关闭屏幕分享画面时，您可以通过监听 onUserSubStreamAvailable(userId，boolean) 来感知到这个状态的变化。

用户进出房间的通知（onRemoteUserEnter/LeaveRoom）

当有远端用户进入当前房间时，您可以通过 onRemoteUserEnterRoom(userId) 来感知到该用户的 userId，当有远端用户离开当前方式时，您可以通过 onRemoteUserLeaveRoom(userId, reason) 来感知到该用户的 userId 以及他/她离开的原因。

注意：

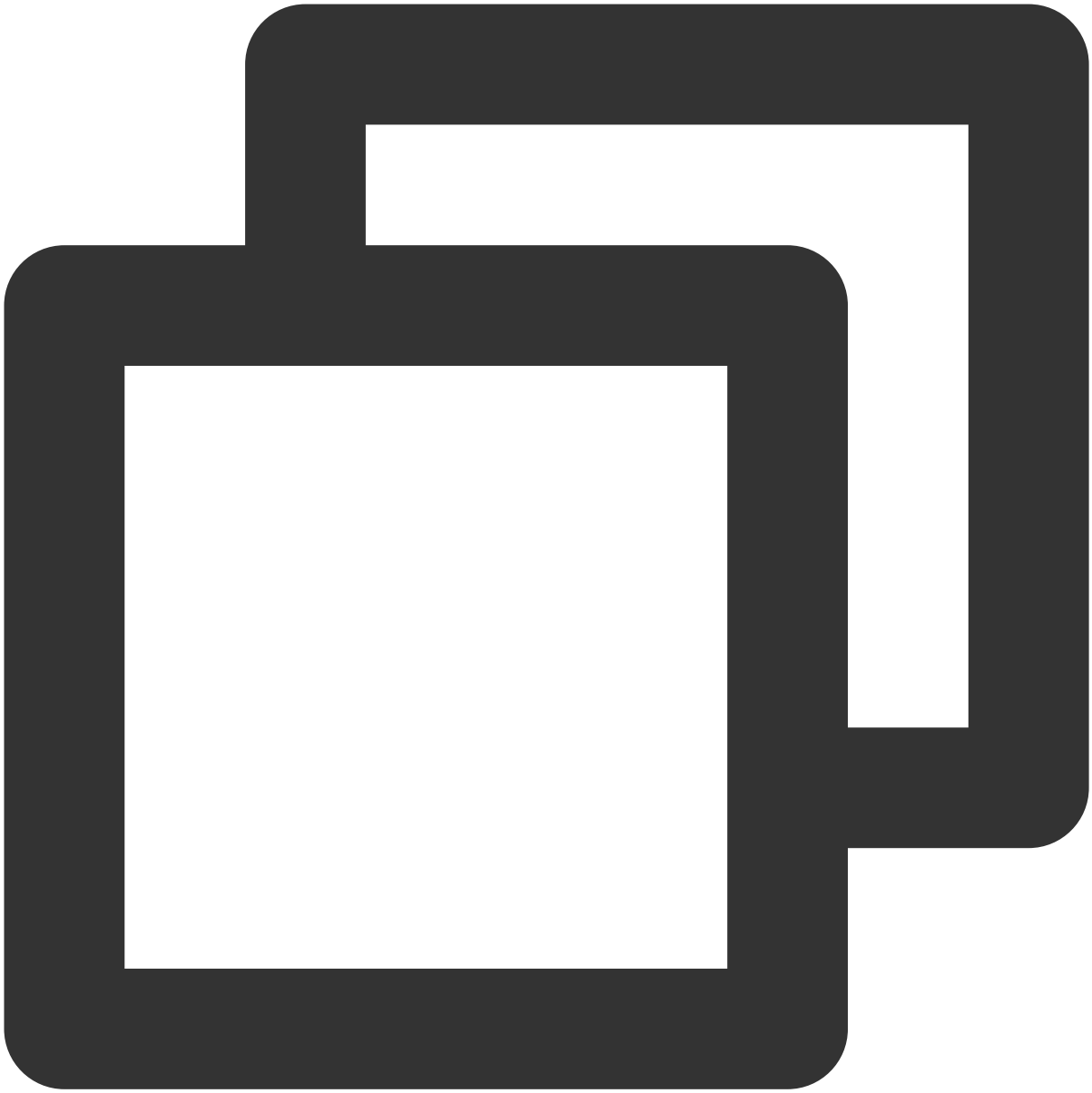
准确地说，onRemoteUserEnter/LeaveRoom 仅能感知角色（role）为主播（anchor）的用户的进出房间的通知，之所以这样设计，是为了避免当一个房间中观众（audience）在线人数较多时，会因为频繁有人进出房间导致房间中的所有用户被其他用户进出房间的“信令风暴“攻击到。

有了这些事件回调，您就可以掌握房间中有哪些用户，以及他们是否开启了摄像头和麦克风，参考如下的示例代码，在这段示例代码中，我们使用了 mCameraUserList、mMicrophoneUserList 以及 mUserList 分别维护了：
房间中的用户（准确的说是主播）是哪些人
其中开启摄像头的用户是哪些人
其中开启麦克风的用户是哪些人

Android

iOS&Mac

Windows

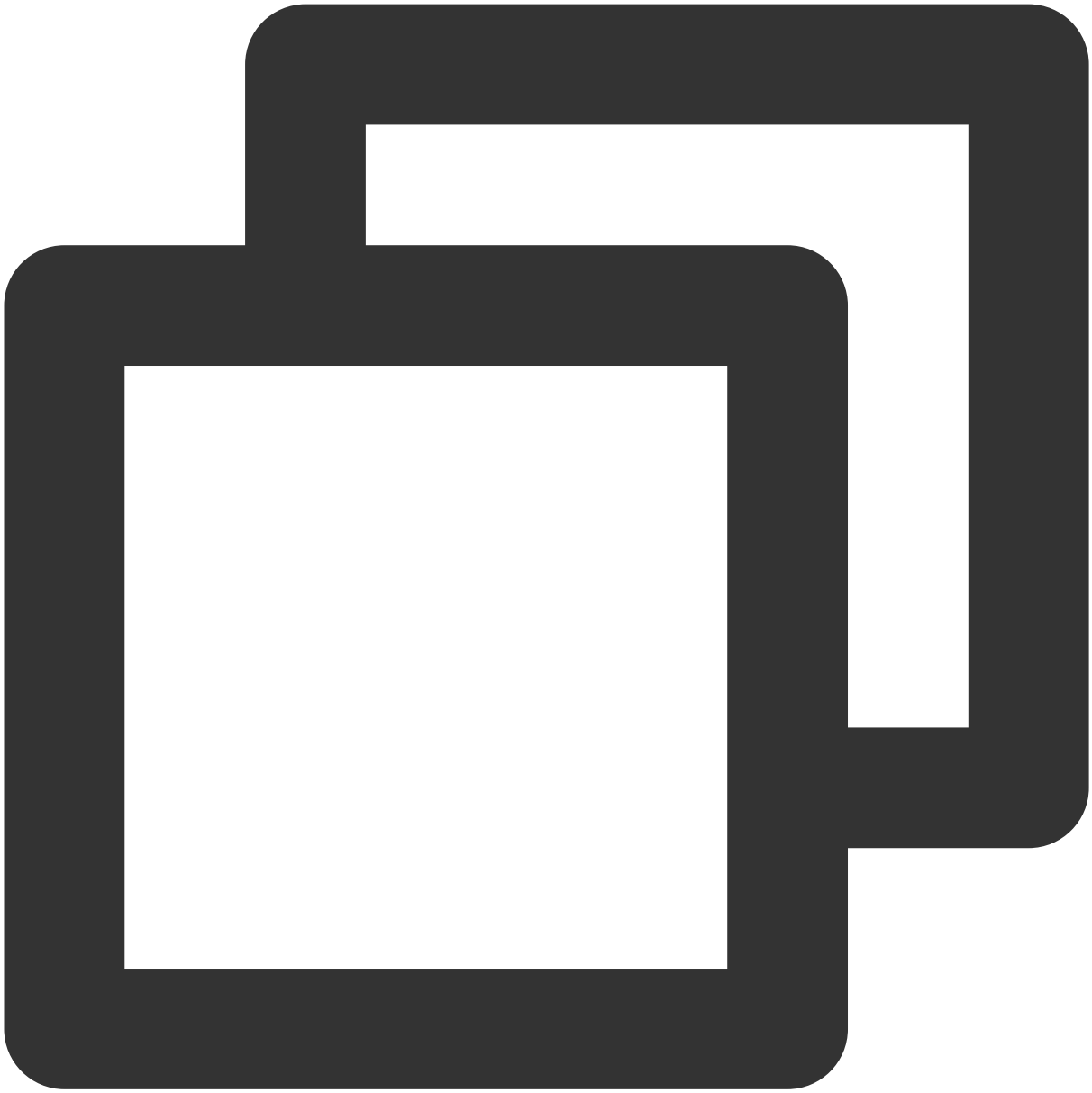


```
// 感知远端用户视频状态的变化，并更新开启了摄像头的用户列表(mCameraUserList)
@Override
public void onUserVideoAvailable(String userId, boolean available) {
    available ? mCameraUserList.add(userId) : mCameraUserList.remove(userId);
}

// 感知远端用户音频状态的变化，并更新开启了麦克风的用户列表(mMicrophoneUserList)
@Override
public void onUserAudioAvailable(String userId, boolean available) {
    available ? mMicrophoneUserList.add(userId) : mMicrophoneUserList.remove(userId);
}
```

```
// 感知远端用户进入房间的通知，并更新远端用户列表(mUserList)
@Override
public void onRemoteUserEnterRoom(String userId) {
    mUserList.add(userId);
}

// 感知远端用户离开房间的通知，并更新远端用户列表(mUserList)
@Override
public void onRemoteUserLeaveRoom(String userId, int reason) {
    mUserList.remove(userId);
}
```



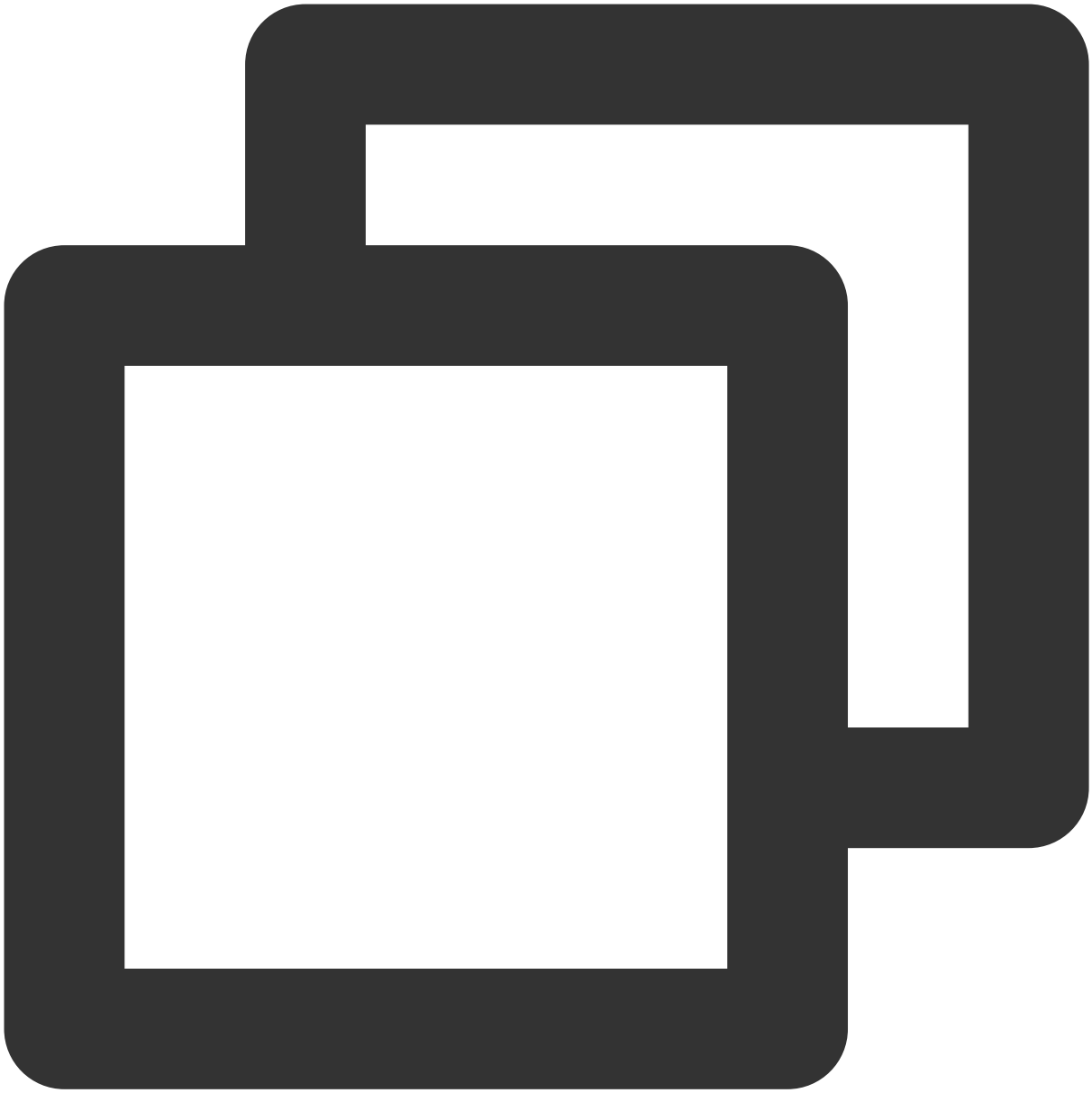
```
// 感知远端用户视频状态的变化，并更新开启了摄像头的用户列表(mCameraUserList)
- (void)onUserVideoAvailable:(NSString *)userId available:(BOOL)available{
    if (available) {
        [mCameraUserList addObject:userId];
    } else {
        [mCameraUserList removeObject:userId];
    }
}

// 感知远端用户音频状态的变化，并更新开启了麦克风的用户列表(mMicrophoneUserList)
- (void)onUserAudioAvailable:(NSString *)userId available:(BOOL)available{
```

```
    if (available) {
        [mMicrophoneUserList addObject:userId];
    } else {
        [mMicrophoneUserList removeObject:userId];
    }
}

// 感知远端用户进入房间的通知，并更新远端用户列表(mUserList)
- (void)onRemoteUserEnterRoom:(NSString *)userId{
    [mUserList addObject:userId];
}

// 感知远端用户离开房间的通知，并更新远端用户列表(mUserList)
- (void)onRemoteUserLeaveRoom:(NSString *)userId reason:(NSInteger)reason{
    [mUserList removeObject:userId];
}
```



```
// 感知远端用户视频状态的变化，并更新开启了摄像头的用户列表(mCameraUserList)
void onUserVideoAvailable(const char* user_id, bool available) {
    available ? mCameraUserList.push_back(user_id) : mCameraUserList.remove(user_id
}

// 感知远端用户音频状态的变化，并更新开启了麦克风的用户列表(mMicrophoneUserList)
void onUserAudioAvailable(const char* user_id, bool available) {
    available ? mMicrophoneUserList.push_back(user_id) : mMicrophoneUserList.remove
}

// 感知远端用户进入房间的通知，并更新远端用户列表(mUserList)
```

```
void onRemoteUserEnterRoom(const char* user_id) {
    mUserList.push_back(user_id);
}

// 感知远端用户离开房间的通知，并更新远端用户列表(mUserList)
void onRemoteUserLeaveRoom(const char* user_id, int reason) {
    mUserList.remove(user_id);
}
```

进阶指引

1. 同样是“静音”，区别在哪里？

随着您的业务需求不断深入，您会发现有三种不同的“静音”，虽然他们都叫“静音”，但是计数原理却完全不同：

第一种：播放端停止订阅音频流

如果您调用了 muteRemoteAudio("denny", true) 函数，代表您不希望再播放远端用户 denny 的声音，此时 SDK 会停止拉取 denny 的音频数据流。这种模式是比较节省流量了。当时当您希望再次播放 denny 的声音时，SDK 需要重新启动一次音频数据的拉取流程，所以从“静音”到“解除静音”的状态，恢复速度会比较慢。

第二种：将播放音量调整为零

如果您的业务场景需要静音切换有更快地反应速度，您可以通过 setRemoteAudioVolume("denny", 0) 将远端用户 denny 的播放音量设置为零，该接口由于不涉及网络操作，因此影响速度非常快。

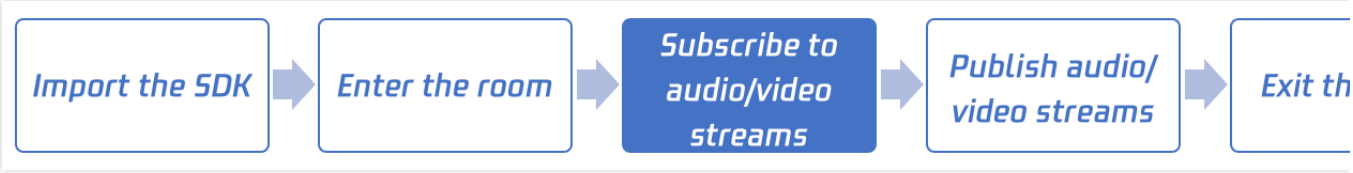
第三种：远端用户自己关闭麦克风

本文档介绍的所有操作都是在介绍播放端的操作，这些操作所产生的效果仅对当前用户生效，比如您通过 muteRemoteAudio("denny", true) 将远端用户 denny 设置为静音，房间中的其他用户还是能听到 denny 的声音。如果要让 denny 彻底“闭嘴”，就需要去影响 denny 的音频发布行为，我们会在下一篇文档 发布音视频流 中详细介绍。

Web

最近更新时间：2023-08-23 11:11:31

本文档主要介绍如何订阅房间中其他用户的音视频流，也就是如何播放其他用户的音频和视频。为了方便起见，我们在接下来的文档中，会将“房间中的其他用户”统称为“远端用户”。



步骤1：进入房间

可以参考文档 进入房间。

步骤2：播放远端音视频

播放远端音频

默认情况下，SDK 会自动播放远端音频，您无需调用 API 播放远端音频。

若您不希望 SDK 自动播放音频，您可以：

1. 在 `trtc.enterRoom({ autoReceiveAudio: false })` 时设置 autoReceiveAudio = false 关闭自动播放音频。
2. 监听 TRTC.EVENT.REMOTE_AUDIO_AVAILABLE 事件。
3. 在收到该事件时，保存远端音频的 userId。
4. 在需要播放音频时，调用 `trtc.muteRemoteAudio(userId, false)` 方法。

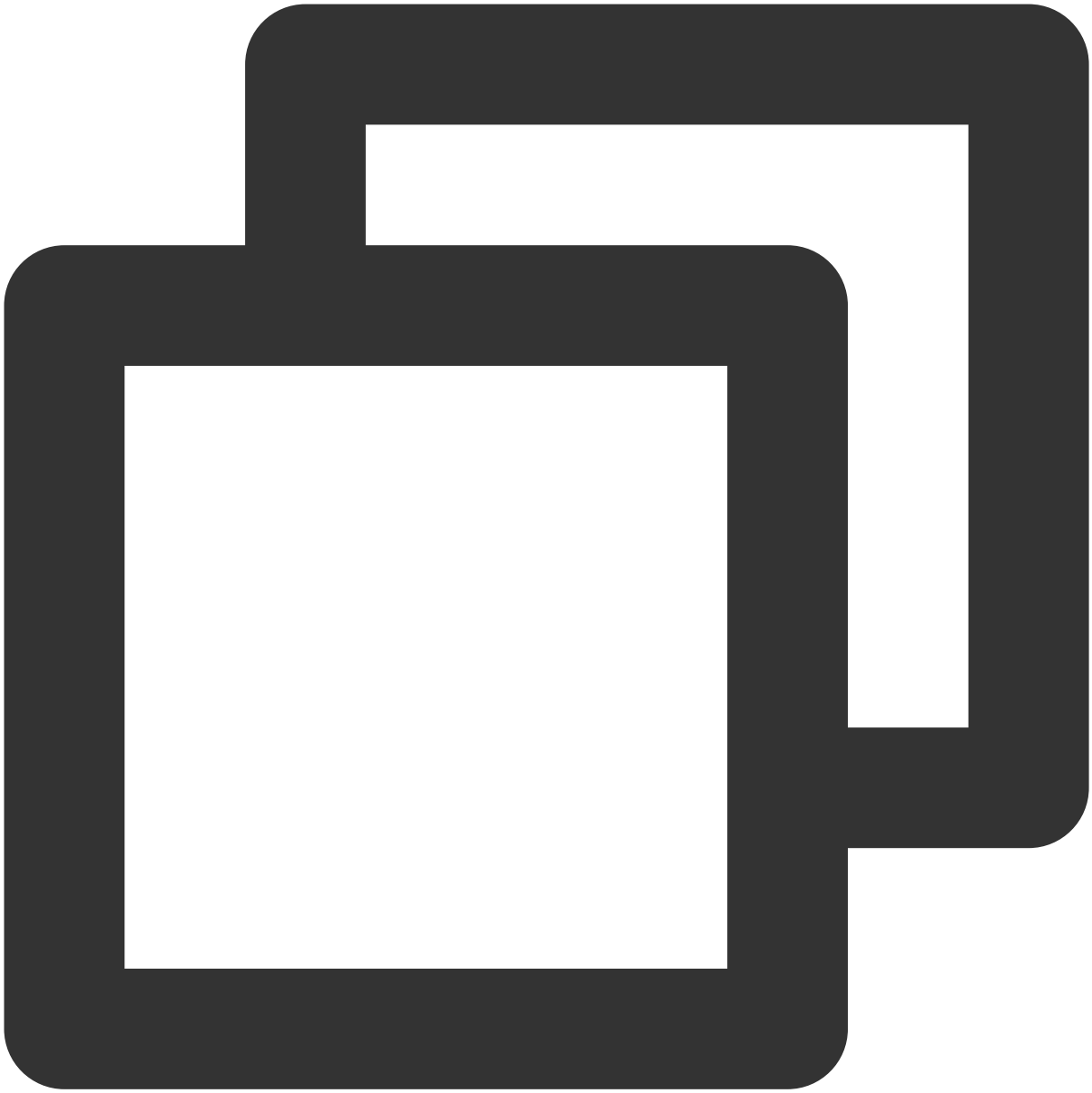
注意：

如果用户在进房前没有与页面产生过交互，自动播放音频可能会因为【浏览器的自动播放策略限制】而失败，您需参考自动播放受限处理建议进行处理。

播放远端视频

1. 在进房前监听 TRTC.EVENT.REMOTE_VIDEO_AVAILABLE 事件。
2. 在收到该事件时，通过 `trtc.startRemoteVideo()` 播放远端视频流。

版权所有：腾讯云计算（北京）有限责任公司

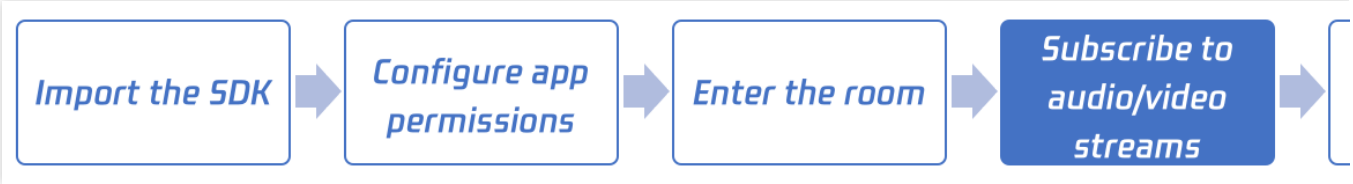


```
trtc.on(TRTC.EVENT.REMOTE_VIDEO_AVAILABLE, ({ userId, streamType }) => {
  // 为了播放视频画面，您需在 DOM 中放置一个 HTMLElement，可以是一个 div 标签，假设其 id 为 `
  const view = `${userId}_${streamType}`;
  trtc.startRemoteVideo({ userId, streamType, view });
});
```

Electron

最近更新时间：2024-06-07 11:20:49

本文档主要介绍如何订阅房间中其他用户的音视频流，也就是如何播放其他用户的音频和视频。为了方便起见，我们在接下来的文档中，会将“房间中的其他用户”统称为“远端用户”。



调用指引

步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入。

步骤2：设定订阅模式（非必须）

您可以通过调用 TRTCCloud 中的 **setDefaultStreamRecvMode** 接口设置订阅模式，TRTC 提供了两种订阅模式：
自动订阅：SDK 会自动播放远端用户的声音，无需您进行额外操作，这是 SDK 的默认行为。
手动订阅：SDK 不会自动拉取和播放远端用户的声音，需要您手动调用 **muteRemoteAudio(userId, false)** 来触发声音的播放。

注意：

需要您注意的是，如果您不调用 setDefaultStreamRecvMode 也是没有关系的，SDK 的默认行为是自动订阅。但如果您希望设置为手动订阅，**请务必注意 setDefaultStreamRecvMode 只有在 enterRoom 之前调用才有效果。**

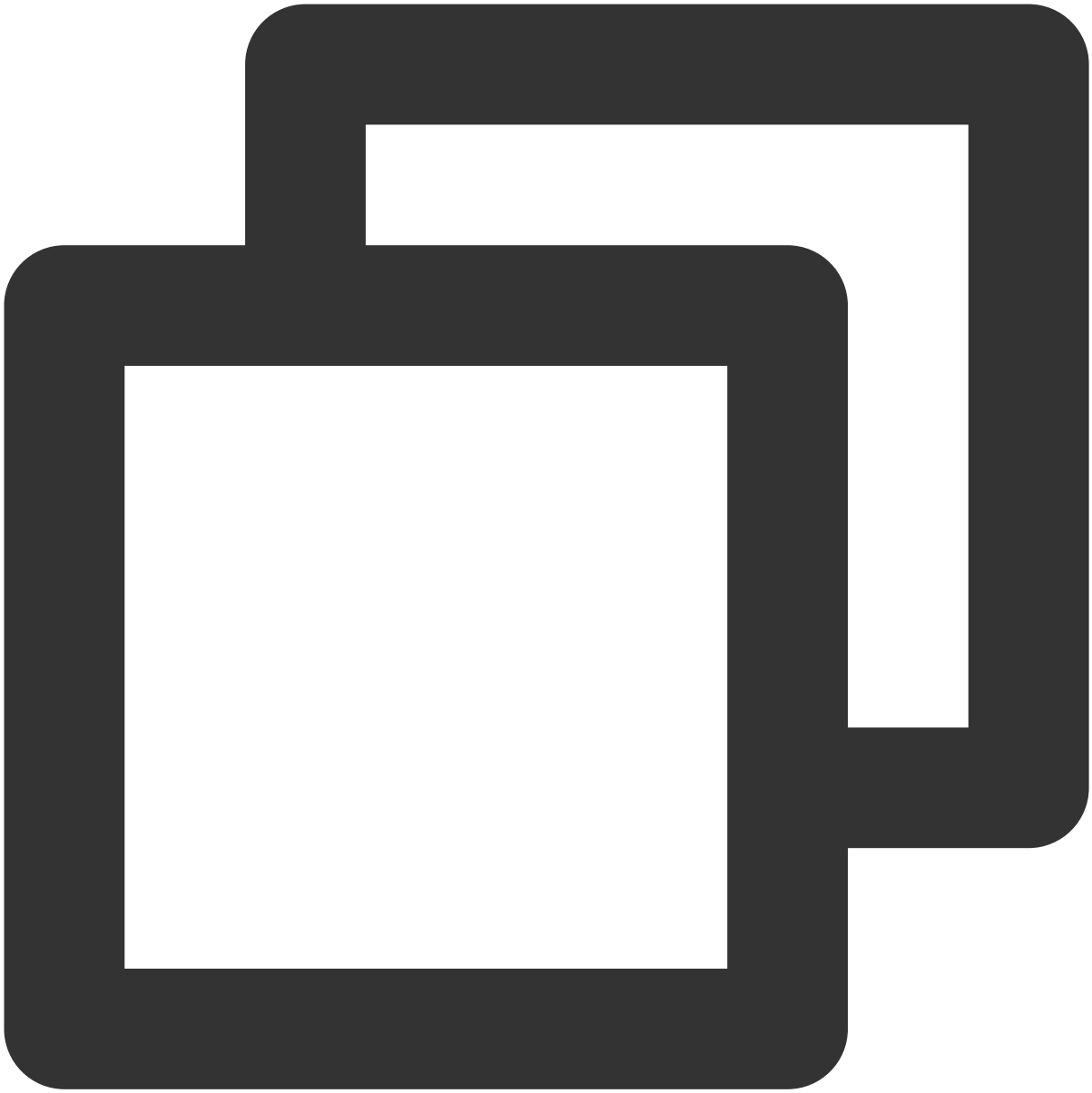
步骤3：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间，只有在成功进入房间之后才能订阅其他用户的音视频流。

步骤4：音频流的播放

您可以通过调用接口 muteRemoteAudio("denny"，true) 来静音远端用户 denny 的声音，之后也可以通过调用接口 muteRemoteAudio('denny'，false) 来解除对他的静音。

版权所有：腾讯云计算（北京）有限责任公司



```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();

// Reference https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/T
// Mute user with id denny
rtcCloud.muteRemoteAudio('denny', true);
// Unmute user with id denny
rtcCloud.muteRemoteAudio('denny', false);
```

步骤5：视频流的播放

1. 开始和停止播放（startRemoteView + stopRemoteView）

您可以通过调用接口 startRemoteView 来播放远端用户的视频画面，但前提是您需要传递给 SDK 一个 view 对象，用来作为承载该用户的视频画面的渲染控件。

startRemoteView 的第一个参数是远端用户的 userId，第二个参数是远端用户的流类型，第三个参数则是您需要传递的 view 对象。其中第二个参数 streamType(流类型) 有三个可选的值，分别是：

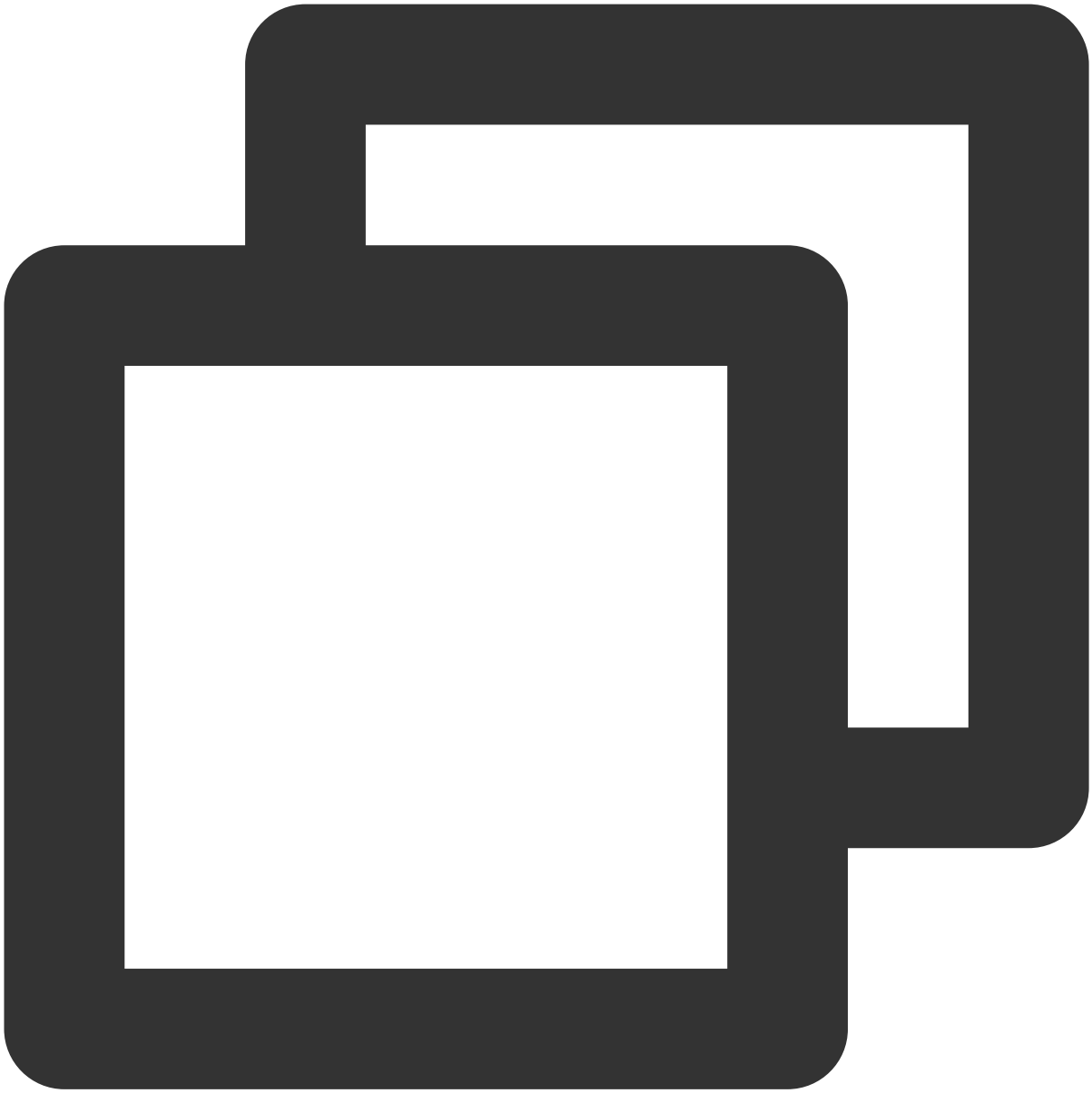
TRTCVideoStreamTypeBig：用户的主路画面，一般用来传输用户摄像头的画面。

TRTCVideoStreamTypeSub：用户的辅路画面，一般用来传输用户屏幕分享的画面。

TRTCVideoStreamTypeSmall：用户的低清小画面，这是相对于主路画面而言的，只有当远端用户开启了“双路编码（enableEncSmallVideoStream）”之后，我们才能播放他/她的低清画面。并且同时只能在主路画面和低清小画面中二选一。

通过调用 stopRemoteView 接口，您可以停止播放某一个远端用户的视频，也可以通过 stopAllRemoteView 接口停止播放所有远端用户的视频。

版权所有：腾讯云计算（北京）有限责任公司



```
// 可参考 https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCC

import TRTCCloud, { TRTCVideoStreamType } from 'trtc-electron-sdk';

const cameraView = document.querySelector('.user-dom');
const screenView = document.querySelector('.screen-dom');
const rtcCloud = new TRTCCloud();
// 播放 denny 的摄像头画面（我们称之为“主路画面”）
rtcCloud.startRemoteView('denny', cameraView, TRTCVideoStreamType.TRTCVideoStreamTy
// 播放 denny 的屏幕分享画面（我们称之为“辅路画面”）
rtcCloud.startRemoteView('denny', screenView, TRTCVideoStreamType.TRTCVideoStreamTy
```

```
// 播放 denny 的低分辨率画面（主路画面和低清画面只能二选一）
rtcCloud.startRemoteView('denny', cameraView, TRTCVideoStreamType.TRTCVideoStreamTy
// 停止播放 denny 的摄像头画面
rtcCloud.stopRemoteView('denny', TRTCVideoStreamType.TRTCVideoStreamTypeBig);
// 停止播放所有视频画面
rtcCloud.stopAllRemoteView();
```

2. 设置播放参数（setRemoteRenderParams）

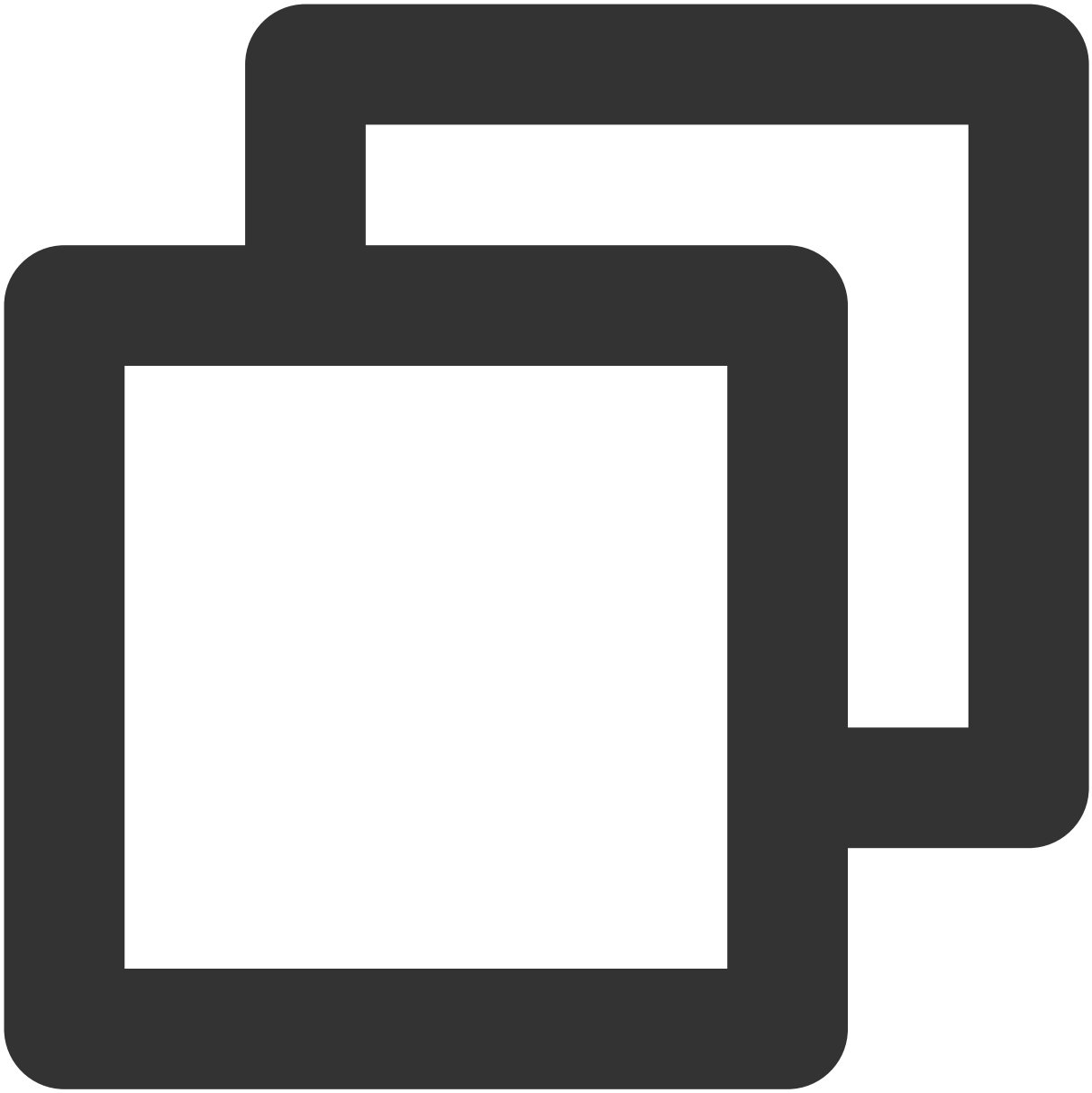
通过 setRemoteRenderParams 您可以设置画面的填充模式、旋转角度和镜像模式。

填充模式：分为填充或者适应，两种模式下画面都能保持原始的宽高比例，区别在于有无黑边。

旋转角度：可以设置 0度、90度、180度 和 270度等四个旋转角度。

镜像模式：即画面的左右镜像模式。

版权所有：腾讯云计算（北京）有限责任公司



```
// 可参考 https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCC
// 将远端用户 denny 的主路画面设置为填充模式，并开启左右镜像模式
import TRTCCloud, {
  TRTCRenderParams, TRTCVideoStreamType, TRTCVideoRotation,
  TRTCVideoFillMode, TRTCVideoMirrorType
} from 'trtc-electron-sdk';

const param = new TTRTCRenderParams(
  TRTCVideoRotation.TRTCVideoRotation0,
  TRTCVideoFillMode.TRTCVideoFillMode_Fill,
  TRTCVideoMirrorType.TRTCVideoMirrorType_Enable
```

```
);
const rtcCloud = new TRTCCloud();
rtcCloud.setRemoteRenderParams('denny', TRTCVideoStreamType.TRTCVideoStreamTypeBig,
```

步骤6：感知房间中远端用户的音视频状态

在 步骤4 和 步骤5 中，您可以控制对远端用户的声音和视频的播放，但是如果没有足够的信息，您并不知道：
当前房间里有哪些用户？
他们是否开启了摄像头和麦克风？
为了解决这个问题，您需要监听来自 SDK 的几个事件回调：

音频状态变化通知（onUserAudioAvailable）

当远端用户开启或关闭麦克风时，您可以通过监听 onUserAudioAvailable(userId，boolean) 来感知到这个状态的变化。

视频状态变化通知（onUserVideoAvailable）

当远端用户开启或者关闭摄像头画面时，您可以通过监听 onUserVideoAvailable(userId，boolean) 来感知到这个状态的变化。

当远端用户开启或关闭屏幕分享画面时，您可以通过监听 onUserSubStreamAvailable(userId，boolean) 来感知到这个状态的变化。

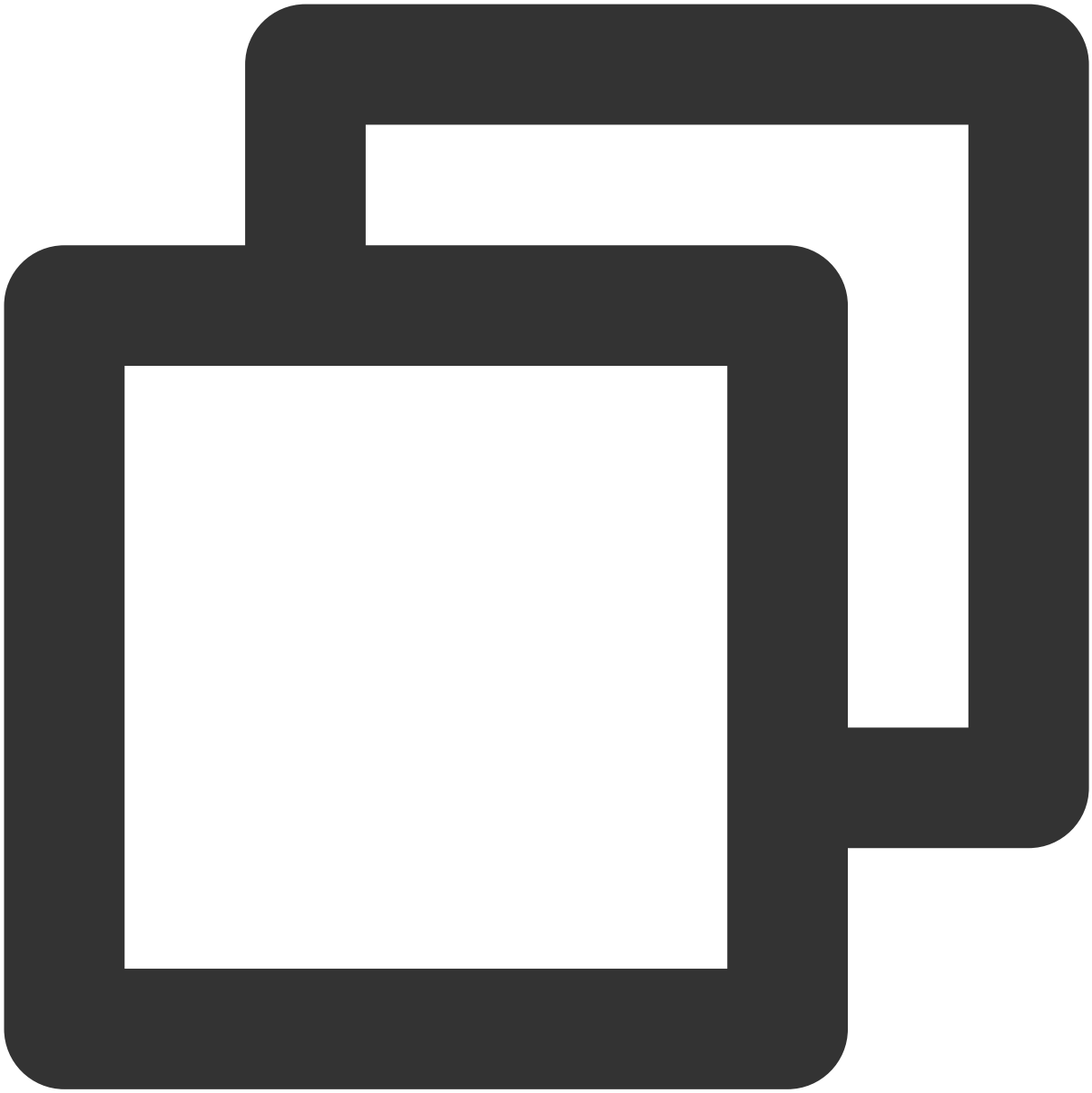
用户进出房间的通知（onRemoteUserEnter/LeaveRoom）

当有远端用户进入当前房间时，您可以通过 onRemoteUserEnterRoom(userId) 来感知到该用户的 userId，当有远端用户离开当前方式时，您可以通过 onRemoteUserLeaveRoom(userId, reason) 来感知到该用户的 userId 以及他/她离开的原因。

注意：

准确地说，onRemoteUserEnter/LeaveRoom 仅能感知角色（role）为主播（anchor）的用户的进出房间的通知，之所以这样设计，是为了避免当一个房间中观众（audience）在线人数较多时，会因为频繁有人进出房间导致房间中的所有用户被其他用户进出房间的“信令风暴“攻击到。

有了这些事件回调，您就可以掌握房间中有哪些用户，以及他们是否开启了摄像头和麦克风，参考如下的示例代码，在这段示例代码中，我们使用了 mCameraUserList、mMicrophoneUserList 以及 mUserList 分别维护了：
房间中的用户（准确的说是主播）是哪些人
其中开启摄像头的用户是哪些人
其中开启麦克风的用户是哪些人



```
import TRTCCloud from 'trtc-electron-sdk';
let openCameraUserList = [];
let openMicUserList = [];
let roomUserList = [];

function onUserVideoAvailable(userId, available) {
  if (available === 1) {
    openCameraUserList.push(userId);
  } else {
    openCameraUserList = openCameraUserList.filter((item) => item !== userId);
  }
```

```
}

function onUserAudioAvailable(userId, available) {
  if (available === 1) {
    openMicUserList.push(userId);
  } else {
    openMicUserList = openMicUserList.filter((item) => item !== userId);
  }
}

function onRemoteUserEnterRoom(userId) {
  roomUserList.push(userId);
}

function onRemoteUserLeaveRoom(userId, reason) {
  roomUserList = roomUserList.filter((item) => item !== userId);
}

const rtcCloud = new TRTCCloud();
rtcCloud.on('onUserVideoAvailable', onUserVideoAvailable);
rtcCloud.on('onUserAudioAvailable', onUserAudioAvailable);
rtcCloud.on('onRemoteUserEnterRoom', onRemoteUserEnterRoom);
rtcCloud.on('onRemoteUserLeaveRoom', onRemoteUserLeaveRoom);
```

进阶指引

同样是“静音”，区别在哪里？

随着您的业务需求不断深入，您会发现有三种不同的“静音”，虽然他们都叫“静音”，但是技术原理却完全不同：

第一种：播放端停止订阅音频流

如果您调用了 muteRemoteAudio("denny", true) 函数，代表您不希望再播放远端用户 denny 的声音，此时 SDK 会停止拉取 denny 的音频数据流。这种模式是比较节省流量了。但是当您希望再次播放 denny 的声音时，SDK 需要重新启动一次音频数据的拉取流程，所以从“静音”到“解除静音”的状态，恢复速度会比较慢。

第二种：将播放音量调整为零

如果您的业务场景需要静音切换有更快地反应速度，您可以通过 setRemoteAudioVolume("denny", 0) 将远端用户 denny 的播放音量设置为零，该接口由于不涉及网络操作，因此影响速度非常快。

第三种：远端用户自己关闭麦克风

本文档介绍的所有操作都是在介绍播放端的操作，这些操作所产生的效果仅对当前用户生效，比如您通过 muteRemoteAudio("denny", true) 将远端用户 denny 设置为静音，房间中的其他用户还是能听到 denny 的声音。

如果要让 denny 彻底“闭嘴”，就需要去影响 denny 的音频发布行为，我们会在下一篇文档 发布音视频流 中详细介绍。

Flutter

最近更新时间：2024-02-02 18:51:38

本文档主要介绍如何订阅房间中其他用户的音视频流，也就是如何播放其他用户的音频和视频。为了方便起见，我们在接下来的文档中，会将“房间中的其他用户”统称为“远端用户”。

调用指引

步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。

步骤2：设定订阅模式（非必须）

您可以通过调用 TRTCCloud 中的 **setDefaultStreamRecvMode** 接口设置订阅模式，TRTC 提供了两种订阅模式：
自动订阅：SDK 会自动播放远端用户的声音，无需您进行额外操作，这是 SDK 的默认行为。
手动订阅：SDK 不会自动拉取和播放远端用户的声音，需要您手动调用 **muteRemoteAudio(userId, false)** 来触发声音的播放。

注意：

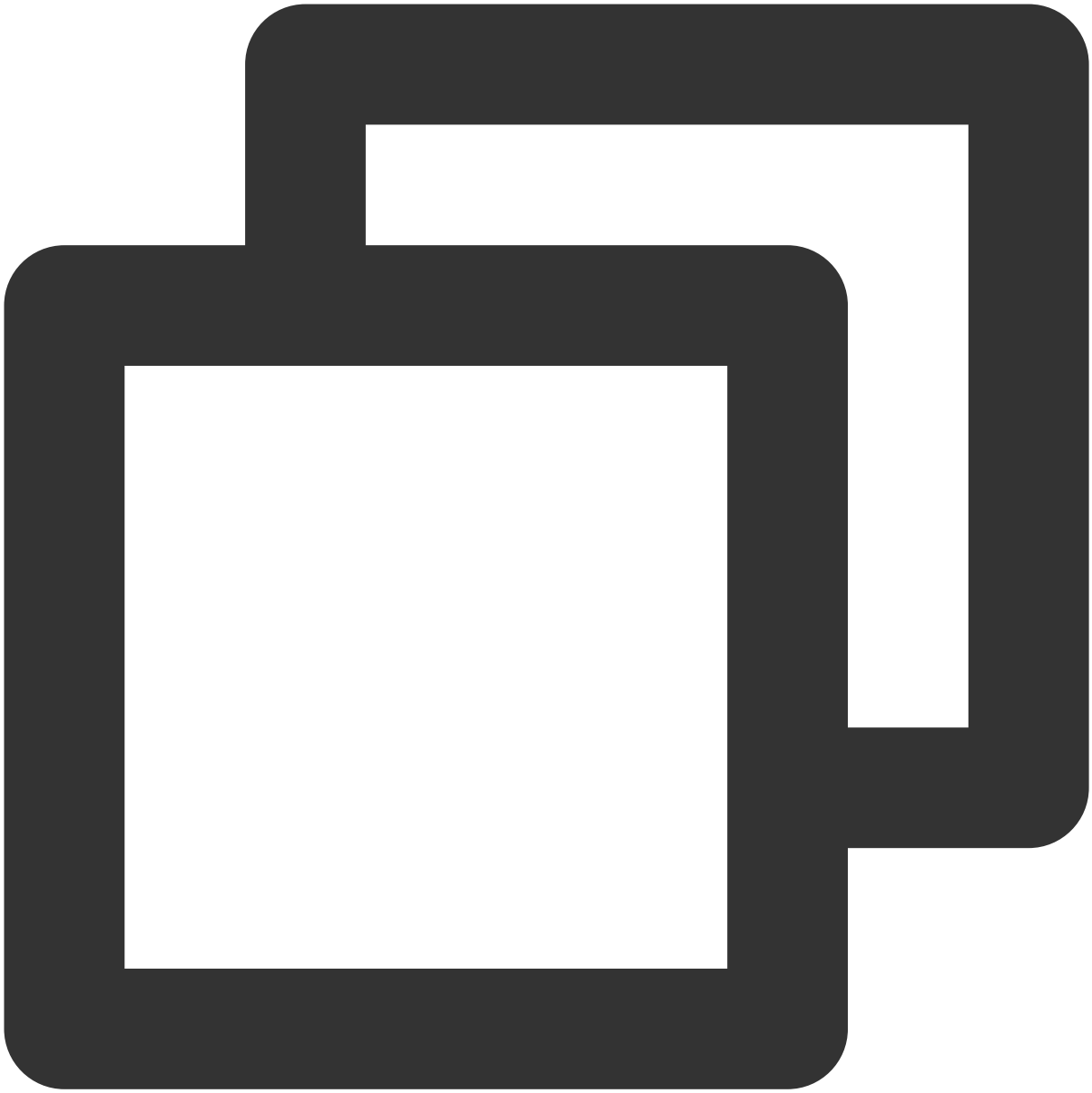
需要您注意的是，如果您不调用 setDefaultStreamRecvMode 也是没有关系的，SDK 的默认行为是自动订阅。但如果您希望设置为手动订阅，请务必注意 setDefaultStreamRecvMode 只有在 enterRoom 之前调用才有效果。

步骤3：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间，只有在成功进入房间之后才能订阅其他用户的音视频流。

步骤4：音频流的播放

您可以通过调用接口 muteRemoteAudio 来静音或者接触指定用户。



```
// Mute user with id denny
trtcCloud.muteRemoteAudio('denny', true);
// Unmute user with id denny
trtcCloud.muteRemoteAudio('denny', false);
```

步骤5：视频流的播放

1. 开始和停止播放(startRemoteView + stopRemoteView)

您可以通过调用接口 startRemoteView 来播放远端用户的视频画面，但前提是您需要传递给 SDK 一个 view 对象，用来作为承载该用户的视频画面的渲染控件。

startRemoteView 的第一个参数是远端用户的 userId，第二个参数是远端用户的流类型，第三个参数则是您需要传递的 view 对象。其中第二个参数 streamType(流类型) 有三个可选的值，分别是：

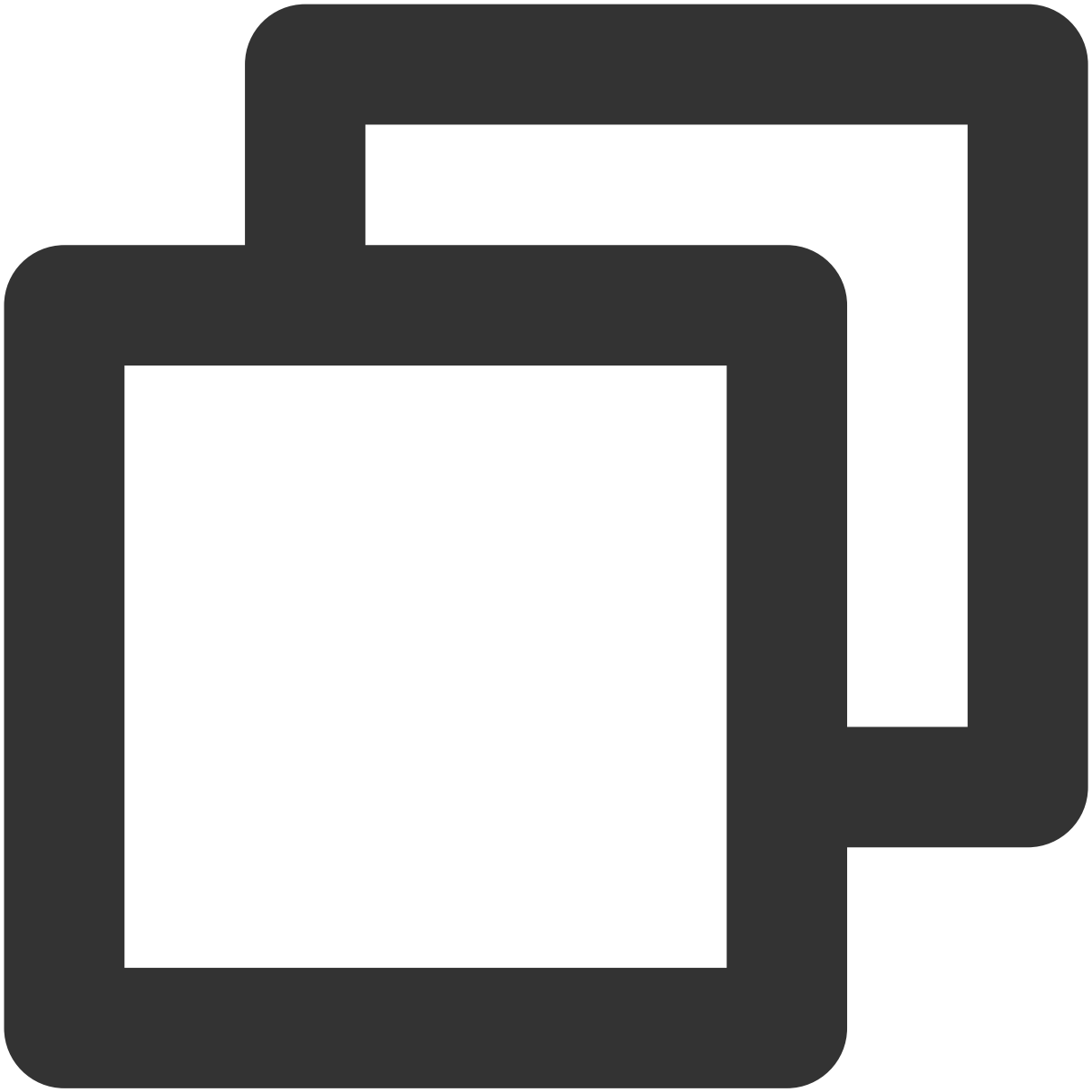
`TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG`：用户的主路画面，一般用来传输用户摄像头的画面。

`TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SUB`：用户的辅路画面，一般用来传输用户屏幕分享的画面。

`TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SMALL`：用户的低清小画面，这是相对于主路画面而言的，只有当远端用户开启了“双路编码（enableEncSmallVideoStream）”之后，我们才能播放他/她的低清画面。并且同时只能在主路画面和低清小画面中二选一。

通过调用 stopRemoteView 接口，您可以停止播放某一个远端用户的视频，也可以通过 stopAllRemoteView 接口停止播放所有远端用户的视频。

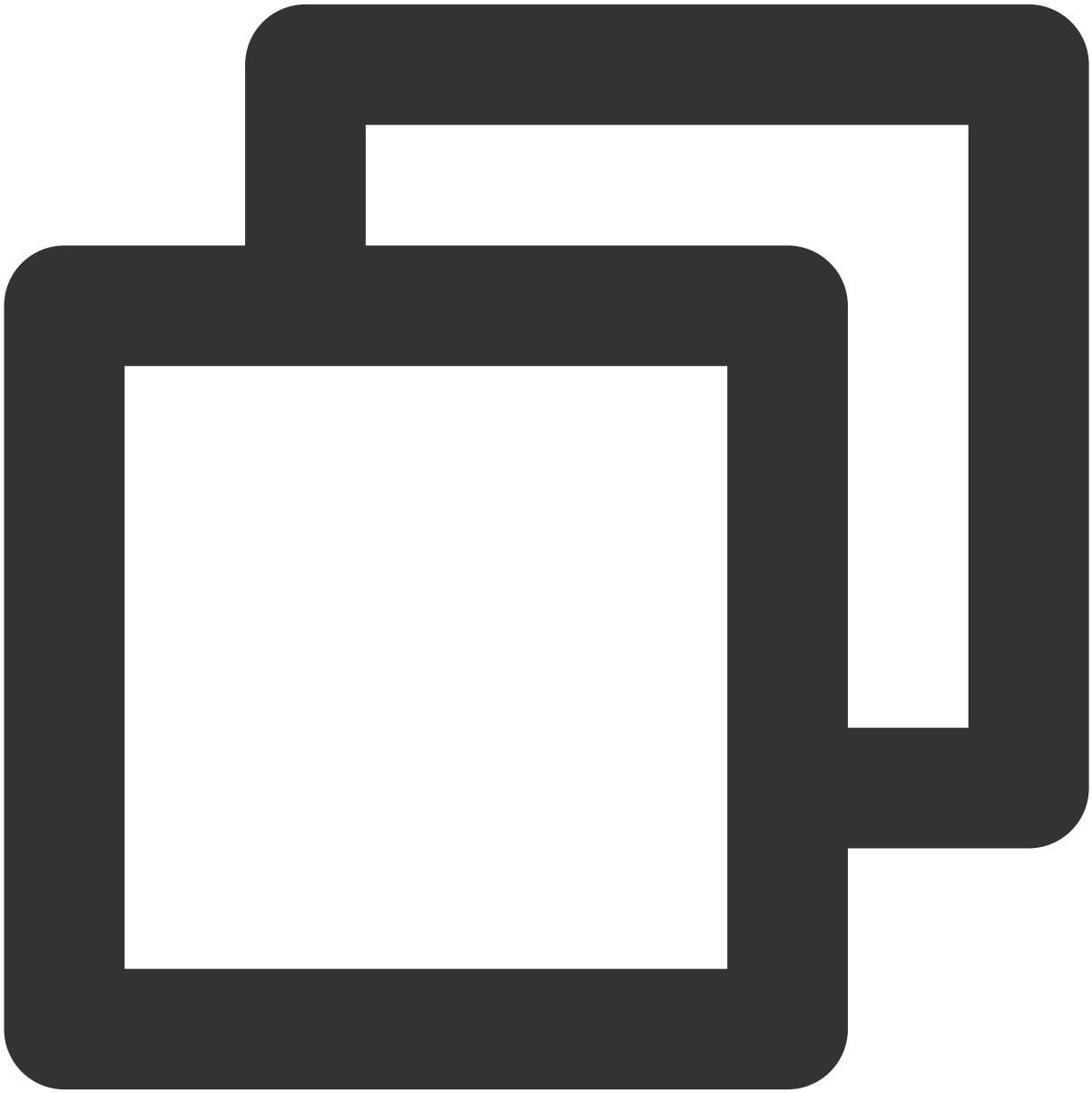
版权所有：腾讯云计算（北京）有限责任公司



```
// 播放 denny 的摄像头画面（我们称之为“主路画面”）
trtcCloud.startRemoteView('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraV
// 播放 denny 的屏幕分享画面（我们称之为“辅路画面”）
trtcCloud.startRemoteView('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SUB, screenV
// 播放 denny 的低分辨率画面（主路画面和低清画面只能二选一）
trtcCloud.startRemoteView('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_SMALL, camer
// 停止播放 denny 的摄像头画面
trtcCloud.stopRemoteView('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, cameraVi
// 停止播放所有视频画面
trtcCloud.stopAllRemoteView();
```

2. 设置播放参数（updateRemoteView + setRemoteRenderParams）

通过调用 updateRemoteView 接口，您可以在播放中变更 view 对象，这在切换视频渲染控件时非常有用。
通过 setRemoteRenderParams 您可以设置画面的填充模式、旋转角度和镜像模式。
填充模式：分为填充或者适应，两种模式下画面都能保持原始的宽高比例，区别在于有无黑边。
旋转角度：可以设置 0度、90度、180度 和 270度等四个旋转角度。
镜像模式：即画面的左右镜像模式。



```
// 将 denny 的主路画面切换到一个悬浮的小窗口中（假如该迷你小窗口为 miniFloatingView）
trtcCloud.updateRemoteView('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, miniFl
```

```
// 将远端用户 denny 的主路画面顺时针旋转90度，同时设置为填充模式，并强制关闭镜像模式
TRTCRenderParams params = TRTCRenderParams(
                    rotation: TRTCCloudDef.TRTC_VIDEO_ROTATION_90,
                    fillMode: TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL,
                    mirrorType: TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_DISABLE);
trtcCloud.setRemoteRenderParams('denny', TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, p
```

步骤6：感知房间中远端用户的音视频状态

在 步骤4 和 步骤5 中，您可以控制对远端用户的声音和视频的播放，但是如果没有足够的信息，您并不知道：
当前房间里有哪些用户？
他们是否开启了摄像头和麦克风？
为了解决这个问题，您需要监听来自 SDK 的几个事件回调：

音频状态变化通知（onUserAudioAvailable）

当远端用户开启或关闭麦克风时，您可以通过监听 onUserAudioAvailable(userId，bool) 来感知到这个状态的变化。

视频状态变化通知（onUserVideoAvailable）

当远端用户开启或者关闭摄像头画面时，您可以通过监听 onUserVideoAvailable(userId，bool) 来感知到这个状态的变化。

当远端用户开启或关闭屏幕分享画面时，您可以通过监听 onUserSubStreamAvailable(userId，bool) 来感知到这个状态的变化。

用户进出房间的通知（onRemoteUserEnter/onRemoteUserLeaveRoom）

当有远端用户进入当前房间时，您可以通过 onRemoteUserEnterRoom(userId) 来感知到该用户的 userId，当有远端用户离开当前方式时，您可以通过 onRemoteUserLeaveRoom(userId, reason) 来感知到该用户的 userId 以及他/她离开的原因。

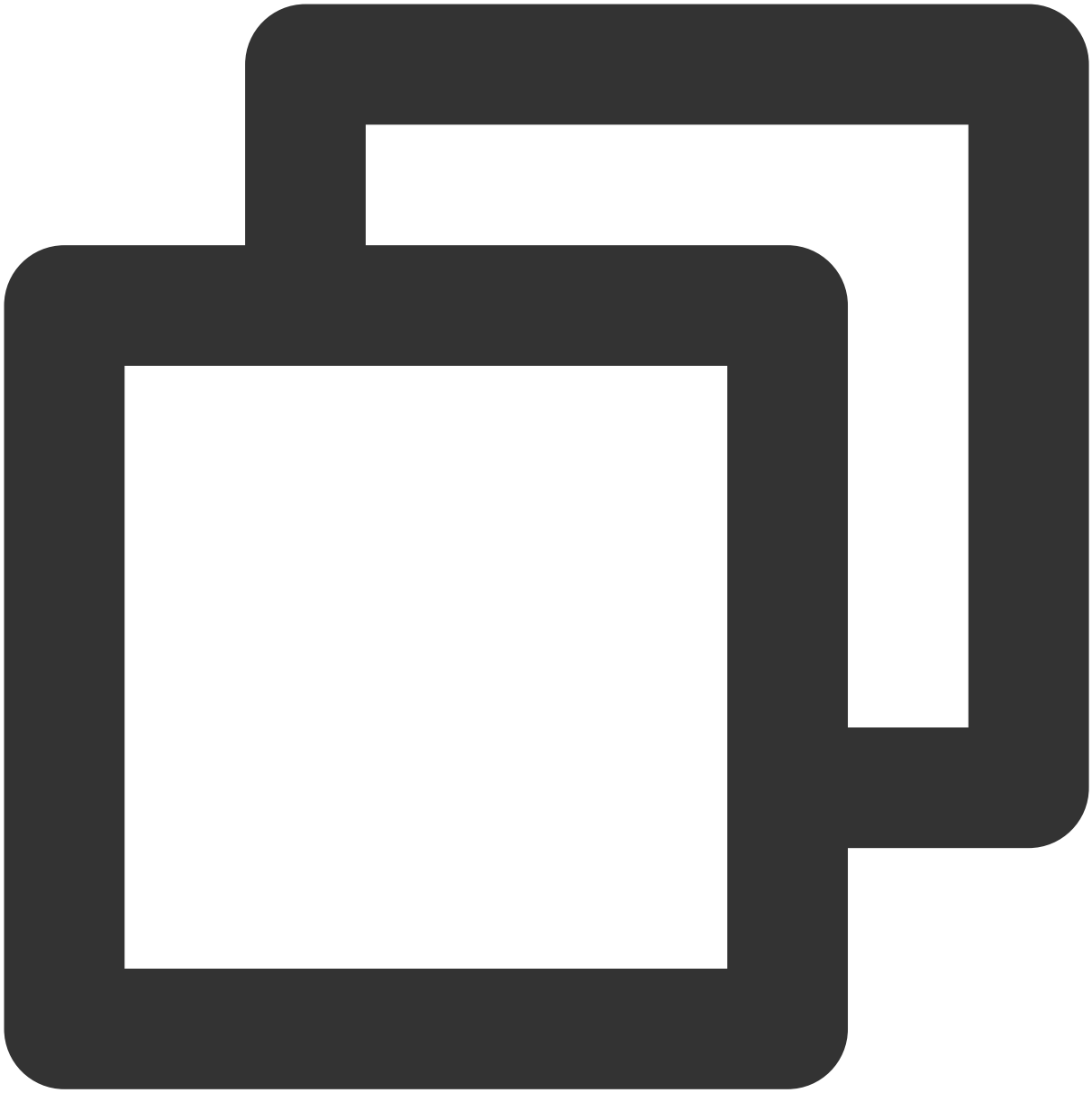
注意：

准确地说，onRemoteUserEnter/LeaveRoom 仅能感知角色（role）为主播（anchor）的用户的进出房间的通知，之所以这样设计，是为了避免当一个房间中观众（audience）在线人数较多时，会因为频繁有人进出房间导致房间中的所有用户被其他用户进出房间的“信令风暴”攻击到。

说明：

在 Dart 中，我们通过一个方法来接收并处理 TRTC SDK 的回调，这个方法的类型被定义为 ListenerValue ，也就是 registerListener 需要传入的参数。ListenerValue 拥有两个参数，分别是代表回调类型的 type 与 SDK 返回的回调的参数 params 。

有了这些事件回调，您就可以掌握房间中有哪些用户，以及他们是否开启了摄像头和麦克风，参考如下的示例代码，在这段示例代码中，我们使用了 userList 、 cameraUserList 以及 microphoneUserList 分别维护了：
房间中的用户（准确的说是主播）是哪些人
其中开启摄像头的用户是哪些人
其中开启麦克风的用户是哪些人



```
onRtcListener(type, param) async {
  if (type == TRTCCloudListener.onUserAudioAvailable) {
    String userId = param['userId'];
    bool available = param['available'];
    available ? microphoneUserList.add(userId) : microphoneUserList.remove(userId);
  }
  if (type == TRTCCloudListener.onUserVideoAvailable) {
    String userId = param['userId'];
    bool available = param['available'];
    available ? cameraUserList.add(userId) : cameraUserList.remove(userId);
```

```
    }
    if (type == TRTCCloudListener.onRemoteUserEnterRoom) {
      userList.add({
        'userId': param
      });
    }
    if (type == TRTCCloudListener.onRemoteUserLeaveRoom) {
      String userId = param['userId'];
      userList.removeWhere((user) => user['userId'] == userId);
    }
  }
```

进阶指引

1. 同样是“静音”，区别在哪里？

随着您的业务需求不断深入，您会发现有三种不同的“静音”，虽然他们都叫“静音”，但是计数原理却完全不同：

第一种：播放端停止订阅音频流

如果您调用了 muteRemoteAudio("denny", true) 函数，代表您不希望再播放远端用户 denny 的声音，此时 SDK 会停止拉取 denny 的音频数据流。这种模式是比较节省流量了。当时当您希望再次播放 denny 的声音时，SDK 需要重新启动一次音频数据的拉取流程，所以从“静音”到“解除静音”的状态，恢复速度会比较慢。

第二种：将播放音量调整为零

如果您的业务场景需要静音切换有更快地反应速度，您可以通过 setRemoteAudioVolume("denny", 0) 将远端用户 denny 的播放音量设置为零，该接口由于不涉及网络操作，因此影响速度非常快。

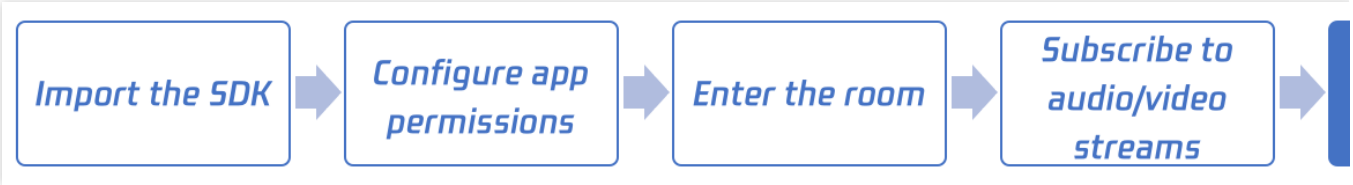
第三种：远端用户自己关闭麦克风

本文档介绍的所有操作都是在介绍播放端的操作，这些操作所产生的效果仅对当前用户生效，比如您通过 muteRemoteAudio("denny", true) 将远端用户 denny 设置为静音，房间中的其他用户还是能听到 denny 的声音。如果要让 denny 彻底“闭嘴”，就需要去影响 denny 的音频发布行为，我们会在下一篇文档 发布音视频流 中详细介绍。

04. 发布音视频流 Android&iOS&Windows&Mac

最近更新时间：2024-06-07 11:20:49

本文档主要介绍主播如何发布自己的音视频流，所谓“发布”，也就是打开麦克风和摄像头，让自己的声音和视频能够被房间中其他用户听到和看到的意思。



调用指引

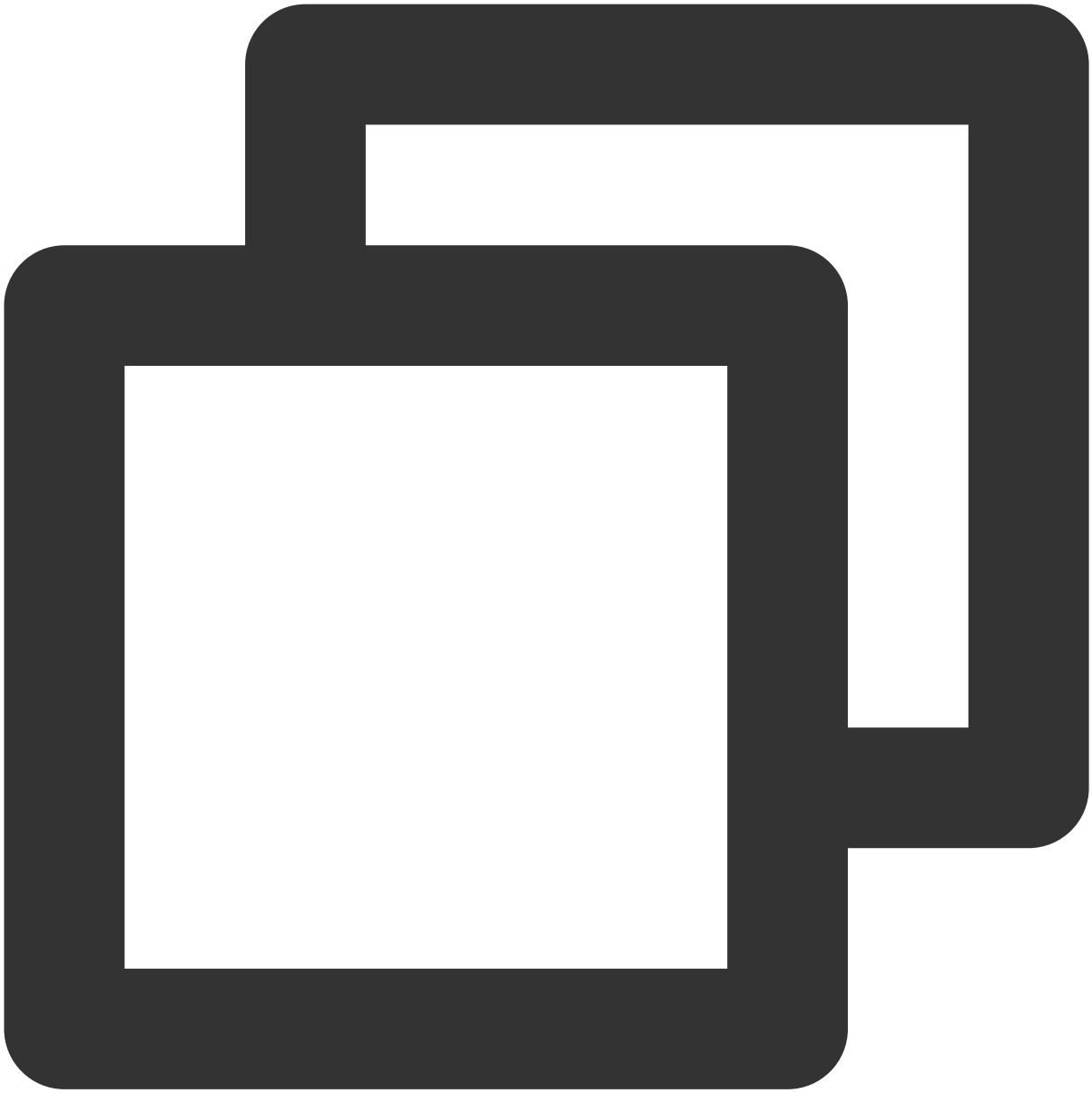
步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。

步骤2：打开摄像头预览

您可以调用 **startLocalPreview** 接口打开摄像头预览，此时 SDK 会向系统申请摄像头的使用权限，需要用户授权通过后才会开启摄像头的采集流程。
如果您希望设置本地画面的渲染参数，可以通过调用 **setLocalRenderParams** 接口来设置本地预览的渲染参数。为防止先开启预览再设置预览参数会出现画面跳动，如果您需要设置预览参数，推荐在开启预览之前调用。
如果您希望控制摄像头的各种控制参数，可以用过调用 **TXDeviceManager** 接口来完成“切换前后摄像头”、“设置对焦模式”、“打开”、“关闭闪光灯”等一系列操作。
如果您希望调节美颜效果和画面质量，我们会在 设定画面质量 中进行详细介绍。

Android
iOS
Mac
Windows

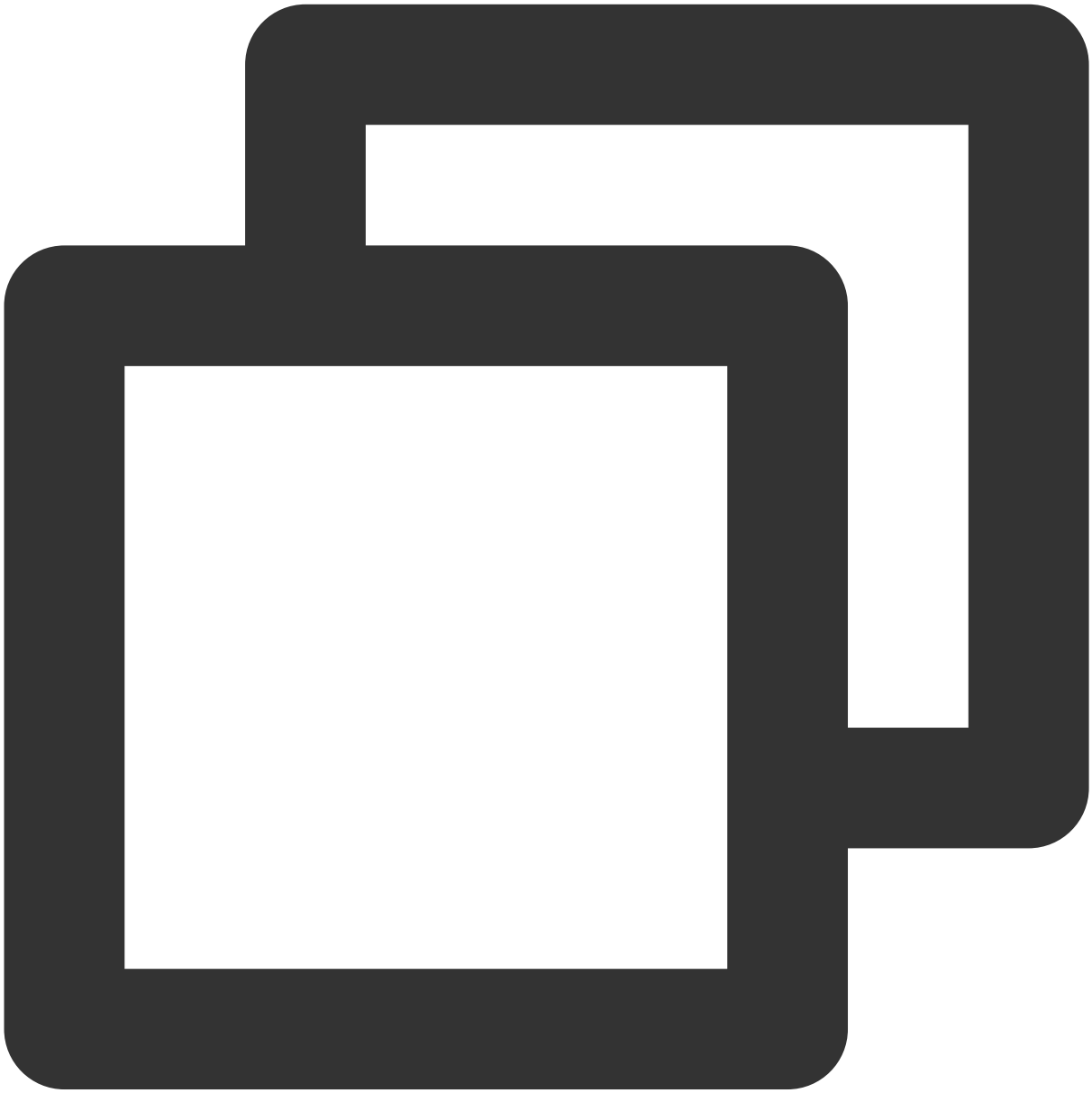


```
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
TRTCCloudDef.TRTCRenderParams param = new TRTCCloudDef.TRTCRenderParams();
param.fillMode   = TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL;
param.mirrorType = TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_AUTO;
mCloud.setLocalRenderParams(param);

// 启动本地摄像头的预览（localCameraVideo 是用于渲染本地渲染画面的控件）
TXCloudVideoView cameraVideo = findViewById(R.id.txcvv_main_local);
mCloud.startLocalPreview(true, cameraVideo);

// 通过 TXDeviceManager 开启自动对焦并将闪光灯打开
```

```
TXDeviceManager manager = mCloud.getDeviceManager();
if (manager.isAutoFocusEnabled()) {
    manager.enableCameraAutoFocus(true);
}
manager.enableCameraTorch(true);
```

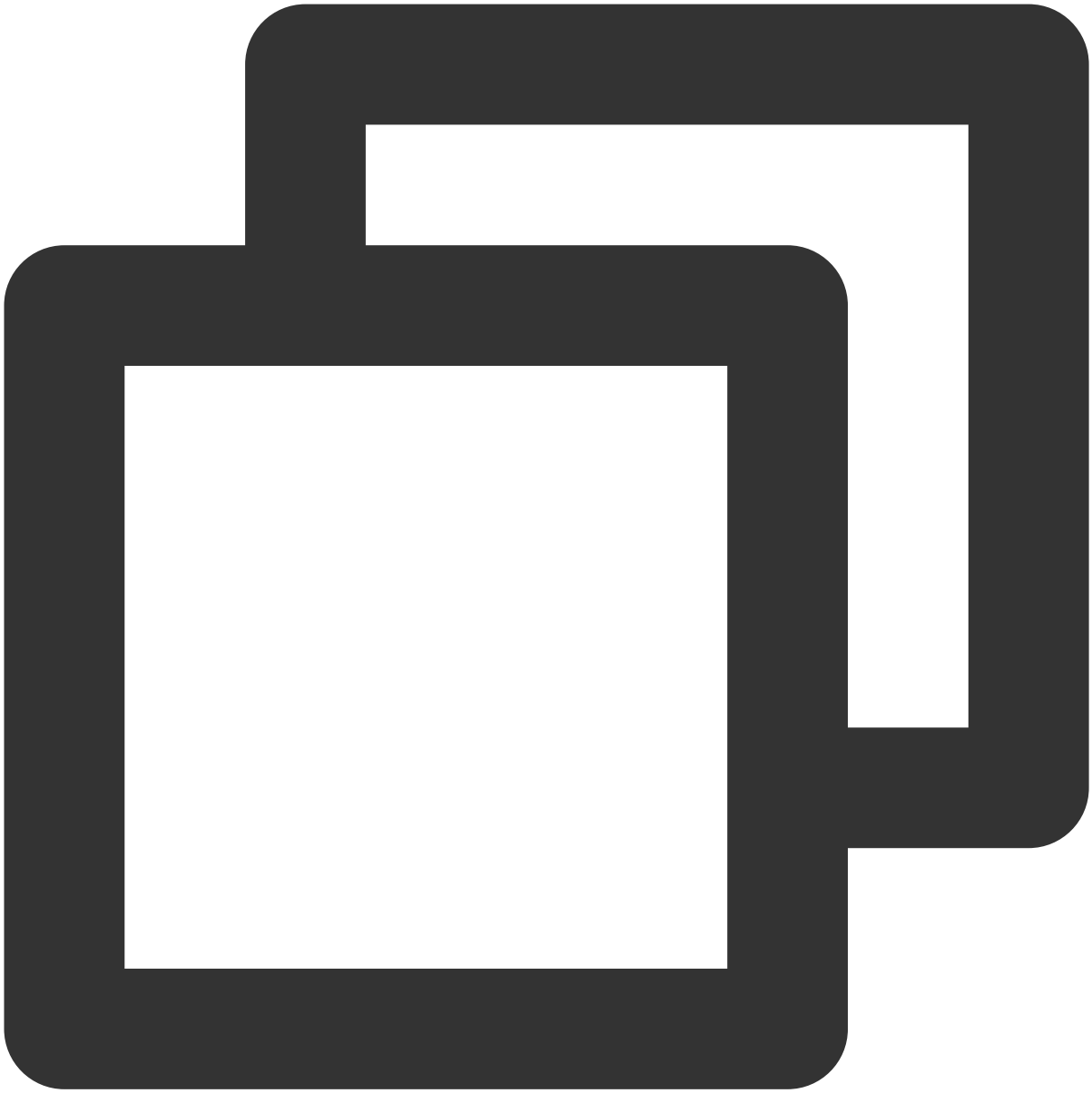


```
self.trtcCloud = [TRTCCloud sharedInstance];
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
TRTCRenderParams *param = [[TRTCRenderParams alloc] init];
param.fillMode   = TRTCVideoFillMode_Fill;
```

```
param.mirrorType = TRTCVideoMirrorTypeAuto;
[self.trtcCloud setLocalRenderParams:param];

// 启动本地摄像头的预览（localCameraVideoView 是用于渲染本地渲染画面的控件）
[self.trtcCloud startLocalPreview:YES view:localCameraVideoView];

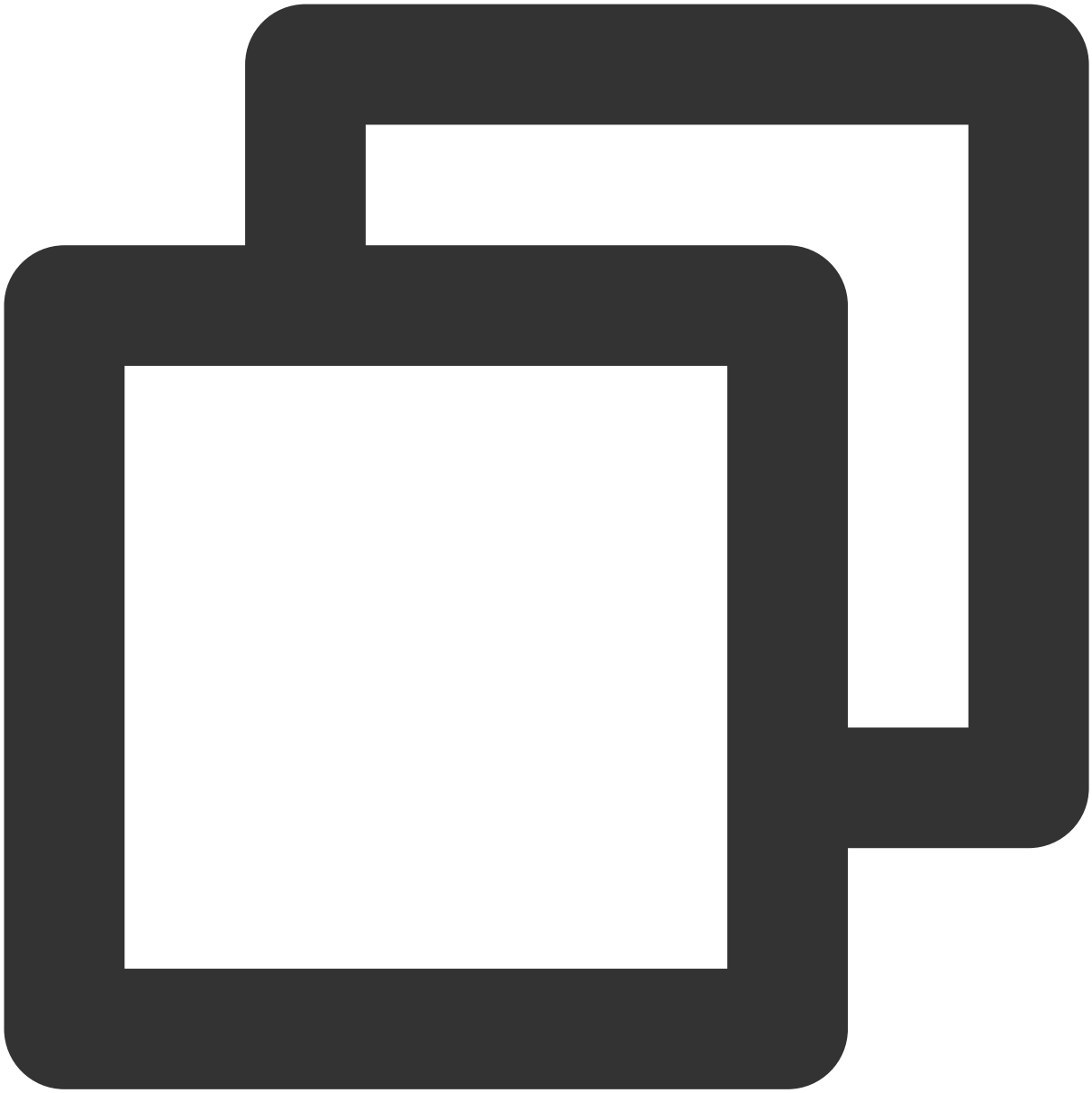
// 通过 TXDeviceManager 开启自动对焦并将闪光灯打开
TXDeviceManager *manager = [self.trtcCloud getDeviceManager];
if ([manager isAutoFocusEnabled]) {
    [manager enableCameraAutoFocus:YES];
}
[manager enableCameraTorch:YES];
```



```
self.trtcCloud = [TRTCCloud sharedInstance];
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
TRTCRenderParams *param = [[TRTCRenderParams alloc] init];
param.fillMode   = TRTCVideoFillMode_Fill;
param.mirrorType = TRTCVideoMirrorTypeAuto;
[self.trtcCloud setLocalRenderParams:param];

// 启动本地摄像头的预览（localCameraVideoView 是用于渲染本地渲染画面的控件）
[self.trtcCloud startLocalPreview:localCameraVideoView];
```

版权所有：腾讯云计算（北京）有限责任公司



```
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
liteav::TRTCRenderParams render_params;
render_params.mirrorType = liteav::TRTCVideoMirrorType_Enable;
render_params.fillMode = TRTCVideoFillMode_Fill;
trtc_cloud_->setLocalRenderParams(render_params);

// 启动本地摄像头的预览（view 是用于渲染本地渲染画面的控件句柄）
liteav::TXView local_view = (liteav::TXView)(view);
trtc_cloud_->startLocalPreview(local_view);
```

版权所有：腾讯云计算（北京）有限责任公司

步骤3：打开麦克风采集

您可以调用 **startLocalAudio** 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式。虽然这个参数的名字叫做 quality，但并不是说质量越高越好，不同的业务场景有最适合的参数选择（这个参数更准确的含义是 scene）。

SPEECH

该模式下的 SDK 音频模块会专注于提炼语音信号，尽最大限度的过滤周围的环境噪音，同时该模式下的音频数据也会获得最好的差质量网络的抵抗能力，因此该模式特别适合于“视频通话”和“在线会议”等侧重于语音沟通的场景。

MUSIC

该模式下的 SDK 会采用很高的音频处理带宽以及立体式模式，在最大限度地提升采集质量的同时也会将音频的 DSP 处理模块调节到最弱的级别，从而最大限度地保证音质。因此该模式适合“音乐直播”场景，尤其适合主播采用专业的声卡进行音乐直播的场景。

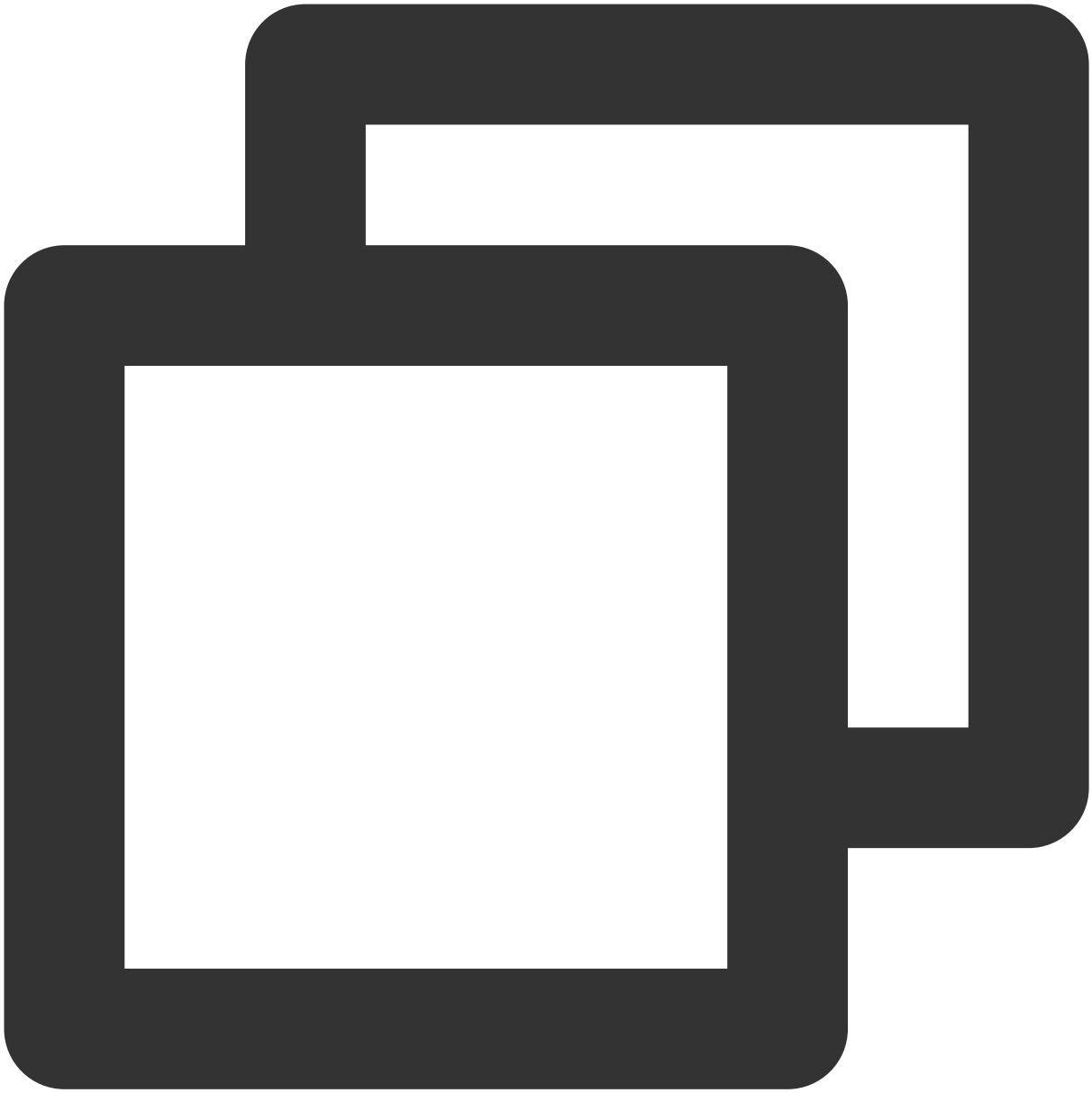
DEFAULT

该模式下的 SDK 会启用智能识别算法来识别当前环境，并针对性地选择最佳的处理模式。不过再好的识别算法也总是有不准确的时候，如果您非常清楚自己的产品定位，更推荐您在专注语音通信的 SPEECH 和专注音乐音质的 MUSIC 之间二选一。

Android

iOS&Mac

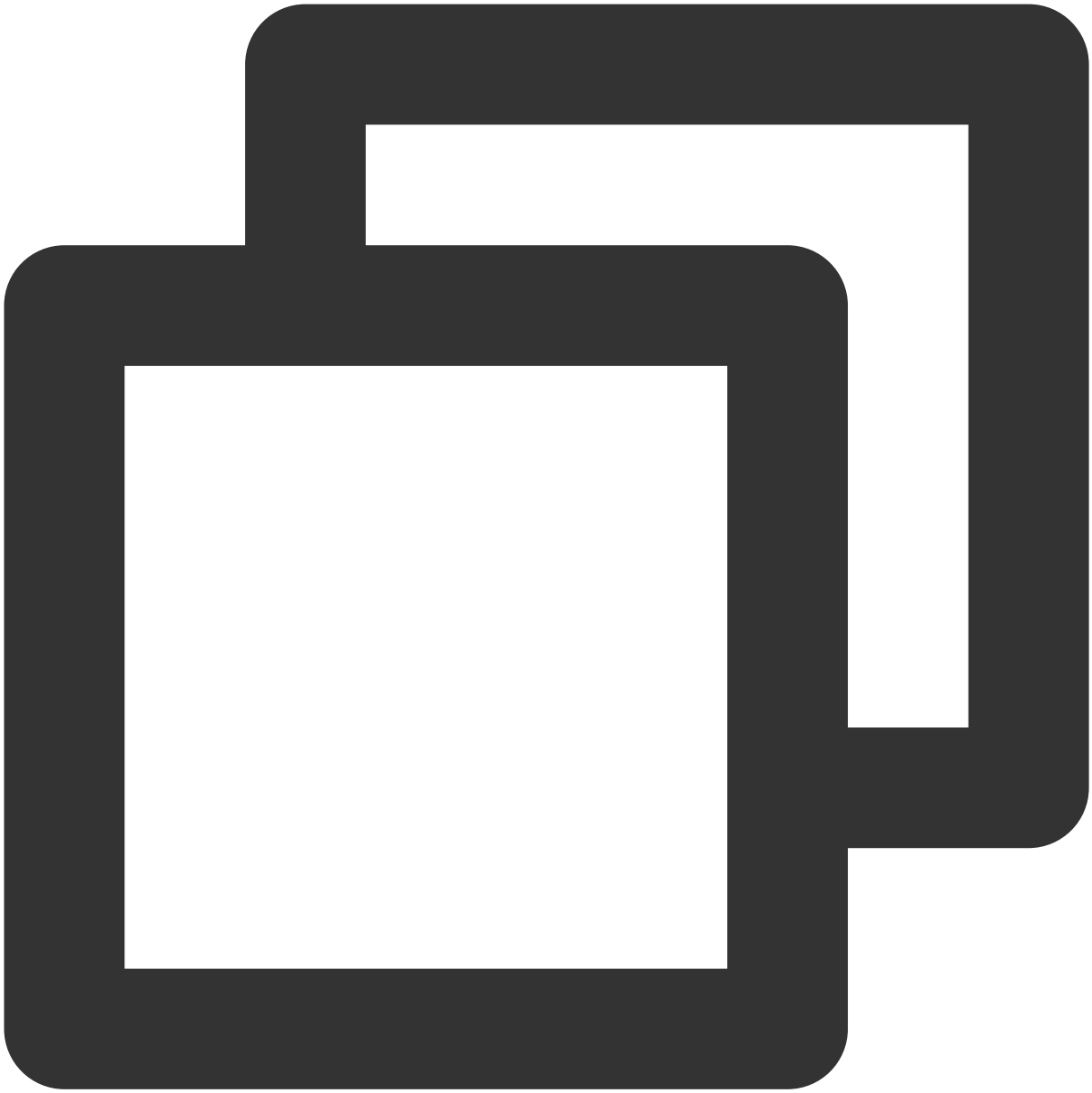
Windows



```
// 开启麦克风采集，并设置当前场景为：语音模式（高噪声抑制能力、强弱网络抗性）
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH );

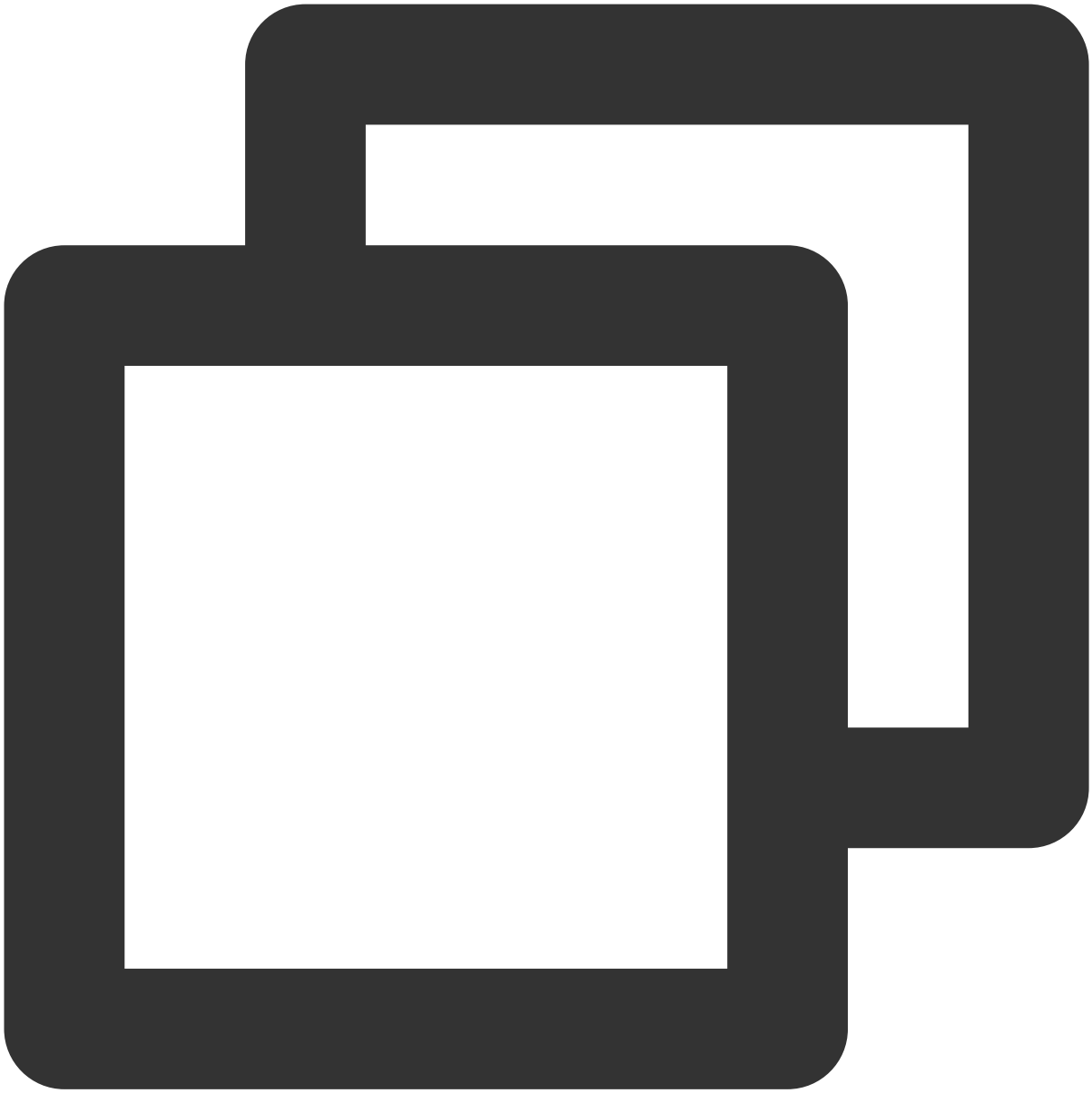
// 开启麦克风采集，并设置当前场景为：音乐模式（高保真采集、低音质损失，推荐配合专业声卡使用）
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
```

版权所有：腾讯云计算（北京）有限责任公司



```
self.trtcCloud = [TRTCCloud sharedInstance];
// 开启麦克风采集，并设置当前场景为：语音模式（高噪声抑制能力、强弱网络抗性）
[self.trtcCloud startLocalAudio:TRTCAudioQualitySpeech];

// 开启麦克风采集，并设置当前场景为：音乐模式（高保真采集、低音质损失，推荐配合专业声卡使用）
[self.trtcCloud startLocalAudio:TRTCAudioQualityMusic];
```



```
// 开启麦克风采集，并设置当前场景为：语音模式（高噪声抑制能力、强弱网络抗性）
trtc_cloud_->startLocalAudio(TRTCAudioQualitySpeech);

// 开启麦克风采集，并设置当前场景为：音乐模式（高保真采集、低音质损失，推荐配合专业声卡使用）
trtc_cloud_->startLocalAudio(TRTCAudioQualityMusic);
```

步骤4：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间。一旦进入房间后，SDK 便会开始向房间中的其他用户发布自己的音频流。

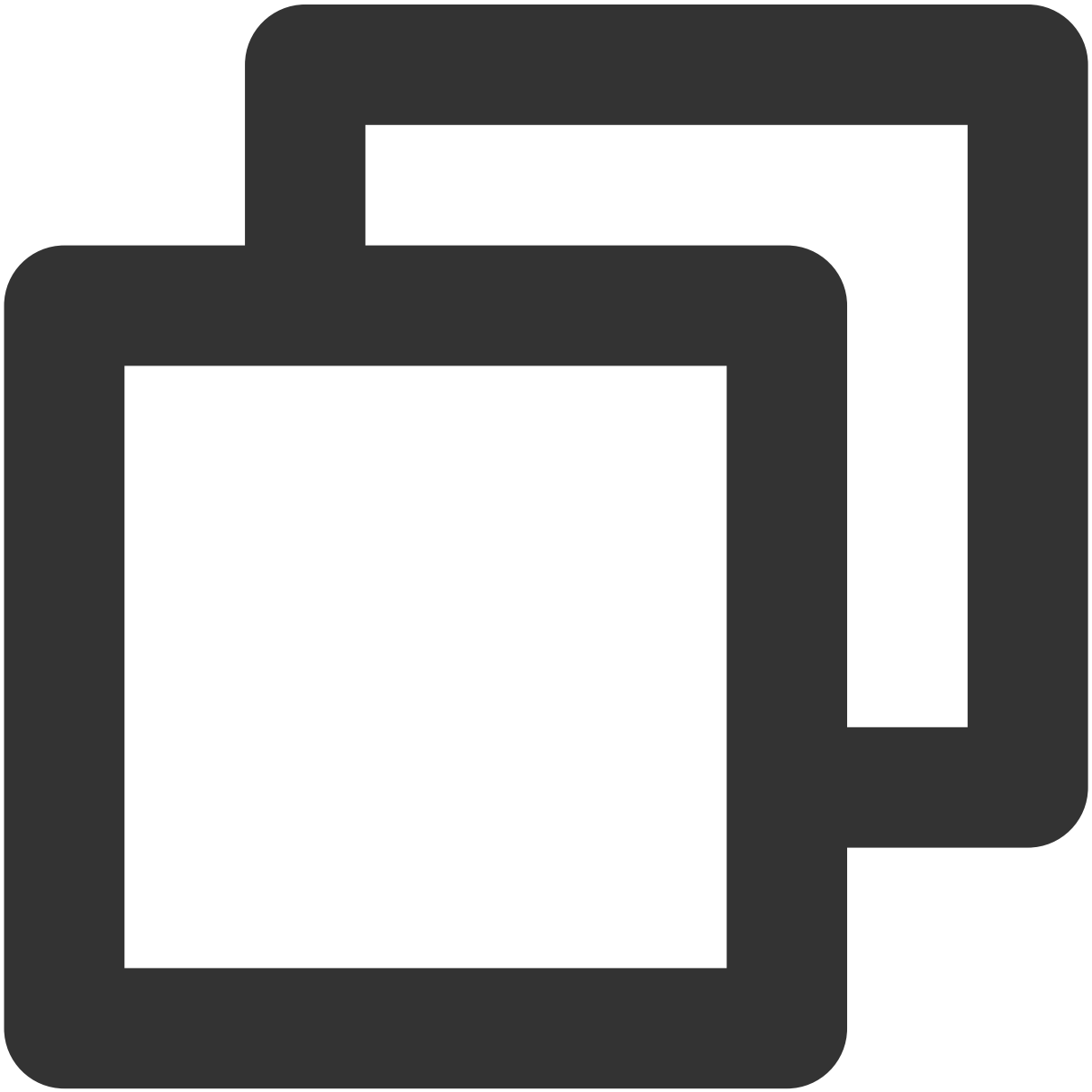
注意：

当然，您可以在进入房间（enterRoom）后再启动摄像头预览和麦克风采集，不过在直播场景下，我们需要先给主播一个测试麦克风和调整美颜的时间，所以更常见的做法是先启动摄像头和麦克风再进入房间。

Android

iOS&Mac

Windows



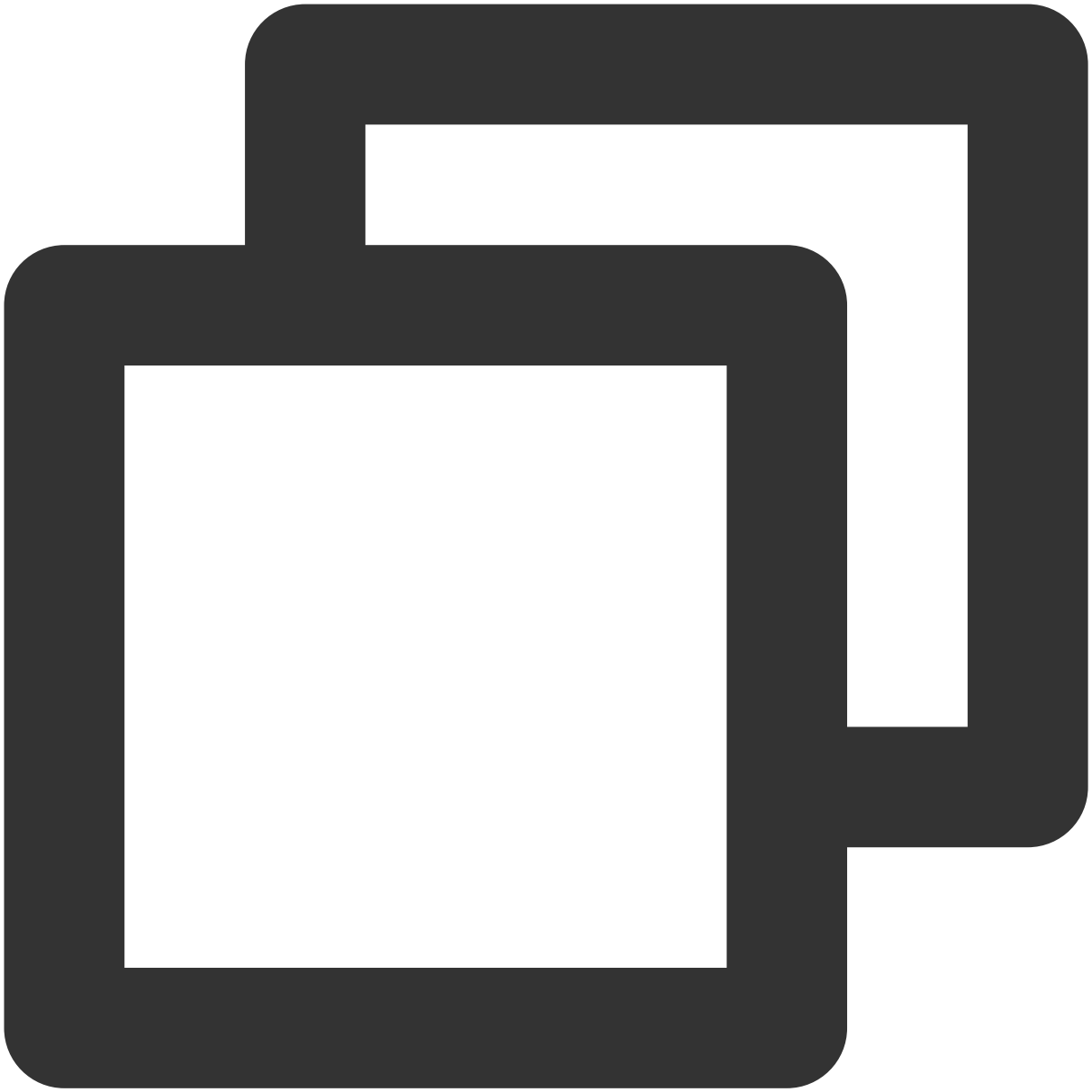
```
mCloud = TRTCCloud.sharedInstance(getApplicationContext());
```

```
mCloud.setListener(mTRTCCloudListener);

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
TRTCCloudDef.TRTCParams param = new TRTCCloudDef.TRTCParams();
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.userId = "denny";       // Please replace with your own userid
params.roomId = 123321;        // Please replace with your own room number
params.userSig = "xxx";        // Please replace with your own userSig
params.role = TRTCCloudDef.TRTCRoleAnchor;

// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
mCloud.enterRoom(param, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

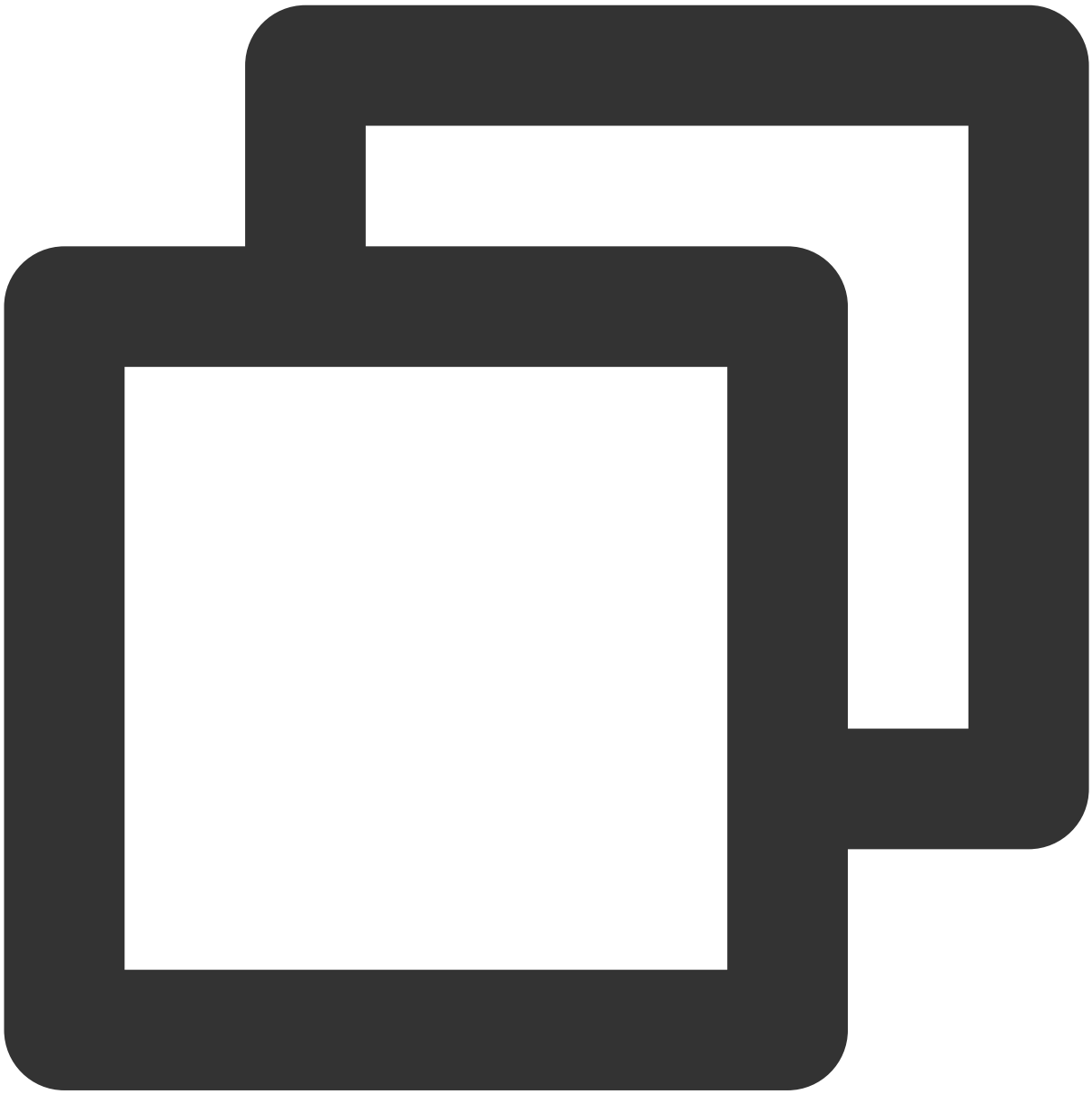
版权所有：腾讯云计算（北京）有限责任公司



```
self.trtcCloud = [TRTCCloud sharedInstance];
self.trtcCloud.delegate = self;

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
TRTCParams *params = [[TRTCParams alloc] init];
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.roomId = 123321; // Please replace with your own room number
params.userId = @"denny";   // Please replace with your own userid
params.userSig = @"xxx"; // Please replace with your own userSig
params.role = TRTCRoleAnchor;
```

```
// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
[self.trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
```



```
trtc_cloud_ = getTRTCShareInstance();

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
liteav::TRTCParams params;
```

```
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.userId = "denny";       // Please replace with your own userid
params.roomId = 123321;        // Please replace with your own room number
params.userSig = "xxx";        // Please replace with your own userSig
params.role = TRTCCloudDef.TRTCRoleAnchor;

// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
trtc_cloud_->enterRoom(params, liteav::TRTCAppSceneLIVE);
```

步骤5：角色和角色的切换

TRTC 中的“角色”

在“视频通话”（TRTC_APP_SCENE_VIDEOCALL ） 和 “语音通话”（TRTC_APP_SCENE_AUDIOCALL）这两个场景中，您无需要在进入房间时设置角色，因为在这两个模式下每个用户默认都是主播（Anchor）。

在“视频直播”（TRTC_APP_SCENE_LIVE） 和 “语音直播”（TRTC_APP_SCENE_VOICE_CHATROOM）这两个场景中，每个用户在进入房间时都需要指定自己的“角色”，要么是“主播（Anchor）”，要么是“观众（Audience）”。

角色的切换

在 TRTC 中，只有“主播（Anchor）”才有权限发布音视频流，“观众（Audience）”是不能发布音视频流的。

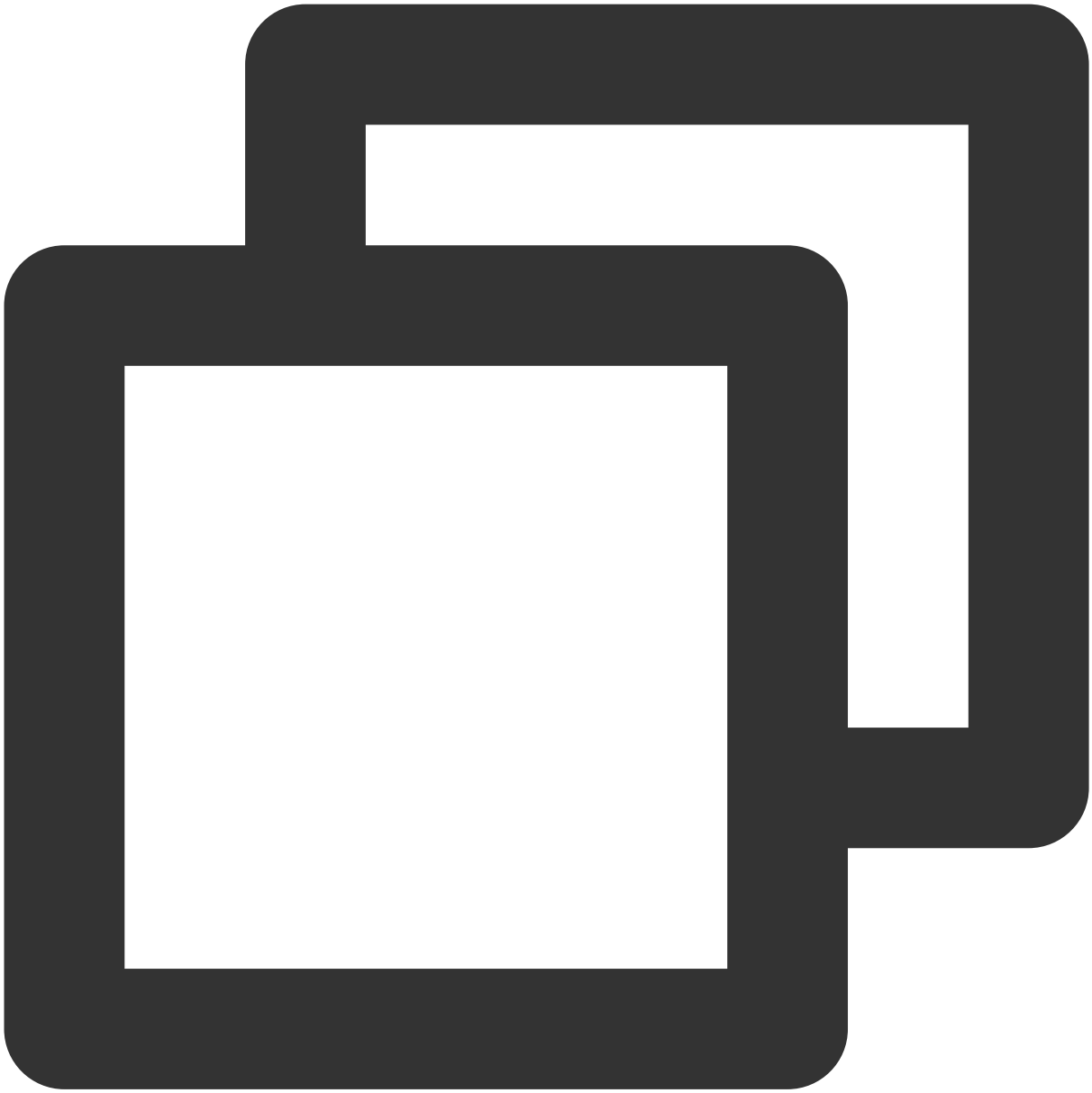
因此，如果您在进房时使用的角色是“观众（Audience）”，需要先调用 **switchRole** 接口将角色切换成“主播（Anchor）”，才能发布音视频流（也就是俗称的“上麦”）。

Android

iOS

Mac

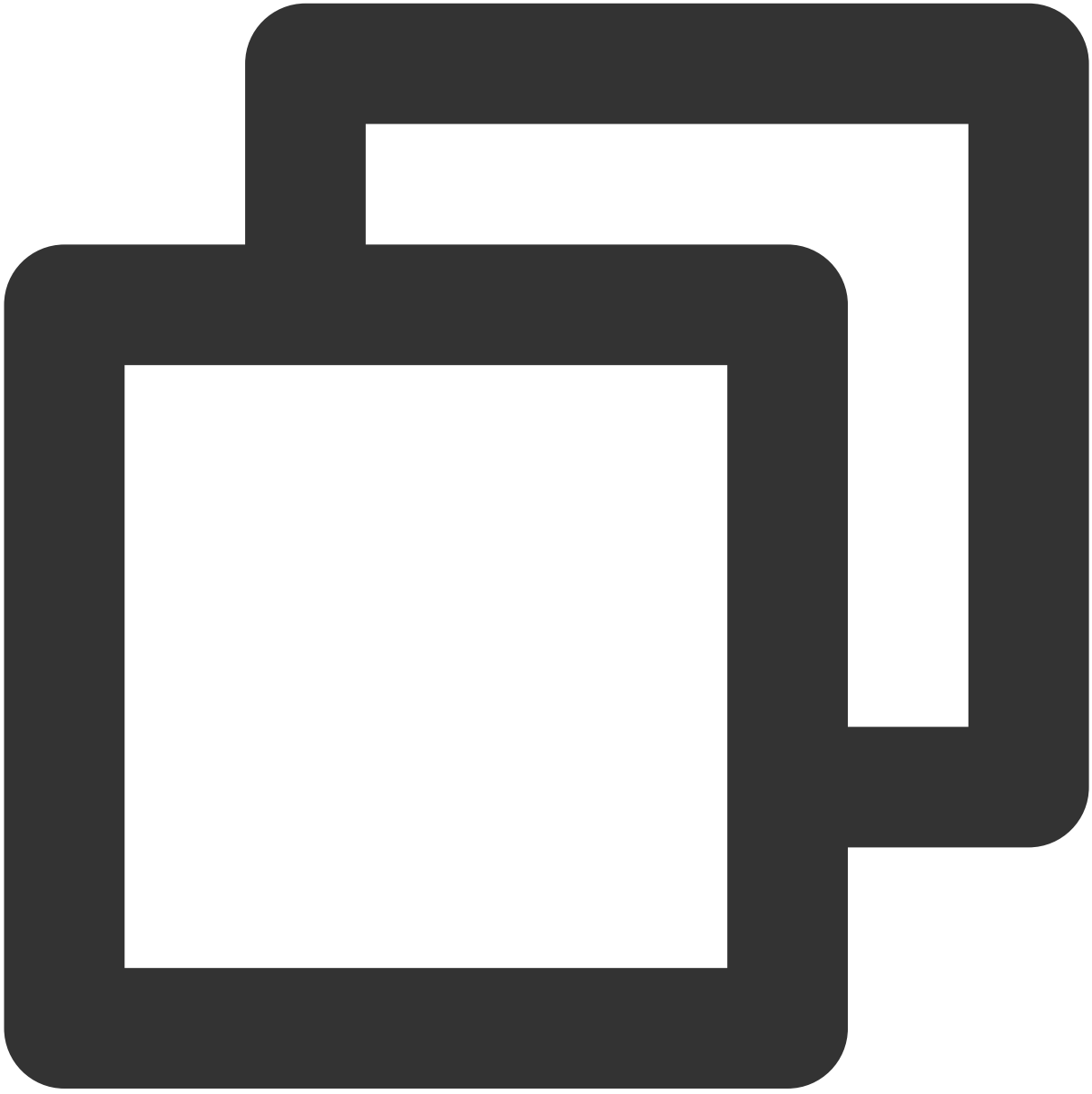
Windows



```
// 如果您当前的角色是观众（Audience），需要先调用 switchRole 切换到主播（Anchor）
// If your current role is 'audience', you need to call switchRole to switch to 'an
mCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
mCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
mCloud.startLocalPreview(true, cameraVide);

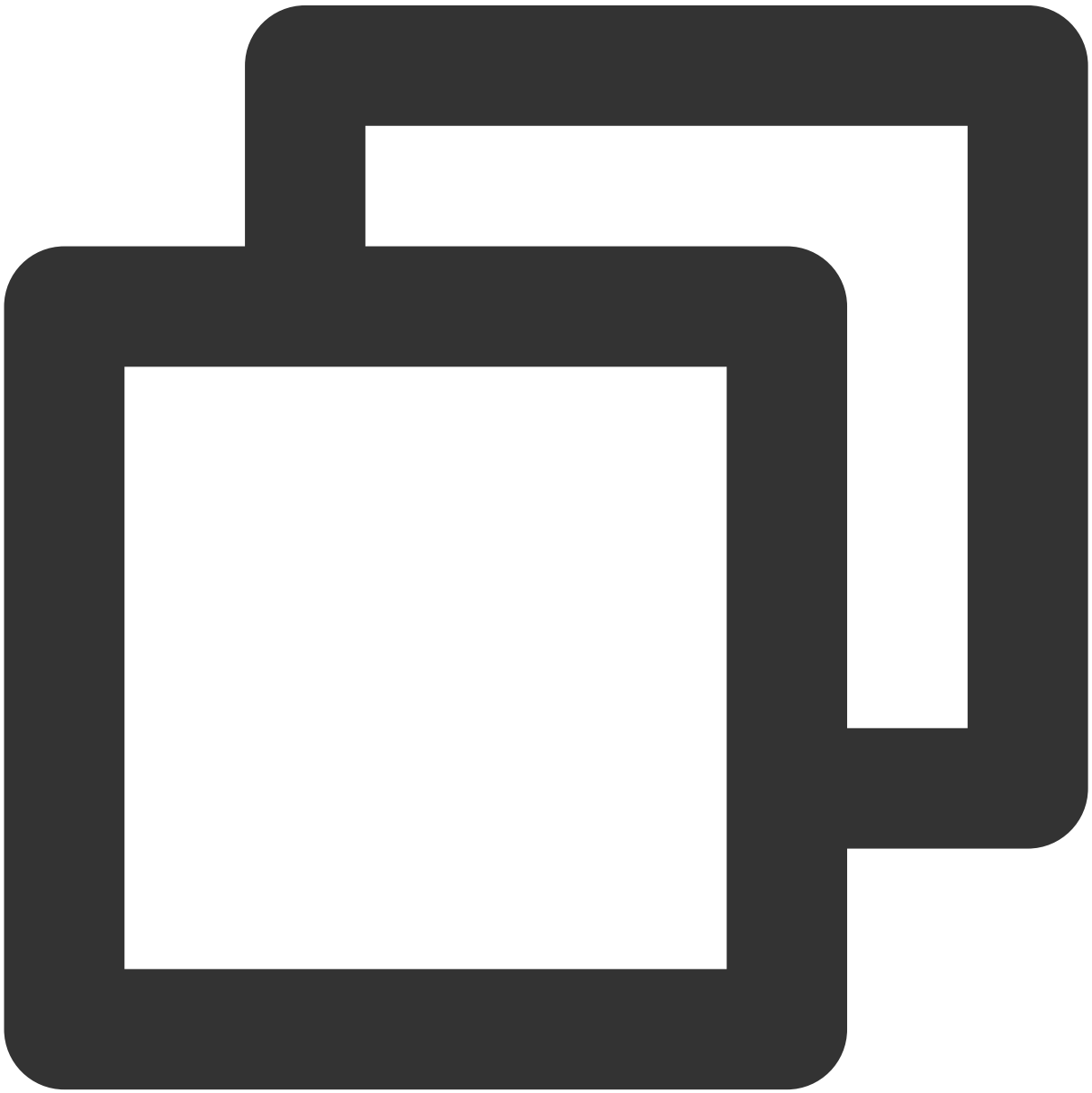
// 如果切换角色失败，onSwitchRole 回调的错误码便不是 0
// If switching operation failed, the error code of the 'onSwitchRole' is not zero
@Override
public void onSwitchRole(final int errCode, final String errMsg) {
    if (errCode != 0) {
```

```
        Log.d(TAG, "Switching operation failed...");
    }
}
```



```
self.trtcCloud = [TRTCCloud sharedInstance];
// 如果您当前的角色是观众（Audience），需要先调用 switchRole 切换到主播（Anchor）
// If your current role is 'audience', you need to call switchRole to switch to 'an
[self.trtcCloud switchRole:TRTCRoleAnchor];
[self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
[self.trtcCloud startLocalPreview:YES view:localCameraVideoView];
```

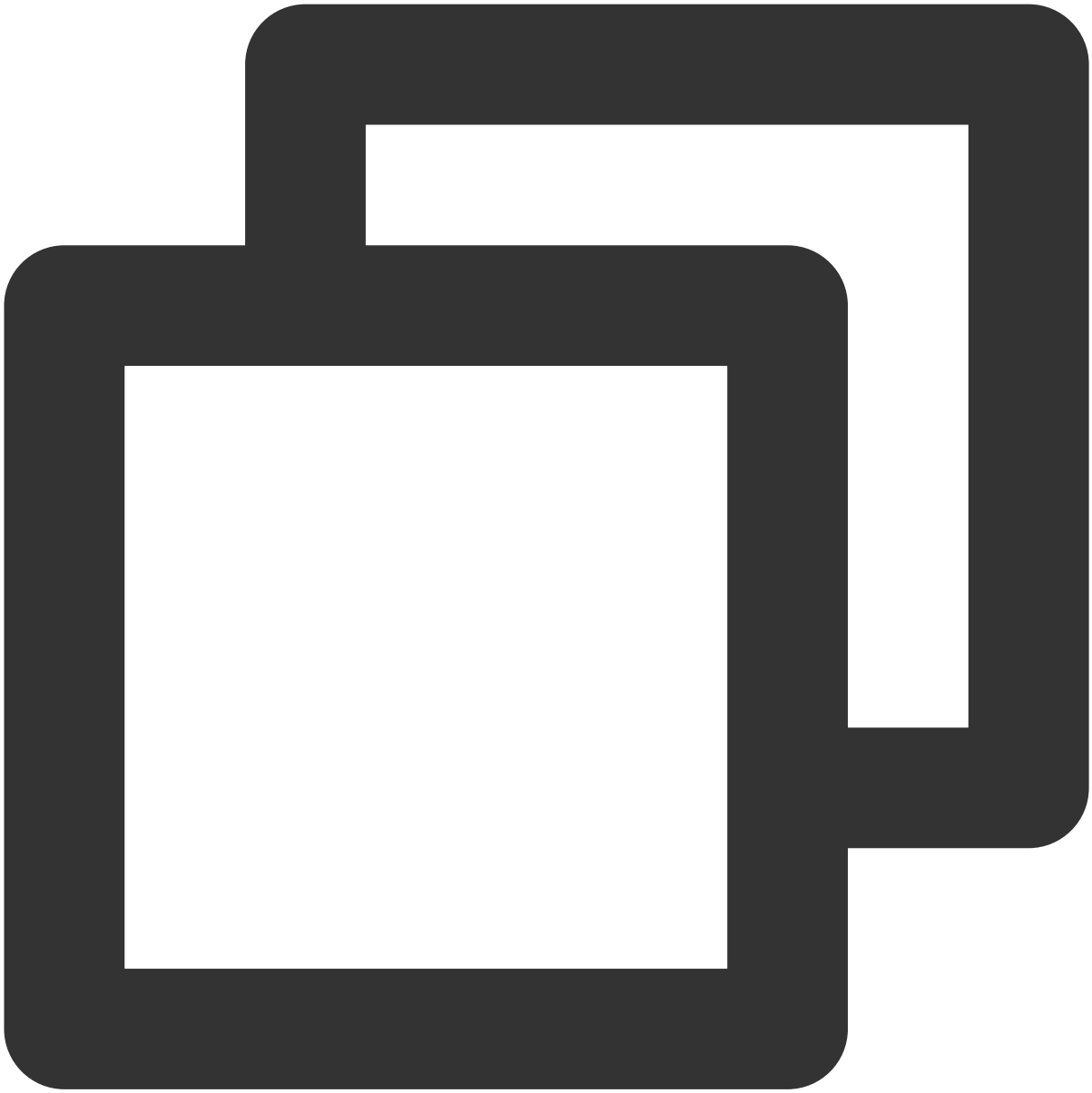
```
// 如果切换角色失败，onSwitchRole 回调的错误码便不是 0
// If switching operation failed, the error code of the 'onSwitchRole' is not zero
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg {
    if (errCode != 0) {
        NSLog(@"Switching operation failed... ");
    }
}
```



```
self.trtcCloud = [TRTCCloud sharedInstance];
```

```
// 如果您当前的角色是观众（Audience），需要先调用 switchRole 切换到主播（Anchor）
// If your current role is 'audience', you need to call switchRole to switch to 'an
[self.trtcCloud switchRole:TRTCRoleAnchor];
[self.trtcCloud startLocalAudio:TRTCAudioQualityDefault];
[self.trtcCloud startLocalPreview:localCameraVideoView];

// 如果切换角色失败，onSwitchRole 回调的错误码便不是 0
// If switching operation failed, the error code of the 'onSwitchRole' is not zero
- (void)onSwitchRole:(TXLiteAVError)errCode errMsg:(nullable NSString *)errMsg {
    if (errCode != 0) {
        NSLog(@"Switching operation failed... ");
    }
}
```



```
// 如果您当前的角色是观众（Audience），需要先调用 switchRole 切换到主播（Anchor）
// If your current role is 'audience', you need to call switchRole to switch to 'an
trtc_cloud_->switchRole(liteav::TRTCRoleAnchor);
trtc_cloud_->startLocalAudio(TRTCAudioQualityDefault);
trtc_cloud_->startLocalPreview(hWnd);

// 切换角色失败，onSwitchRole 回调的错误码便不是 ERR_NULL（即0）
// If switching operation failed, the error code of the 'onSwitchRole' is not zero
void onSwitchRole(TXLiteAVError errCode, const char* errMsg) {
    if (errCode != ERR_NULL) {
        printf("Switching operation failed...");
```

```
        }
    }
```

注意： 如果房间中已有的主播已经太多，会导致 switchRole 角色切换失败，并通过 TRTC 的 onSwitchRole 将错误码回调通知您。所以，当您不再需要发布音视频流（也就是俗称的“下麦”）时，就需要您再次调用 switchRole 并切换为“观众（Audience）”。

说明：

您可以能有一个疑问，既然只要主播才能发布音视频流，那我是不是可以让每一个用户都用主播的角色进入房间呢？答案肯定是不行的，原因需要看进阶指引 一个房间中同时最多可以有多少路音频和视频？

进阶指引

1. 一个房间中同时最多可以有多少路音频和视频？

一个 TRTC 房间中最多允许同时有 **50** 路音视频流，超出的音视频流会按照“先到先得”的原则被放弃掉。

在绝大多数场景下，小到两个人之间的视频通话，大到几万人同时观看的在线直播，50 路的音视频流都能满足应用场景的需求，但前提是您要做好**角色管理**。

所谓的“角色管理”就是您需要为进入房间的用户分配角色。

如果一个用户本身就是直播场景中的“主播”，或者是在线教育场景中的“老师”，抑或是在线会议场景中的“主持人”，都可以分配给“主播（Anchor）”这个角色。

如果一个用户本身就是直播场景中的“观众”，或者是在线教育场景中的“学生”，抑或是在线会议场景中的“旁听者”，应当分配为“观众（Audience）”这个角色，否则他们庞大的人数会瞬间将主播的人数限制“挤爆”。

只有当“观众”需要发布音视频流（“上麦”）时，才需要通过 switchRole 切换成“主播”角色，等不再需要发布音视频流（“下麦”）时，则需要立刻切回观众角色。

通过合理得角色管理，您会发现房间中同时需要发布音视频流的“主播”是通常不会超过 50 路的，否则整个房间中就会“乱糟糟”，毕竟同时说话的人声一旦超过 6 路，普通人就很难从声音中辨识出当前究竟是谁在说话了。

Web

最近更新时间：2023-08-22 17:00:37

本文档主要介绍主播如何发布自己的音视频流，所谓“发布”，也就是打开麦克风和摄像头，让自己的声音和视频能够被房间中其他用户听到和看到的意思。

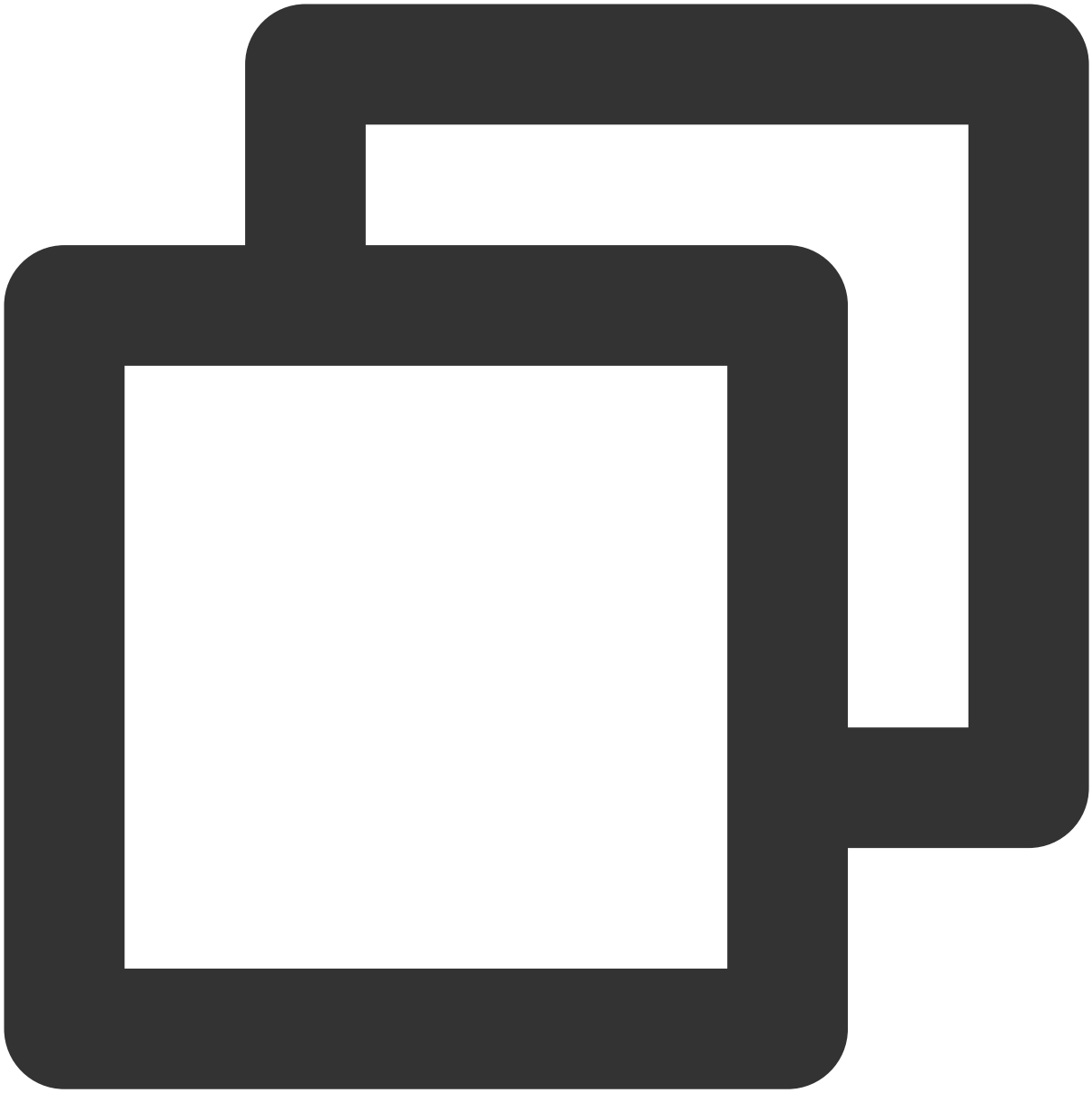


步骤1：完成创建 trtc 实例

可以参考文档 进入房间 - 步骤 1，创建 trtc 实例，为接下来开启麦克风和摄像头做准备。

步骤 2：开启摄像头

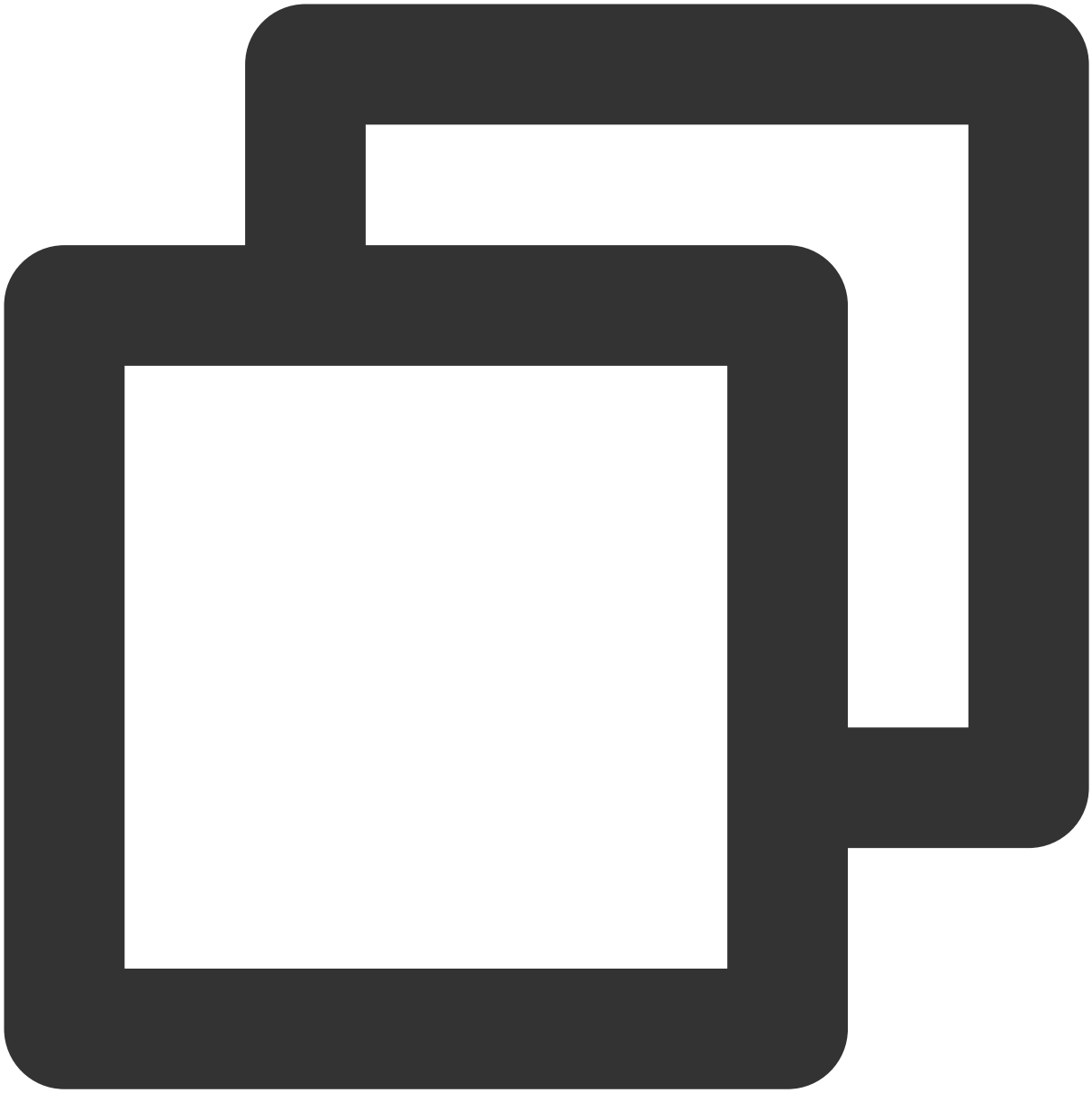
使用 trtc.startLocalVideo() 方法开启摄像头，并发布到房间。



```
// 为了预览摄像头画面，您需在 DOM 中放置一个 HTMLElement，可以是一个 div 标签，假设其 id 为 lo
const view = 'local-video';
await trtc.startLocalVideo({ view });
```

步骤 3：开启麦克风

使用 trtc.startLocalAudio() 方法开启麦克风，并发布到房间。



```
await trtc.startLocalAudio();
```

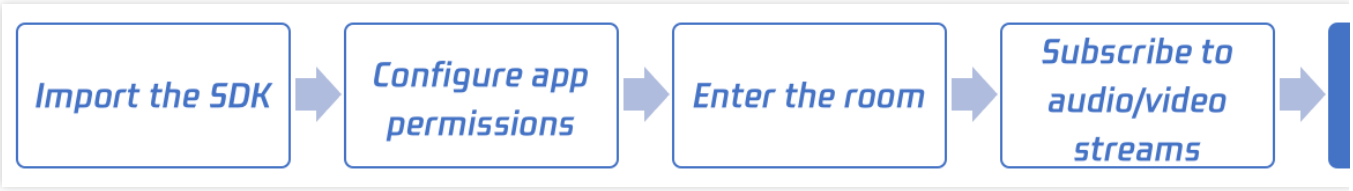
步骤 4：关闭摄像头、麦克风

使用 trtc.stopLocalVideotrtc.stopLocalAudio 关闭摄像头和麦克风。

Electron

最近更新时间：2024-06-07 11:20:49

本文档主要介绍主播如何发布自己的音视频流，所谓“发布”，也就是打开麦克风和摄像头，让自己的声音和视频能够被房间中其他用户听到和看到的意思。



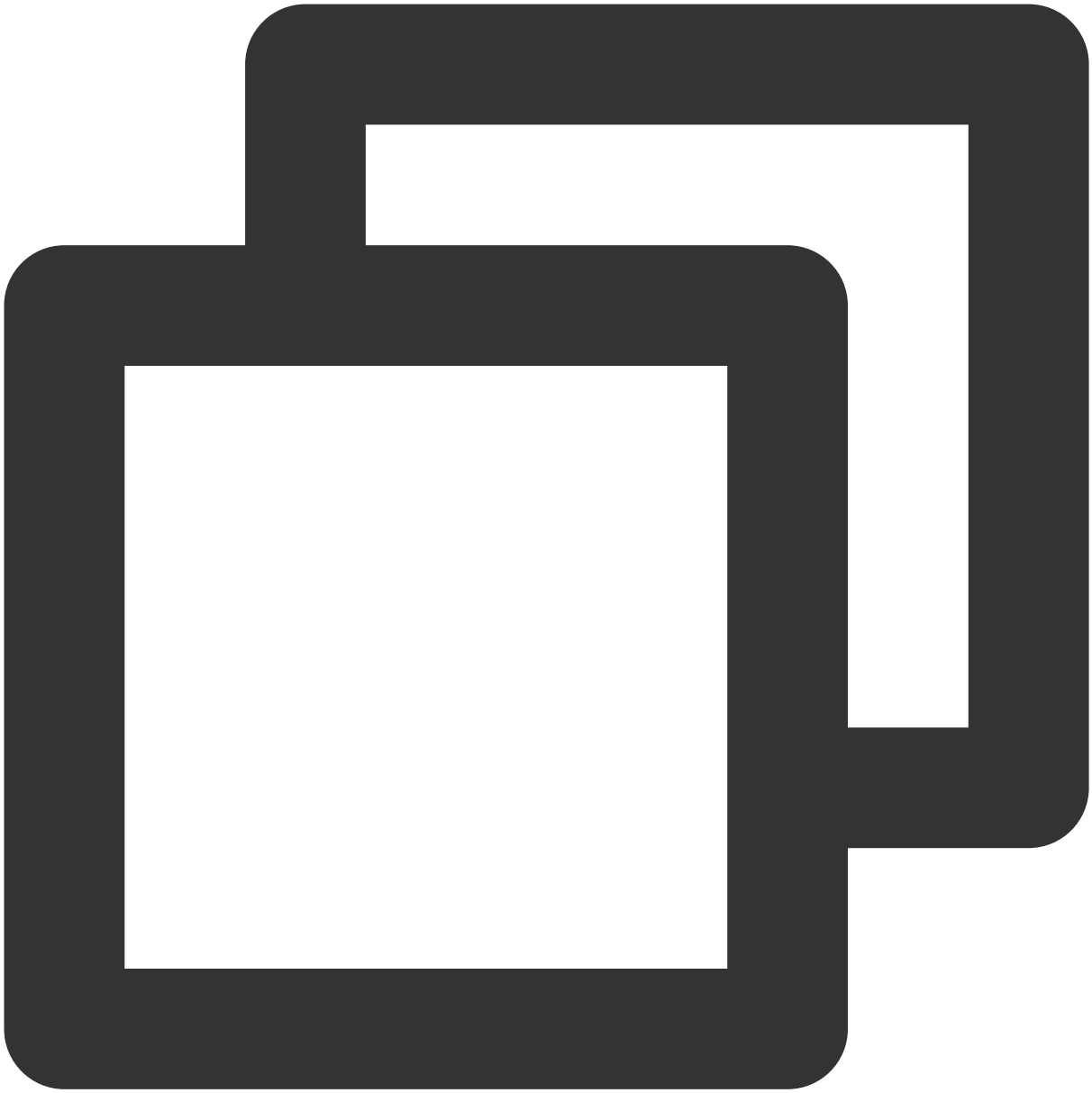
调用指引

步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和配置。

步骤2：打开摄像头预览

您可以调用 **startLocalPreview** 接口打开摄像头预览，此时 SDK 会向系统申请摄像头的使用权限，需要用户授权通过后才会开启摄像头的采集流程。
如果您希望设置本地画面的渲染参数，可以通过调用 **setLocalRenderParams** 接口来设置本地预览的渲染参数。为防止先开启预览再设置预览参数会出现画面跳动，如果您需要设置预览参数，推荐在开启预览之前调用。



```
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
import TRTCCloud, {
    TRTCRenderParams, TRTCVideoRotation,
    TRTCVideoFillMode, TRTCVideoMirrorType
} from 'trtc-electron-sdk';

const param = new TRTCRenderParams(
    TRTCVideoRotation.TRTCVideoRotation0,
    TRTCVideoFillMode.TRTCVideoFillMode_Fill,
    TRTCVideoMirrorType.TRTCVideoMirrorType_Auto
);
```

```
const rtcCloud = new TRTCCloud();
rtcCloud.setLocalRenderParams(param);
const cameraVideoDom = document.querySelector('.camera-dom');
rtcCloud.startLocalPreview(cameraVideoDom);
```

步骤3：打开麦克风采集

您可以调用 **startLocalAudio** 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式。虽然这个参数的名字叫做 quality，但并不是说质量越高越高，不同的业务场景有最适合的参数选择（这个参数更准确的含义是 scene）。

SPEECH

该模式下的 SDK 音频模块会专注于提炼语音信号，尽最大限度的过滤周围的环境噪音，同时该模式下的音频数据也会获得最好的差质量网络的抵抗能力，因此该模式特别适合于“视频通话”和“在线会议”等侧重于语音沟通的场景。

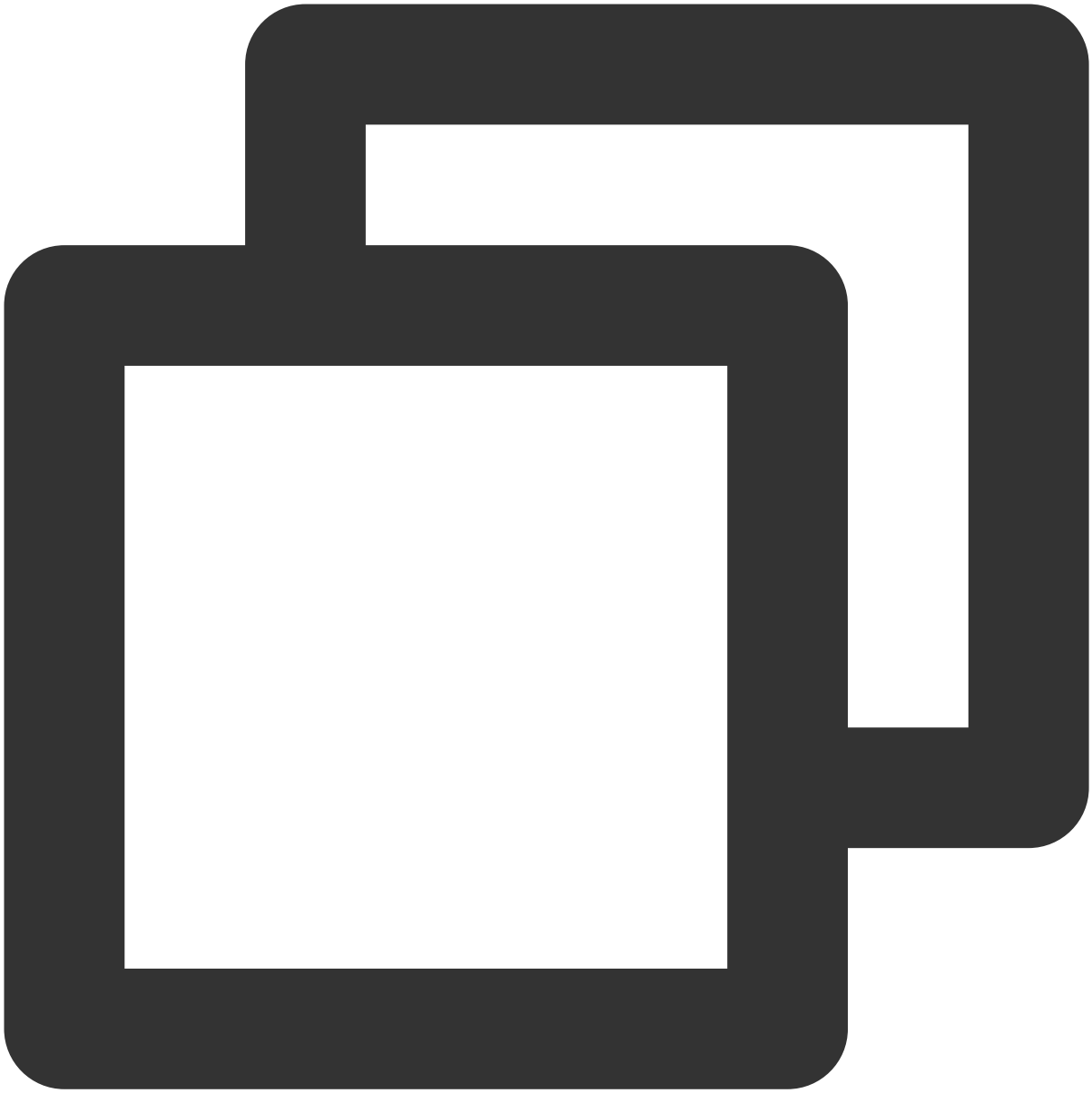
MUSIC

该模式下的 SDK 会采用很高的音频处理带宽以及立体式模式，在最大限度地提升采集质量的同时也会将音频的 DSP 处理模块调节到最弱的级别，从而最大限度地保证音质。因此该模式适合“音乐直播”场景，尤其适合主播采用专业的声卡进行音乐直播的场景。

DEFAULT

该模式下的 SDK 会启用智能识别算法来识别当前环境，并针对性地选择最佳的处理模式。不过再好的识别算法也总是有不准确的时候，如果您非常清楚自己的产品定位，更推荐您在专注语音通信的 SPEECH 和专注音乐音质的 MUSIC 之间二选一。

版权所有：腾讯云计算（北京）有限责任公司



```
import { TRTCAudioQuality } from 'trtc-electron-sdk';
// 开启麦克风采集，并设置当前场景为：语音模式（高噪声抑制能力、强弱网络抗性）
rtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualitySpeech);

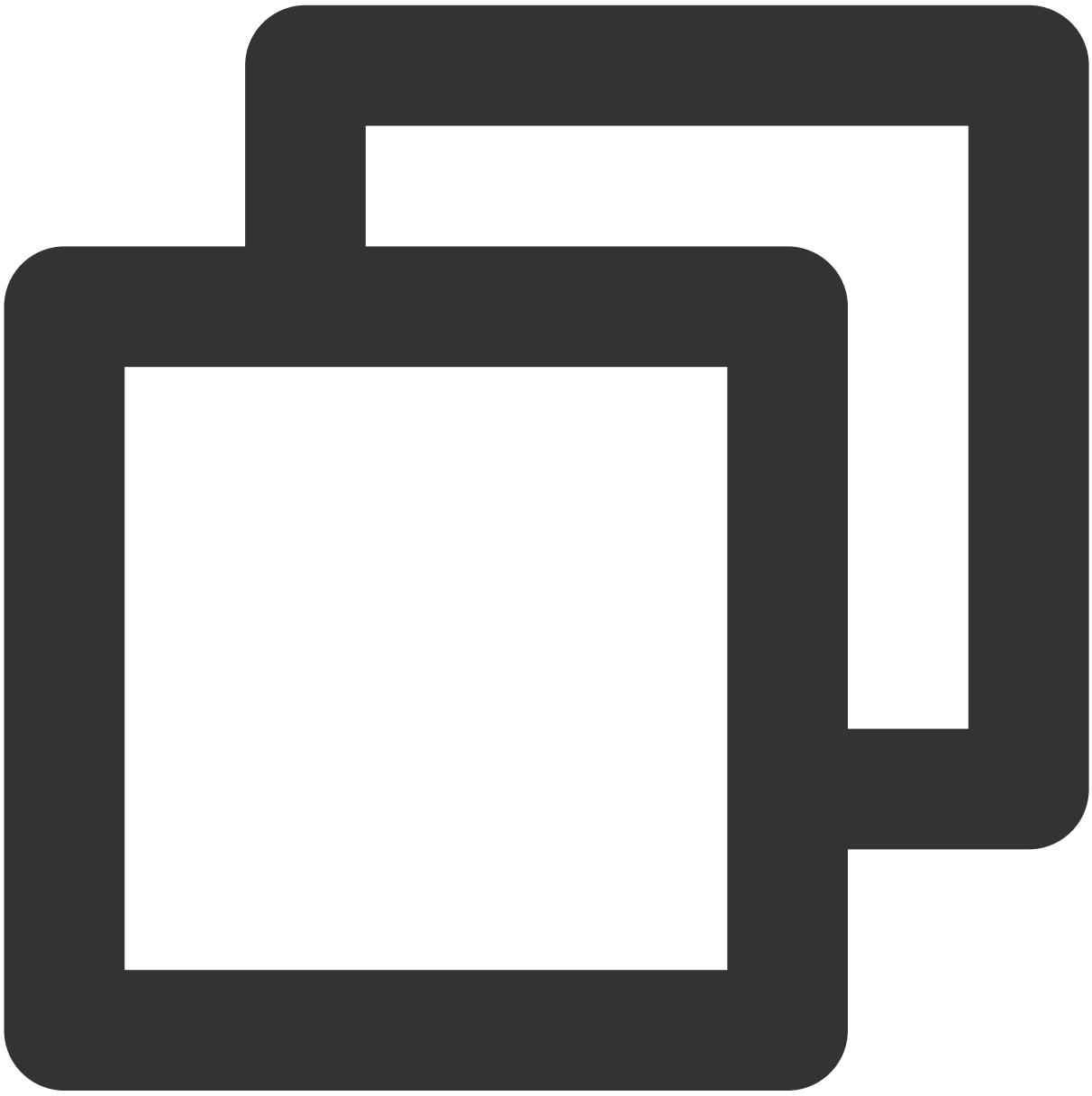
// 开启麦克风采集，并设置当前场景为：音乐模式（高保真采集、低音质损失，推荐配合专业声卡使用）
rtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualityMusic);
```

步骤4：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间。一旦进入房间后，SDK 便会开始向房间中的其他用户发布自己的音频流。

注意：

当然，您可以在进入房间（enterRoom）后再启动摄像头预览和麦克风采集，不过在直播场景下，我们需要先给主播一个测试麦克风和调整美颜的时间，所以更常见的做法是先启动摄像头和麦克风再进入房间。



```
import { TRTCParams, TRTCRoleType, TRTCAppScene } from 'trtc-electron-sdk';

// 组装 TRTC 进房参数，请将 TRTCParams 中的各字段都替换成您自己的参数
// Please replace each field in TRTCParams with your own parameters
```

```
const param = new TRTCParams();
params.sdkAppId = 1400000123;  // Please replace with your own SDKAppID
params.userId = "denny";       // Please replace with your own userid
params.roomId = 123321;        // Please replace with your own room number
params.userSig = "xxx";        // Please replace with your own userSig
params.role = TRTCRoleType.TRTCRoleAnchor;

// 如果您的场景是“在线直播”，请将应用场景设置为 TRTC_APP_SCENE_LIVE
// If your application scenario is a video call between several people, please use
rtcCloud.enterRoom(param, TRTCAppScene.TRTCAppSceneLIVE);
```

Flutter

最近更新时间：2024-02-02 18:52:06

本文档主要介绍主播如何发布自己的音视频流，所谓“发布”，也就是打开麦克风和摄像头，让自己的声音和视频能够被房间中其他用户听到和看到的意思。

调用指引

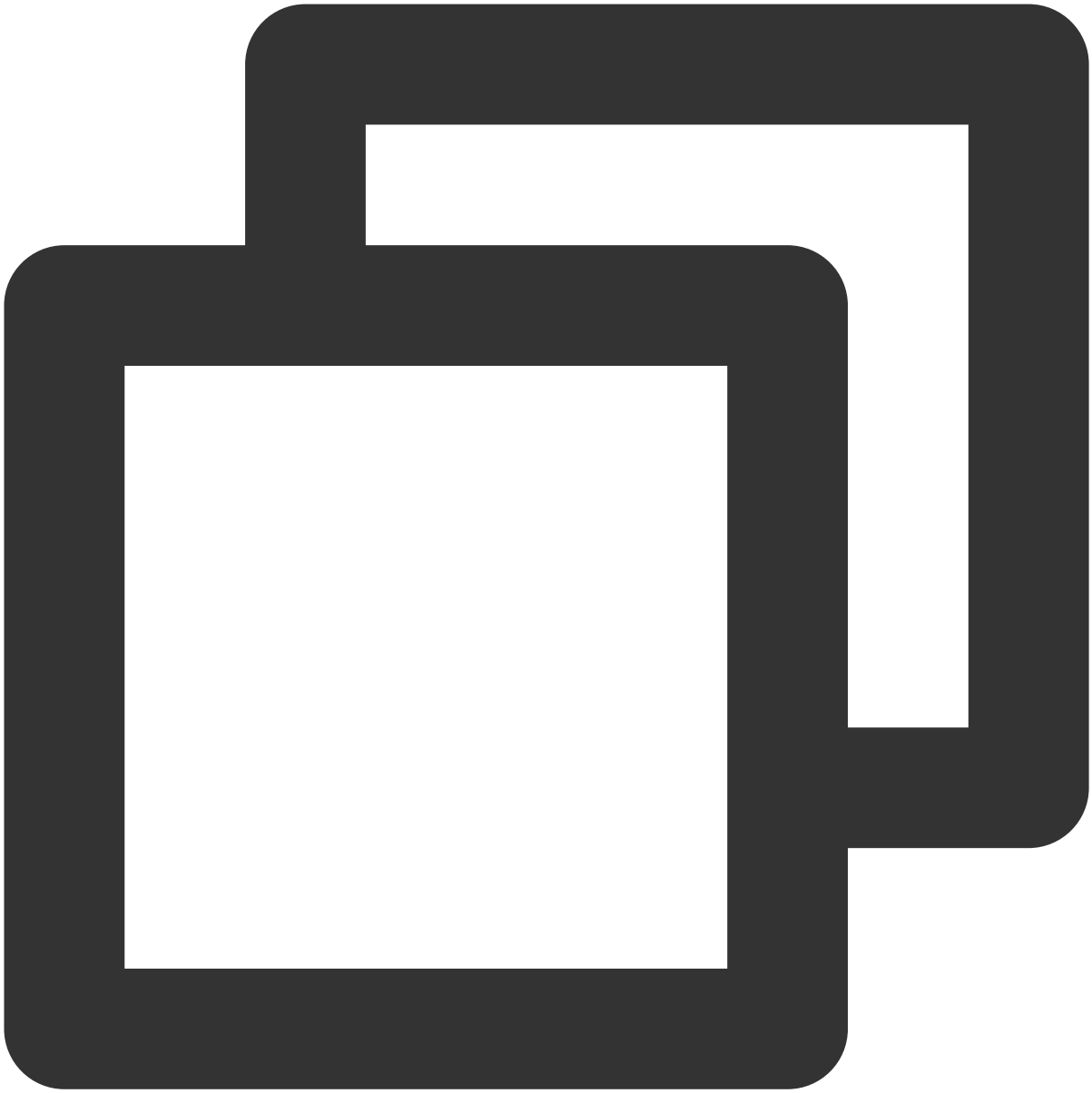
步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。

步骤2：打开摄像头预览

您可以调用 **startLocalPreview** 接口打开摄像头预览，此时 SDK 会向系统申请摄像头的使用权限，需要用户授权通过后才会开启摄像头的采集流程。
如果您希望设置本地画面的渲染参数，可以通过调用 **setLocalRenderParams** 接口来设置本地预览的渲染参数。为防止先开启预览再设置预览参数会出现画面跳动，如果您需要设置预览参数，推荐在开启预览之前调用。
如果您希望控制摄像头的各种控制参数，可以用过调用 **TXDeviceManager** 接口来完成“切换前后摄像头”、“设置对焦模式”、“打开、关闭闪光灯”等一系列操作。
如果您希望调节美颜效果和画面质量，我们会在 设定画面质量 中进行详细介绍。

版权所有：腾讯云计算（北京）有限责任公司



```
// 设置本地画面的预览模式：开启左右镜像，设置画面为填充模式
trtcCloud.setLocalRenderParams(TRTCRenderParams(
    fillMode: TRTCCloudDef.TRTC_VIDEO_RENDER_MODE_FILL
    mirrorType: TRTCCloudDef.TRTC_VIDEO_MIRROR_TYPE_ENABLE);

// 启动本地摄像头的预览（viewId 是 TRTCCloudVideoView 创建过程中 onViewCreated 调用时给予的
trtcCloud.startLocalPreview(isFrontCamera, viewId);

// 通过 TXDeviceManager 开启自动对焦并将闪光灯打开
bool? isAutoFocusEnabled = await manager.isAutoFocusEnabled();
if (isAutoFocusEnabled ?? false) {
```

```
    manager.enableCameraAutoFocus(true);
}
manager.enableCameraTorch(true);
```

步骤3：打开麦克风采集

您可以调用 **startLocalAudio** 来开启麦克风采集，该接口需要您通过 `quality` 参数确定采集模式。虽然这个参数的名字叫做 quality，但并不是说质量越高越好，不同的业务场景有最适合的参数选择（这个参数更准确的含义是 scene）。

TRTC_AUDIO_QUALITY_SPEECH

该模式下的 SDK 音频模块会专注于提炼语音信号，尽最大限度的过滤周围的环境噪音，同时该模式下的音频数据也会获得最好的差质量网络的抵抗能力，因此该模式特别适合于“视频通话”和“在线会议”等侧重于语音沟通的场景。

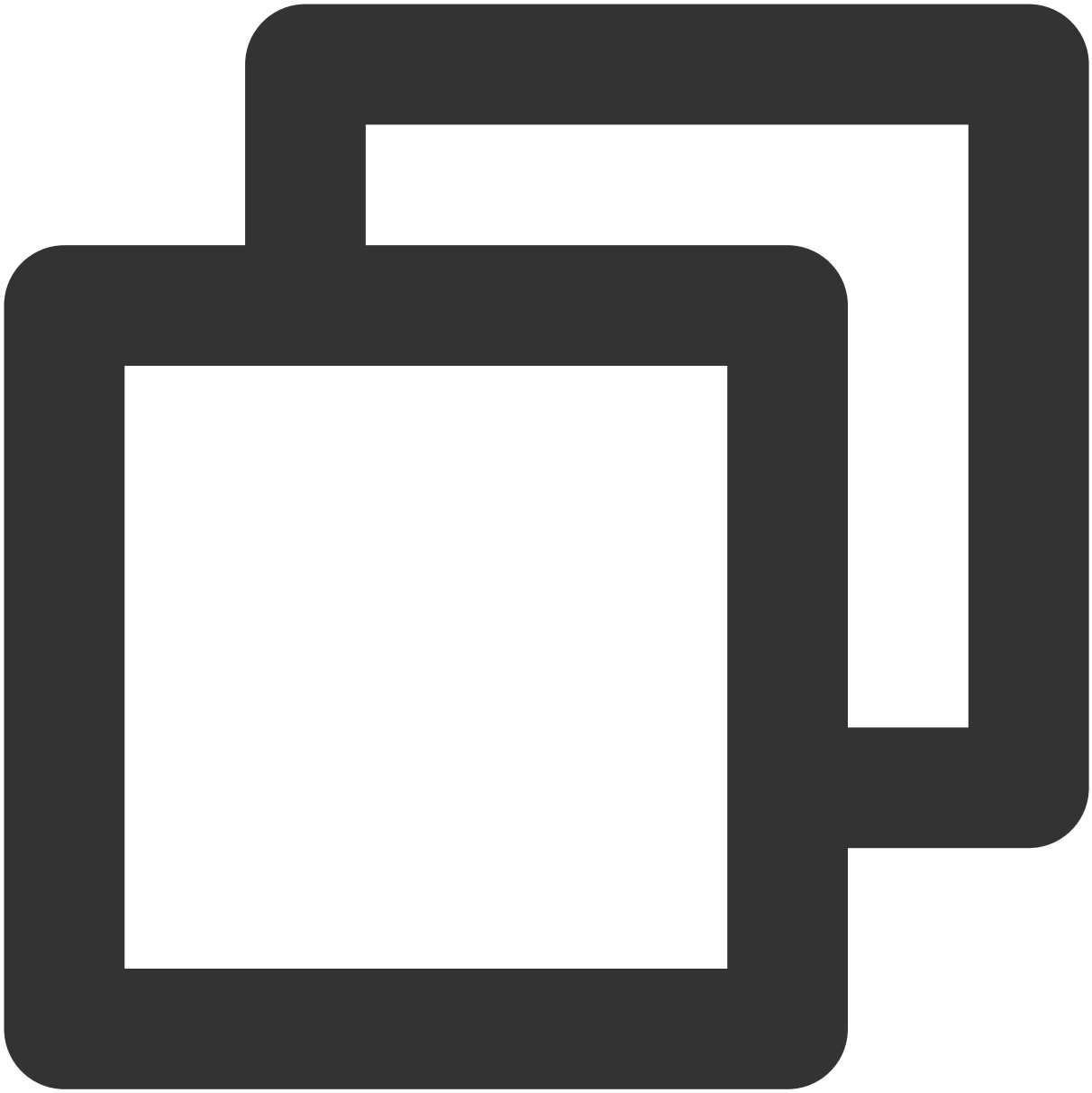
TRTC_AUDIO_QUALITY_MUSIC

该模式下的 SDK 会采用很高的音频处理带宽以及立体式模式，在最大限度地提升采集质量的同时也会将音频的 DSP 处理模块调节到最弱的级别，从而最大限度地保证音质。因此该模式适合“音乐直播”场景，尤其适合主播采用专业的声卡进行音乐直播的场景。

TRTC_AUDIO_QUALITY_DEFAULT

该模式下的 SDK 会启用智能识别算法来识别当前环境，并针对性地选择最佳的处理模式。不过再好的识别算法也总是有不准确的时候，如果您非常清楚自己的产品定位，更推荐您在专注语音通信的 SPEECH 和专注音乐音质的 MUSIC 之间二选一。

版权所有：腾讯云计算（北京）有限责任公司



```
// 开启麦克风采集，并设置当前场景为：语音模式（高噪声抑制能力、强弱网络抗性）
trtcCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_SPEECH );

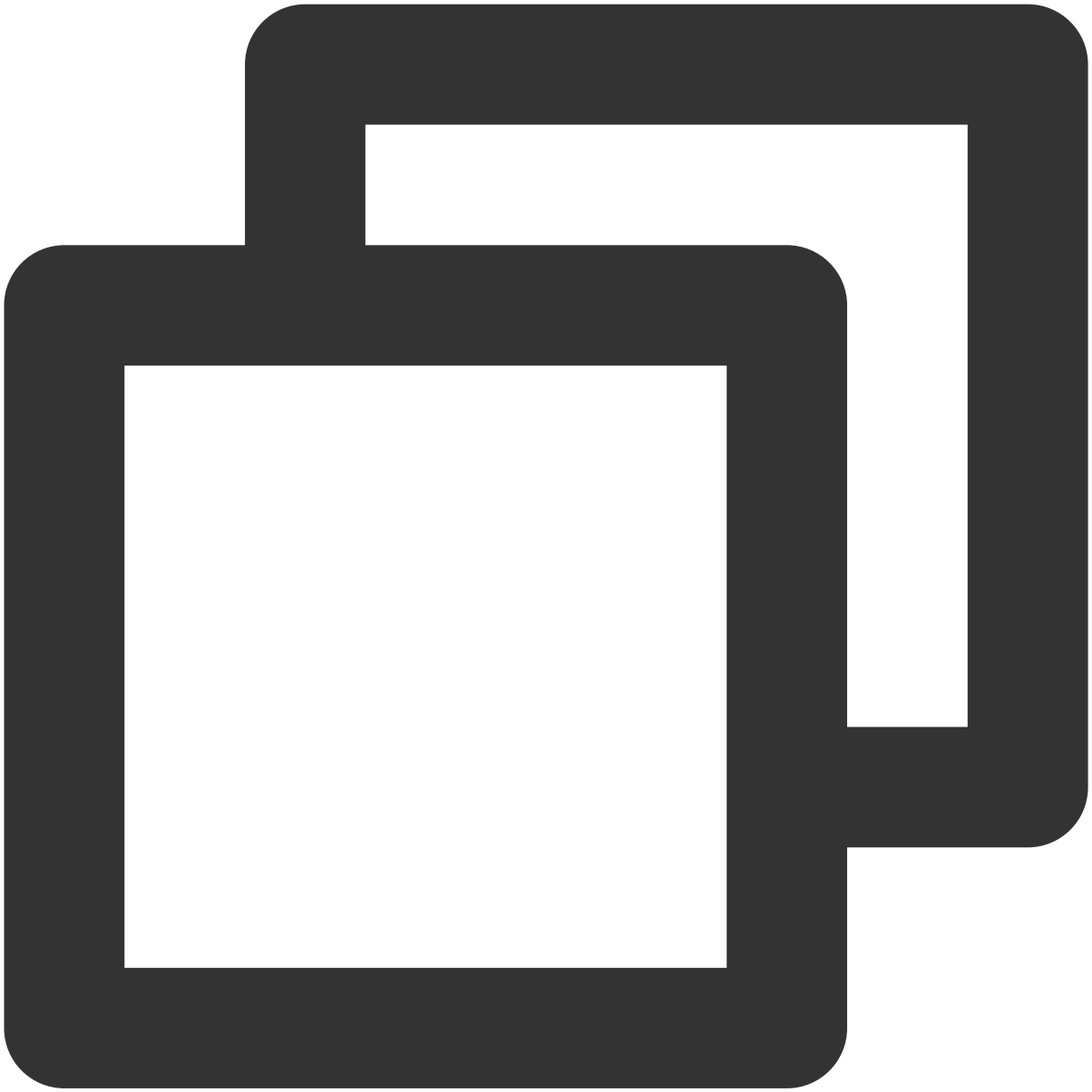
// 开启麦克风采集，并设置当前场景为：音乐模式（高保真采集、低音质损失，推荐配合专业声卡使用）
trtcCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
```

步骤4：进入 TRTC 房间

参考文档 进入房间 让当前用户进入 TRTC 房间。一旦进入房间后，SDK 便会开始向房间中的其他用户发布自己的音频流。

注意：

当然，您可以在进入房间（enterRoom）后再启动摄像头预览和麦克风采集，不过在直播场景下，我们需要先给主播一个测试麦克风和调整美颜的时间，所以更常见的做法是先启动摄像头和麦克风再进入房间。



```
// 创建 TRTCCloud 单例
trtcCloud = (await TRTCCloud.sharedInstance())!;
// 注册 TRTC 事件回调
trtcCloud.registerListener(onRtcListener);

enterRoom() async {
  try {
```

```
    userInfo['userSig'] =
        await GenerateTestUserSig.genTestSig(userInfo['userId']);
    meetModel.setUserInfo(userInfo);
  } catch (err) {
    userInfo['userSig'] = '';
    print(err);
  }
  // 如果您的场景为“互动视频直播”，请将场景设置为 TRTC_APP_SCENE_LIVE，并在TRTCParams中为rol
  await trtcCloud.enterRoom(
      TRTCParams(
          sdkAppId: GenerateTestUserSig.sdkAppId,
          userId: userInfo['userId'],
          userSig: userInfo['userSig'],
          role: TRTCCloudDef.TRTCRoleAnchor,
          roomId: meetId!),
      TRTCCloudDef.TRTC_APP_SCENE_LIVE);
}
```

步骤5：角色和角色的切换

TRTC 中的“角色”

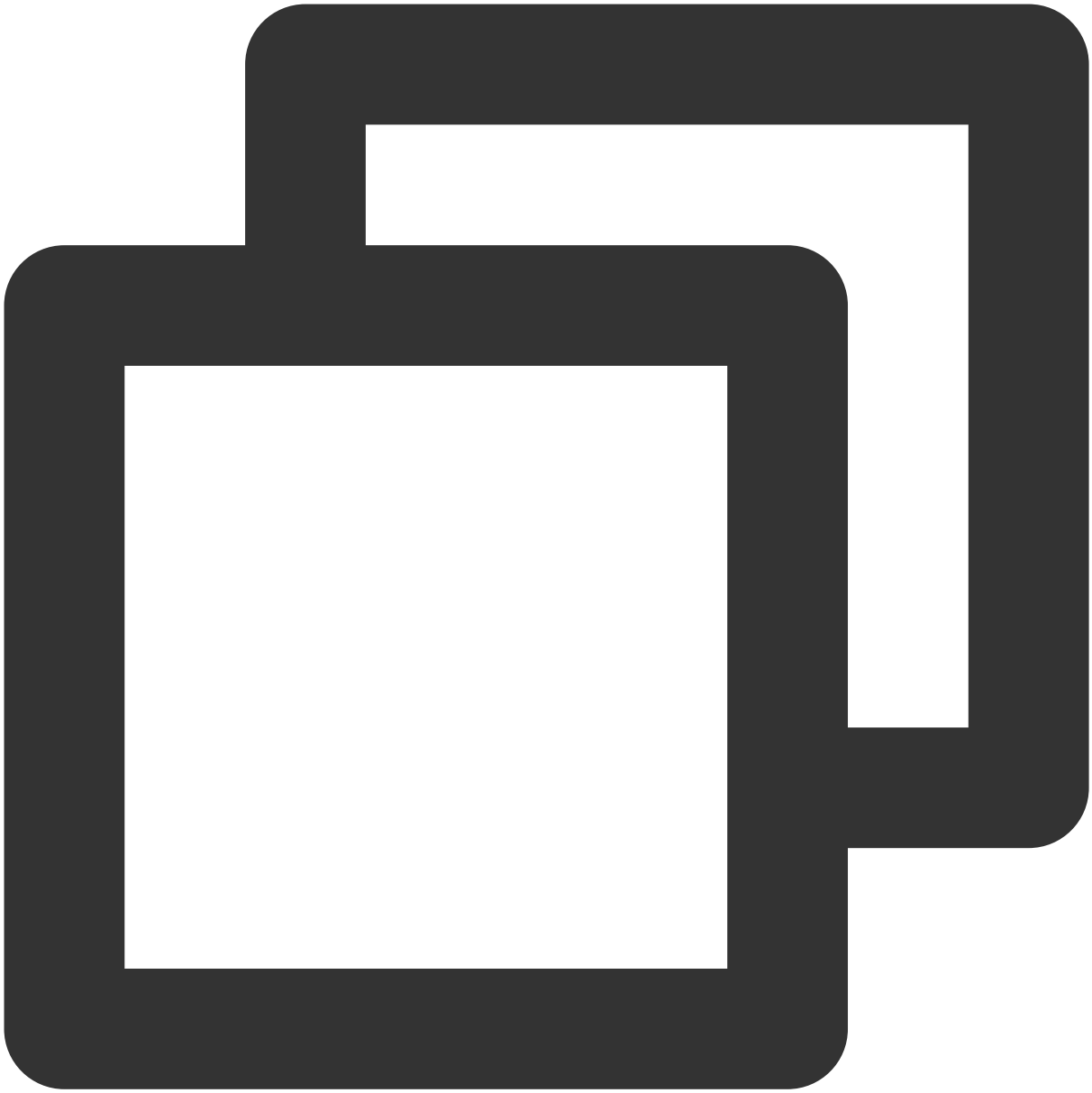
在“视频通话”（TRTC_APP_SCENE_VIDEOCALL ） 和 “语音通话”（TRTC_APP_SCENE_AUDIOCALL）这两个场景中，您无需要在进入房间时设置角色，因为在这两个模式下每个用户默认都是主播（Anchor）。

在“视频直播”（TRTC_APP_SCENE_LIVE） 和 “语音直播”（TRTC_APP_SCENE_VOICE_CHATROOM）这两个场景中，每个用户在进入房间时都需要指定自己的“角色”，要么是“主播（Anchor）”，要么是“观众（Audience）”。

角色的切换

在 TRTC 中，只有“主播（Anchor）”才有权限发布音视频流，“观众（Audience）”是不能发布音视频流的。

因此，如果您在进房时使用的角色是“观众（Audience）”，需要先调用 **switchRole** 接口将角色切换成“主播（Anchor）”，才能发布音视频流（也就是俗称的“上麦”）。



```
// 如果您当前的角色是观众（Audience），需要先调用 switchRole 切换到主播（Anchor）
// If your current role is 'audience', you need to call switchRole to switch to 'an
trtcCloud.switchRole(TRTCCloudDef.TRTCRoleAnchor);
trtcCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);
trtcCloud.startLocalPreview(true, cameraVideo);

// 如果切换角色失败，onSwitchRole 回调的错误码便不是 0
// If switching operation failed, the error code of the 'onSwitchRole' is not zero
onRtcListener(type, param) async {
  if (type == TRTCCloudListener.onSwitchRole) {
    if (param['errCode'] != 0) {
```

```
        // TO DO
      }
    }
  }
```

注意：

如果房间中已有的主播已经太多，会导致 switchRole 角色切换失败，并通过 TRTC 的 onSwitchRole 将错误码回调通知您。所以，当您不再需要发布音视频流（也就是俗称的“下麦”）时，就需要您再次调用 switchRole 并切换为“观众（Audience）”。

说明：

您可以能有一个疑问，既然只要主播才能发布音视频流，那我是不是可以让每一个用户都用主播的角色进入房间呢？答案肯定是不行的，原因需要看进阶指引 一个房间中同时最多可以有多少路音频和视频？

进阶指引

1. 一个房间中同时最多可以有多少路音频和视频？

一个 TRTC 房间中最多允许同时有 **50** 路音视频流，超出的音视频流会按照“先到先得”的原则被放弃掉。

在绝大多数场景下，小到两个人之间的视频通话，大到几万人同时观看的在线直播，50 路的音视频流都能满足应用场景的需求，但前提是您要做好**角色管理**。

所谓的“角色管理”就是您需要为进入房间的用户分配角色。

如果一个用户本身就是直播场景中的“主播”，或者是在线教育场景中的“老师”，抑或是在线会议场景中的“主持人”，都可以分配给“主播（Anchor）”这个角色。

如果一个用户本身就是直播场景中的“观众”，或者是在线教育场景中的“学生”，抑或是在线会议场景中的“旁听者”，应当分配为“观众（Audience）”这个角色，否则他们庞大的人数会瞬间将主播的人数限制“挤爆”。

只有当“观众”需要发布音视频流（“上麦”）时，才需要通过 switchRole 切换成“主播”角色，等不再需要发布音视频流（“下麦”）时，则需要立刻切回观众角色。

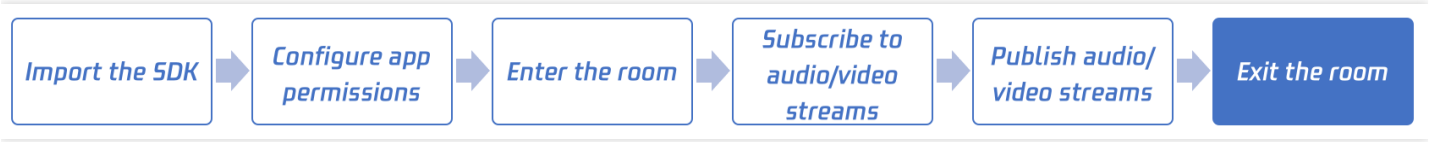
通过合理得角色管理，您会发现房间中同时需要发布音视频流的“主播”是通常不会超过 50 路的，否则整个房间中就会“乱糟糟”，毕竟同时说话的人声一旦超过 6 路，普通人就很难从声音中辨识出当前究竟是谁在说话了。

版权所有：腾讯云计算（北京）有限责任公司

05. 退出房间
Android&iOS&Windows&Mac

最近更新时间：2022-07-04 12:19:46

本文档主要介绍如何主动退出当前 TRTC 房间，同时还会介绍在什么情况下会被迫退出房间：



调用指引

步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。
请参考文档 进入房间 完成进房流程。

步骤2：主动退出当前房间

调用 exitRoom 接口即可退出当前的房间，SDK 会在退房结束后通过 onExitRoom(int reason) 回调事件通知您。

- Android java
- iOS&Mac swift
- Windows C++

```
// 退出当前房间
mCloud.exitRoom();
```

当您调用了 exitRoom 接口之后，SDK 会进入退房流程，其中有两项非常重要的任务：

- **关键任务一：通告自己的离开**
  通知房间中的其他用户自己将要退出当前的这个房间，房间中的其他用户会收到来自该用户的 **onRemoteUserLeaveRoom** 回调，否则其他用户可能误以为该用户已经“卡死了”。
- **关键任务二：释放设备权限**
  如果用户在退房之前正在发布音视频流，则退房流程中还需要关闭摄像头和麦克风并释放设备的使用权限。

因此，如果您希望释放 TRTCCloud 实例，建议等收到 onExitRoom 回调之后再释放。

步骤3：被迫退出当前房间

除了用户主动退出房间，还有两种情况下您也会收到 onExitRoom(int reason) 回调：

- **情况一：被踢出当前房间**
  您可以通过服务端的 RemoveUser | RemoveUserByStrRoomId 接口将某个用户踢出某个 TRTC 房间，将该用户踢出后，该用户会收到 onExitRoom(1) 的回调。

- **情况二：当前房间被解散**
  您可以通过服务端的 DismissRoom | DismissRoomByStrRoomId接口将某个 TRTC 房间解散，解散房间之后，该房间的所有用户都会收到 onExitRoom(2) 的回调。

- Android java
- iOS&Mac ObjC
- Windows C++

```
// 监听 onExitRoom 回调即可获知自己的退房原因
@Override
public void onExitRoom(int reason) {
if (reason == 0) {
Log.d(TAG, "Exit current room by calling the 'exitRoom' api of sdk ...");
} else if (reason == 1) {
Log.d(TAG, "Kicked out of the current room by server through the restful api..."
);
} else if (reason == 2) {
Log.d(TAG, "Current room is dissolved by server through the restful api...");
}
}
```

Web

最近更新时间：2023-08-22 17:01:55

本文档主要介绍如何主动退出当前 TRTC 房间，同时还会介绍在什么情况下会被迫退出房间：

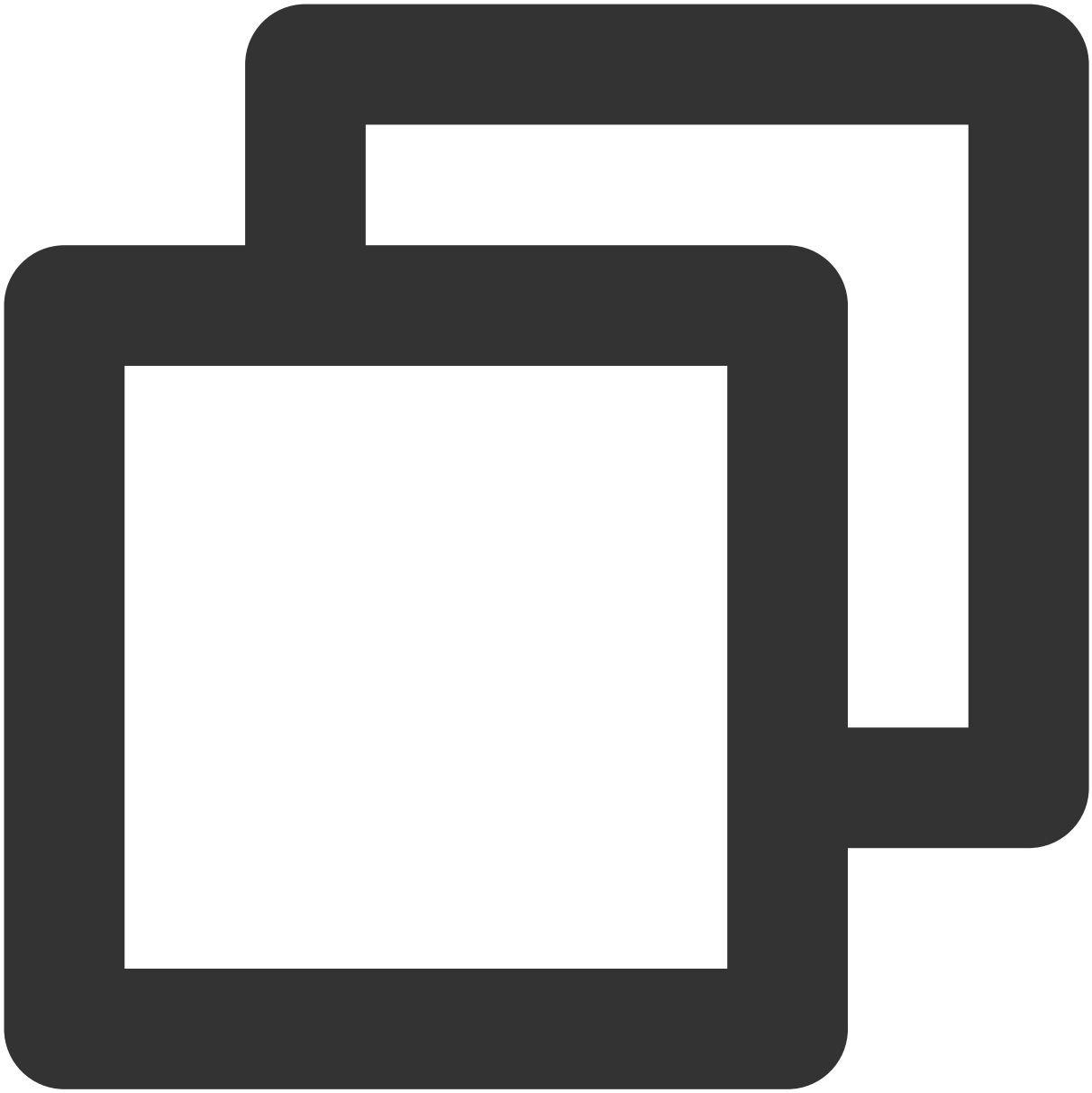


步骤1：完成前序步骤

可以参考文档 进入房间，创建 trtc 并进入房间。

步骤2：退出当前房间

调用 `trtc.exitRoom()` 方法退出房间，结束音视频通话。



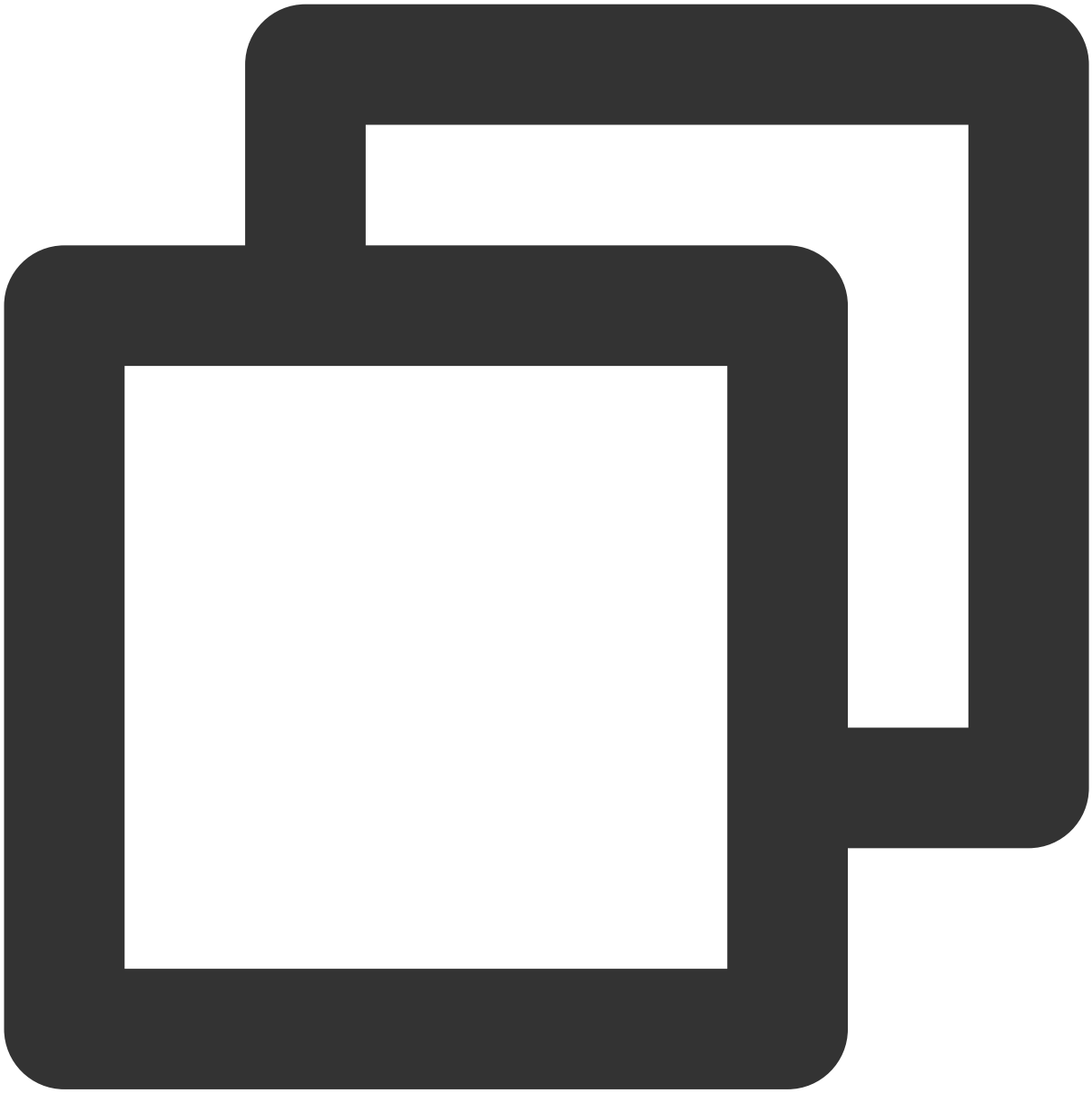
```
await trtc.exitRoom();
// 退房成功后，若后续无需使用 trtc 实例，则可以调用 trtc.destroy 方法销毁实例，及时释放相关资源
trtc.destroy();
```

处理被踢出

除了用户主动退出房间之外，用户也有可能因为如下原因被踢出房间，此时 SDK 会抛出KICKED_OUT 事件，这时不需要调用 `trtc.exitRoom()` 退房，SDK 自动进入退房状态。

1. `kick` ：两个相同 userId 的用户进入相同房间，前一个进房的用户会被踢出。同名用户同时进入同一房间是不允许的行为，可能会导致双方音视频通话异常，应避免出现这种情况。

2. `banned` ：通过服务端的RemoveUser | RemoveUserByStrRoomId接口将某个用户踢出某个 TRTC 房间，该用户会收到被踢事件，reason 为 `banned` 。
3. `room-disband` ：通过服务端的 DismissRoom | DismissRoomByStrRoomId 接口将某个 TRTC 房间解散，解散房间之后，该房间的所有用户都会收到被踢事件，reason 为 `room-disband` 。



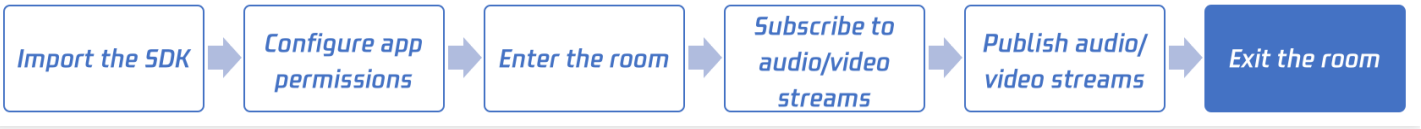
```
trtc.on(TRTC.EVENT.KICKED_OUT, error => {
  console.error(`kicked out, reason:${error.reason}, message:${error.message}`);
  // error.reason 有以下几种情况
  // 'kick' 由于相同 userId 进相同房间，导致先进入的用户被踢。
  // 'banned' 被管理员移出房间
```

```
  // 'room-disband' 管理员解散了房间
});
```

Electron

最近更新时间：2022-07-21 16:46:53

本文档主要介绍如何主动退出当前 TRTC 房间，同时还会介绍在什么情况下会被迫退出房间：



调用指引

步骤1：完成前序步骤

1. 请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。
2. 请参考文档 进入房间 完成进房流程。

步骤2：主动退出当前房间

调用 exitRoom 接口即可退出当前的房间，SDK 会在退房结束后通过 onExitRoom(int reason) 回调事件通知您。

```
import TRTCCloud from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
// 退出当前房间
trtcCloud.exitRoom();
```

当您调用了 exitRoom 接口之后，SDK 会进入退房流程，其中有两项非常重要的任务：

- **关键任务一：通告自己的离开**
  通知房间中的其他用户自己将要退出当前的这个房间，房间中的其他用户会收到来自该用户的 **onRemoteUserLeaveRoom** 回调，否则其他用户可能误以为该用户已经“卡死了”。
- **关键任务二：释放设备权限**
  如果用户在退房之前正在发布音视频流，则退房流程中还需要关闭摄像头和麦克风并释放设备的使用权限。

因此，如果您希望释放 TRTCCloud 实例，建议等收到 onExitRoom 回调之后再释放。

步骤3：被迫退出当前房间

除了用户主动退出房间，还有两种情况下您也会收到 onExitRoom 回调：

- **情况一：被踢出当前房间**
  您可以通过服务端的 RemoveUser | RemoveUserByStrRoomId 接口将某个用户踢出某个 TRTC 房间，将该用户踢出后，该用户会收到 onExitRoom(1) 的回调。

- **情况二：当前房间被解散**
  您可以通过服务端的 DismissRoom | DismissRoomByStrRoomId接口将某个 TRTC 房间解散，解散房间之后，该房间的所有用户都会收到 onExitRoom(2) 的回调。

```
// 监听 onExitRoom 回调即可获知自己的退房原因
function onExitRoom(reason) {
console.log(`onExitRoom reason: ${reason}`);
}
trtcCloud.on('onExitRoom', onExitRoom);
```

Flutter

最近更新时间：2024-02-02 18:46:38

本文档主要介绍如何主动退出当前 TRTC 房间，同时还会介绍在什么情况下会被迫退出房间：

调用指引

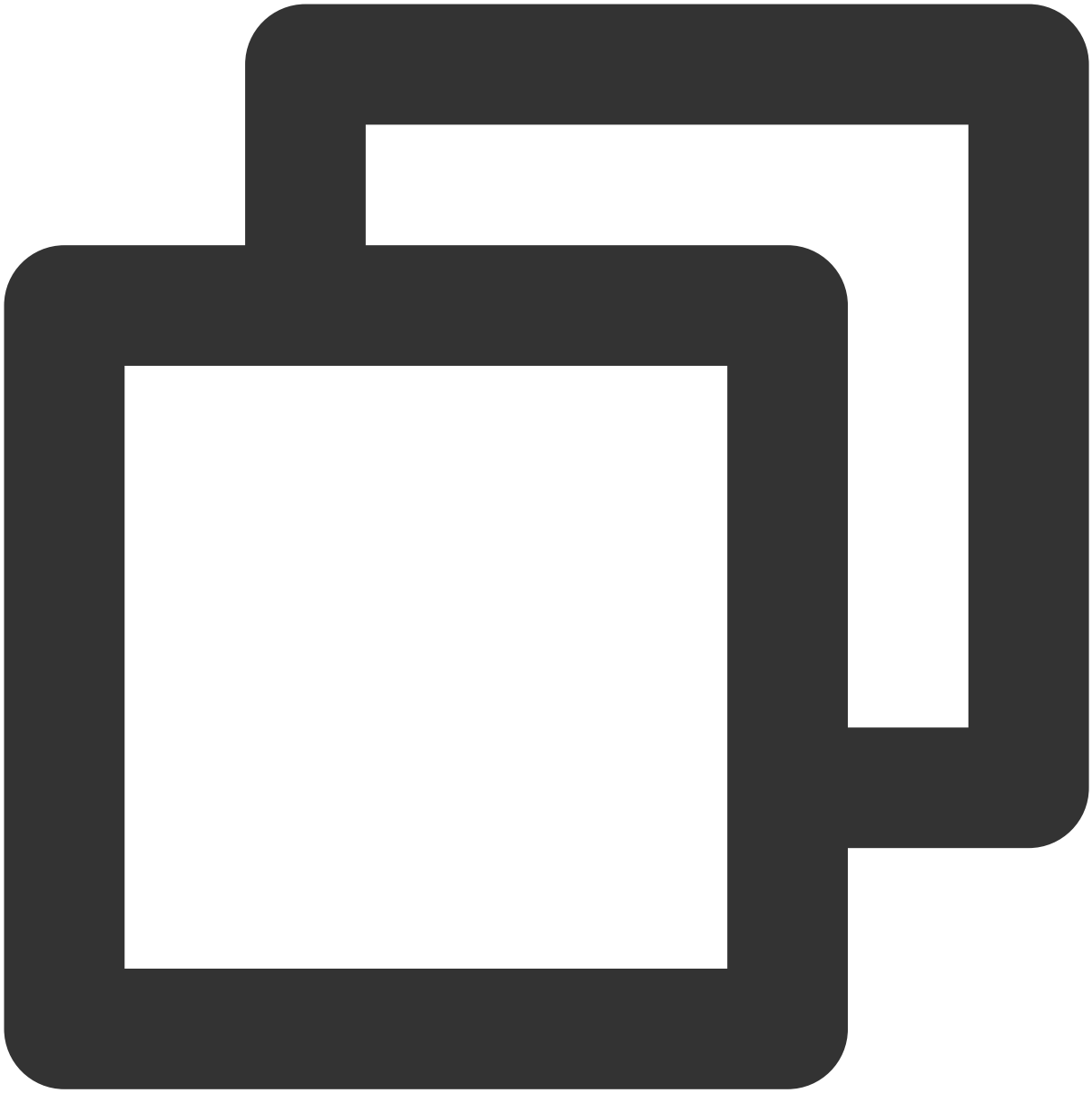
步骤1：完成前序步骤

请参考文档 导入 SDK 到项目中 完成 SDK 的导入和 App 权限的配置。
请参考文档 进入房间 完成进房流程。

步骤2：主动退出当前房间

调用 exitRoom 接口即可退出当前的房间，SDK 会在退房结束后通过 onExitRoom(int reason) 回调事件通知您。

版权所有：腾讯云计算（北京）有限责任公司



```
// 退出当前房间
await trtcCloud.exitRoom();
```

当您调用了 exitRoom 接口之后，SDK 会进入退房流程，其中有两项非常重要的任务：

关键任务一：通告自己的离开

通知房间中的其他用户自己将要退出当前的这个房间，房间中的其他用户会收到来自该用户的 **onRemoteUserLeaveRoom** 回调，否则其他用户可能误以为该用户已经“卡死了”。

关键任务二：释放设备权限

如果用户在退房之前正在发布音视频流，则退房流程中还需要关闭摄像头和麦克风并释放设备的使用权限。

因此，如果您希望释放 TRTCCloud 实例，建议等收到 onExitRoom 回调之后再释放。

步骤3：被迫退出当前房间

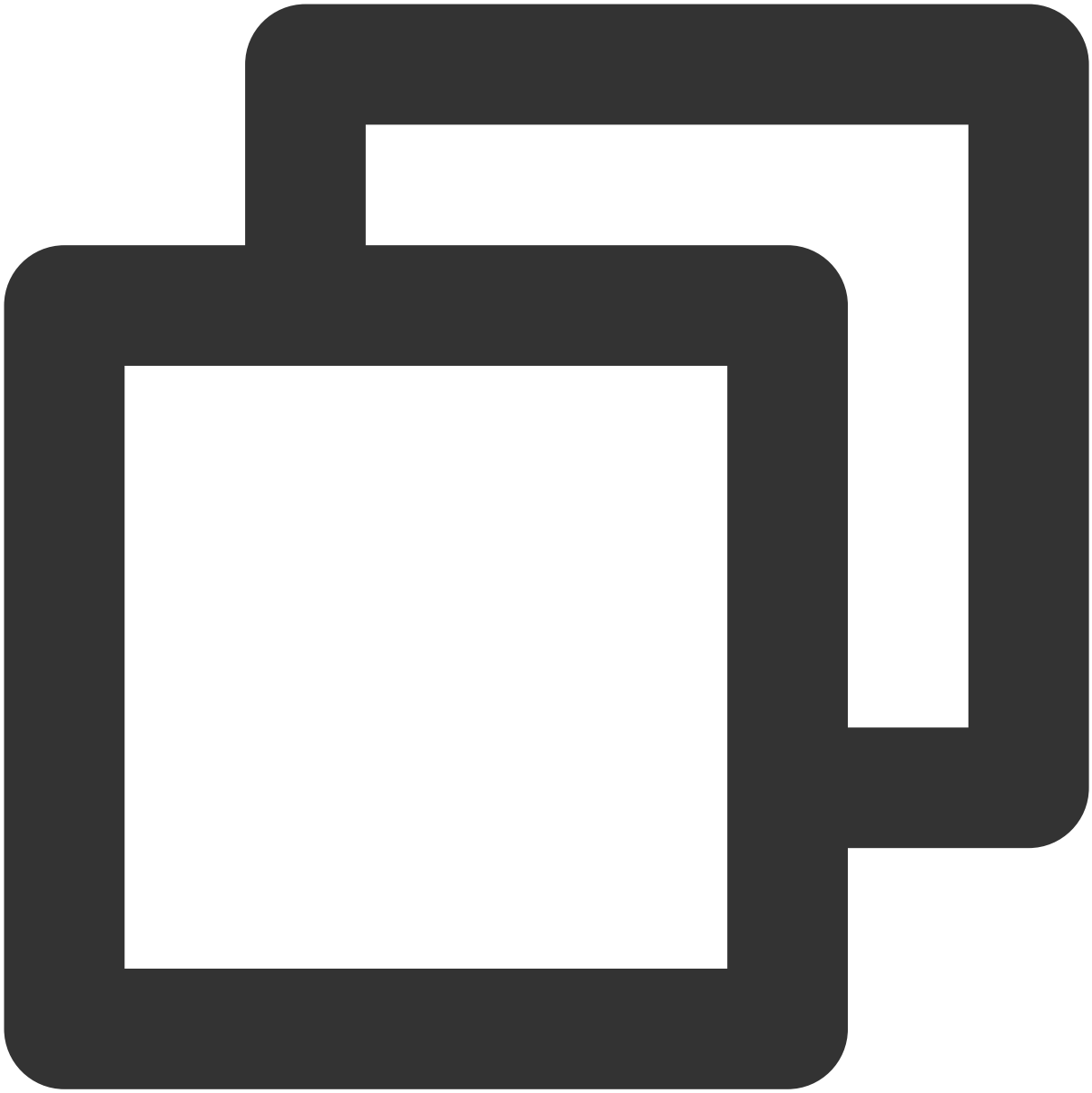
除了用户主动退出房间，还有两种情况下您也会收到 onExitRoom(int reason) 回调：

情况一：被踢出当前房间

您可以通过服务端的 RemoveUser | RemoveUserByStrRoomId 接口将某个用户踢出某个 TRTC 房间，将该用户踢出后，该用户会收到 onExitRoom(1) 的回调。

情况二：当前房间被解散

您可以通过服务端的 DismissRoom | DismissRoomByStrRoomId接口将某个 TRTC 房间解散，解散房间之后，该房间的所有用户都会收到 onExitRoom(2) 的回调。



```
onRtcListener(type, param) async {
    if (type == TRTCCloudListener.onExitRoom) {
      if (param == 0) {
          log('Exit current room by calling the 'exitRoom' api of sdk ...');
      } else if (param == 1) {
          log('Kicked out of the current room by server through the restful api...'
      } else if (param == 2) {
          log('Current room is dissolved by server through the restful api...');
      }
    }
}
```

版权所有：腾讯云计算（北京）有限责任公司

06. 进阶指引
感知网络质量
Android&iOS&Windows&Mac

最近更新时间：2022-07-04 12:19:46

本文档主要介绍如何感知当前网络的好与坏。

当我们在使用微信视频通话的时候，如果遇到网络环境较差的情况（比如在进入电梯以后），微信会在视频通话的界面上提示”您当前的网络质量较差“。本文档主要介绍如何通过 TRTC 完成同样的交互。



调用指引

TRTC 提供了一个叫做 **onNetworkQuality** 的回调事件，它会每隔两秒钟一次向您汇报当前的网络质量，其参数包括 localQuality 和 remoteQuality 两个部分：

- **localQuality** ：代表您当前的网络质量，分为 6 个等级，分别是 Excellent、Good、Poor、Bad、VeryBad 和 Down。

- **remoteQuality**：代表远端用户的网络质量，这是一个素组，素组中的每个元素代表一个远端用户的网络质量。

Quality	名称	说明
0	Unknown	未感知到
1	Excellent	当前网络非常好
2	Good	当前网络比较好
3	Poor	当前网络一般
4	Bad	当前网络较差，可能会出现明显的卡顿和通话延迟
5	VeryBad	当前网络很差，TRTC 只能勉强保持连接，但无法保证通讯质量
6	Down	当前网络不满足 TRTC 的最低要求，无法进行正常的音视频通话

您只需要监听 TRTC 的 onNetworkQuality 并在界面上做相应地提示即可：

- Android java
- iOS&Mac ObjC
- Windows C++

```
// 监听 onNetworkQuality 回调并感知当前网络状态的变化
@Override
public void onNetworkQuality(TRTCCloudDef.TRTCQuality localQuality,
ArrayList<trtcclouddef.trtcquality> remoteQuality)
{
// Get your local network quality
switch(localQuality) {
case TRTCQuality_Unknown:
Log.d(TAG, "SDK has not yet sensed the current network quality.");
break;
case TRTCQuality_Excellent:
Log.d(TAG, "The current network is very good.");
break;
case TRTCQuality_Good:
Log.d(TAG, "The current network is good.");
break;
case TRTCQuality_Poor:
Log.d(TAG, "The current network quality barely meets the demand.");
break;
case TRTCQuality_Bad:
Log.d(TAG, "The current network is poor, and there may be significant freezes an
```

```
d call delays.");
break;
case TRTCQuality_VeryBad:
Log.d(TAG, "The current network is very poor, the communication quality cannot b
e guaranteed");
break;
case TRTCQuality_Down:
Log.d(TAG, "The current network does not meet the minimum requirements.");
break;
default:
break;
}
// Get the network quality of remote users
for (TRTCCloudDef.TRTCQuality info : arrayList) {
Log.d(TAG, "remote user : = " + info.userId + ", quality = " + info.quality);
}
}
```

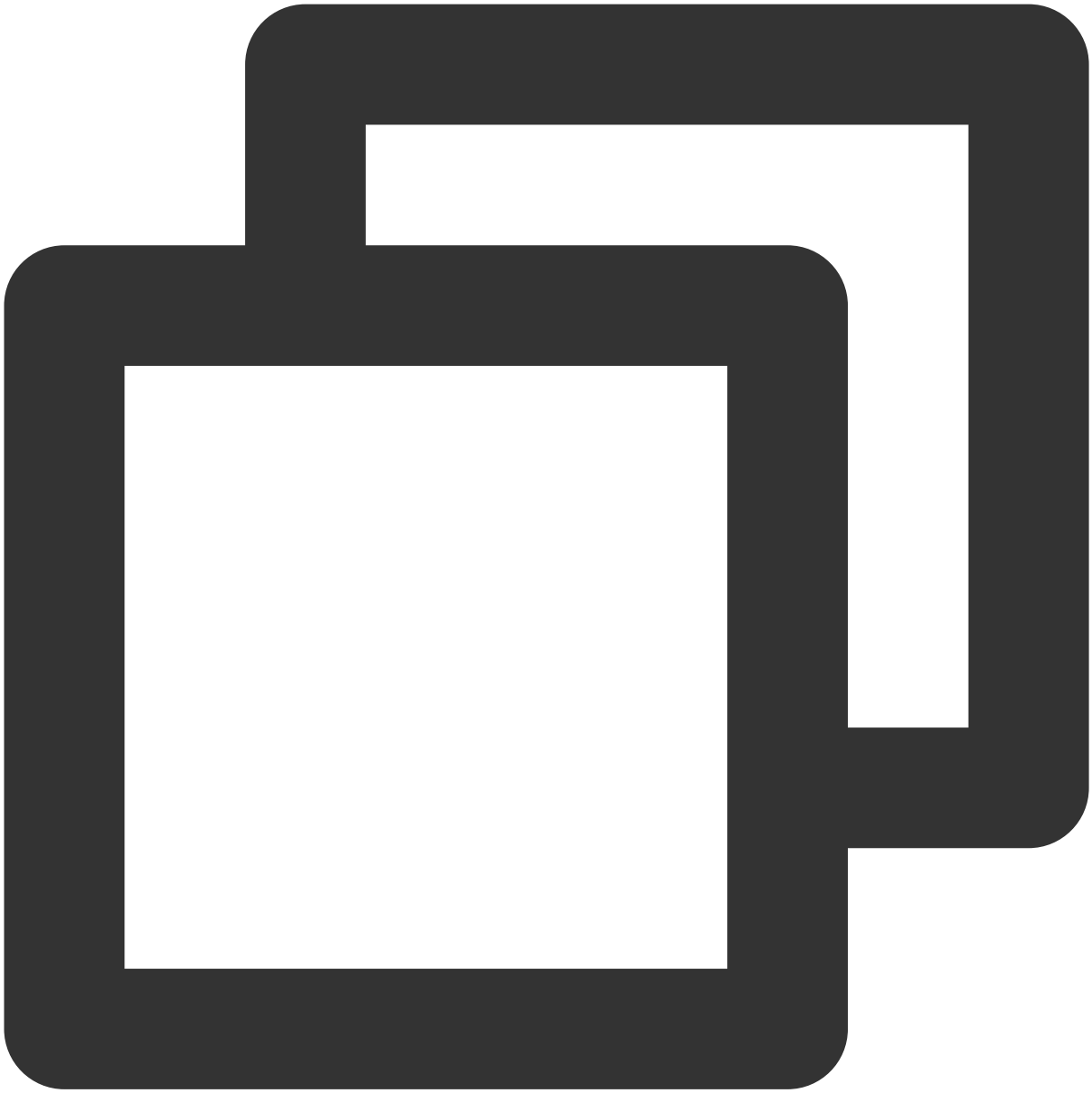
Web

最近更新时间：2023-07-21 14:18:45

在进房之前或在通话过程中，检测用户的网络质量，可以判断用户当下的网络质量情况。若用户网络质量太差，应建议用户检查网络或尝试更换网络，以保证正常通话质量。

本文主要介绍如何基于 NETWORK_QUALITY 事件实现通话前网络质量检测。通话过程中感知网络质量，只需监听 NETWORK_QUALITY 事件即可。

通话过程中的网络质量检测



```
const trtc = TRTC.create();
trtc.on(TRTC.EVENT.NETWORK_QUALITY, event => {
    console.log(`network-quality, uplinkNetworkQuality:${event.uplinkNetworkQuality}
    console.log(`uplink rtt:${event.uplinkRTT} loss:${event.uplinkLoss}`)
    console.log(`downlink rtt:${event.downlinkRTT} loss:${event.downlinkLoss}`)
})
```

通话前的网络质量检测

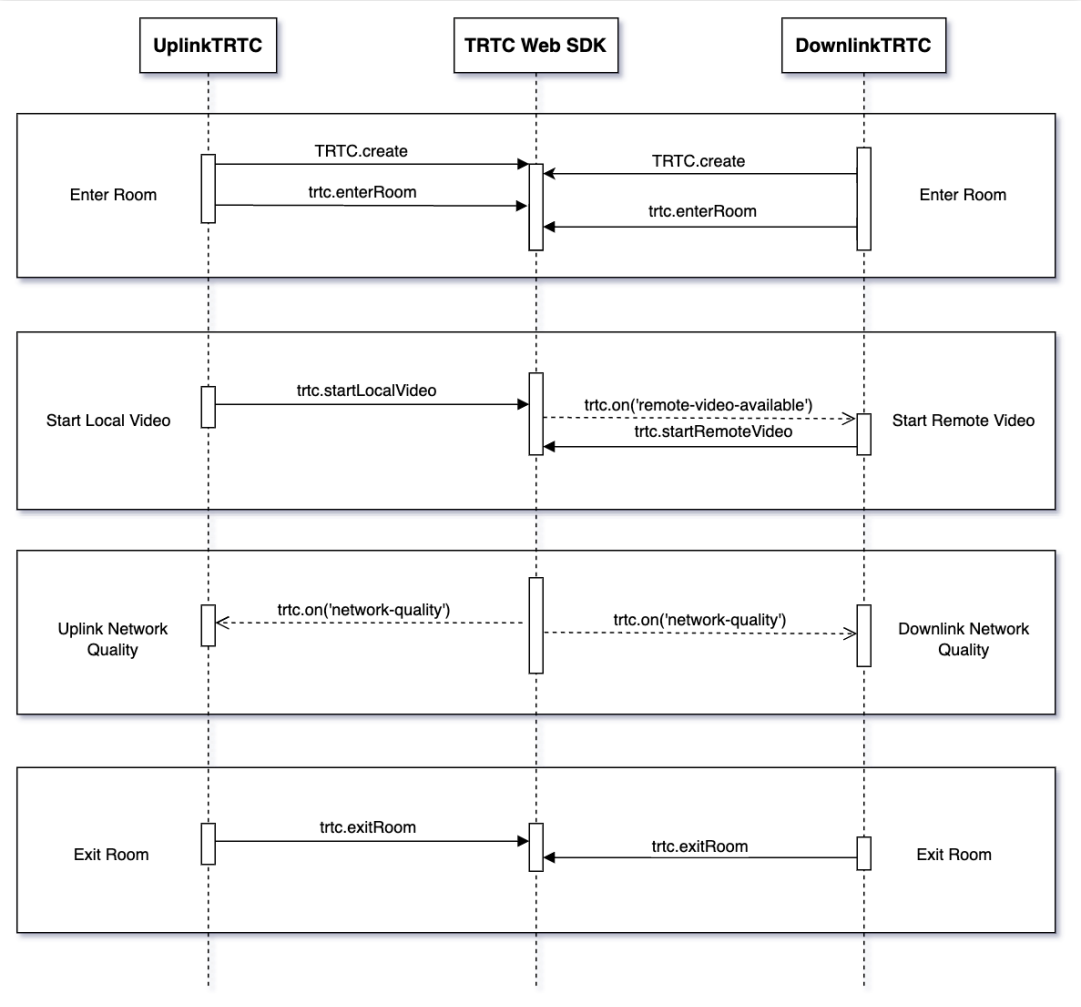
实现流程

1. 调用 `TRTC.create()` 方法创建两个 `TRTC`，分别称为 `uplinkTRTC` 和 `downlinkTRTC`。
2. 这两个 `TRTC` 都进入同一个房间。
3. 使用 `uplinkTRTC` 进行推流，监听 `TRTC.EVENT.NETWORK_QUALITY` 事件来检测上行网络质量。
4. 使用 `downlinkTRTC` 进行拉流，监听 `TRTC.EVENT.NETWORK_QUALITY` 事件来检测下行网络质量。
5. 整个过程可持续 15s 左右，最后取平均网络质量，从而大致判断出上下行网络情况。

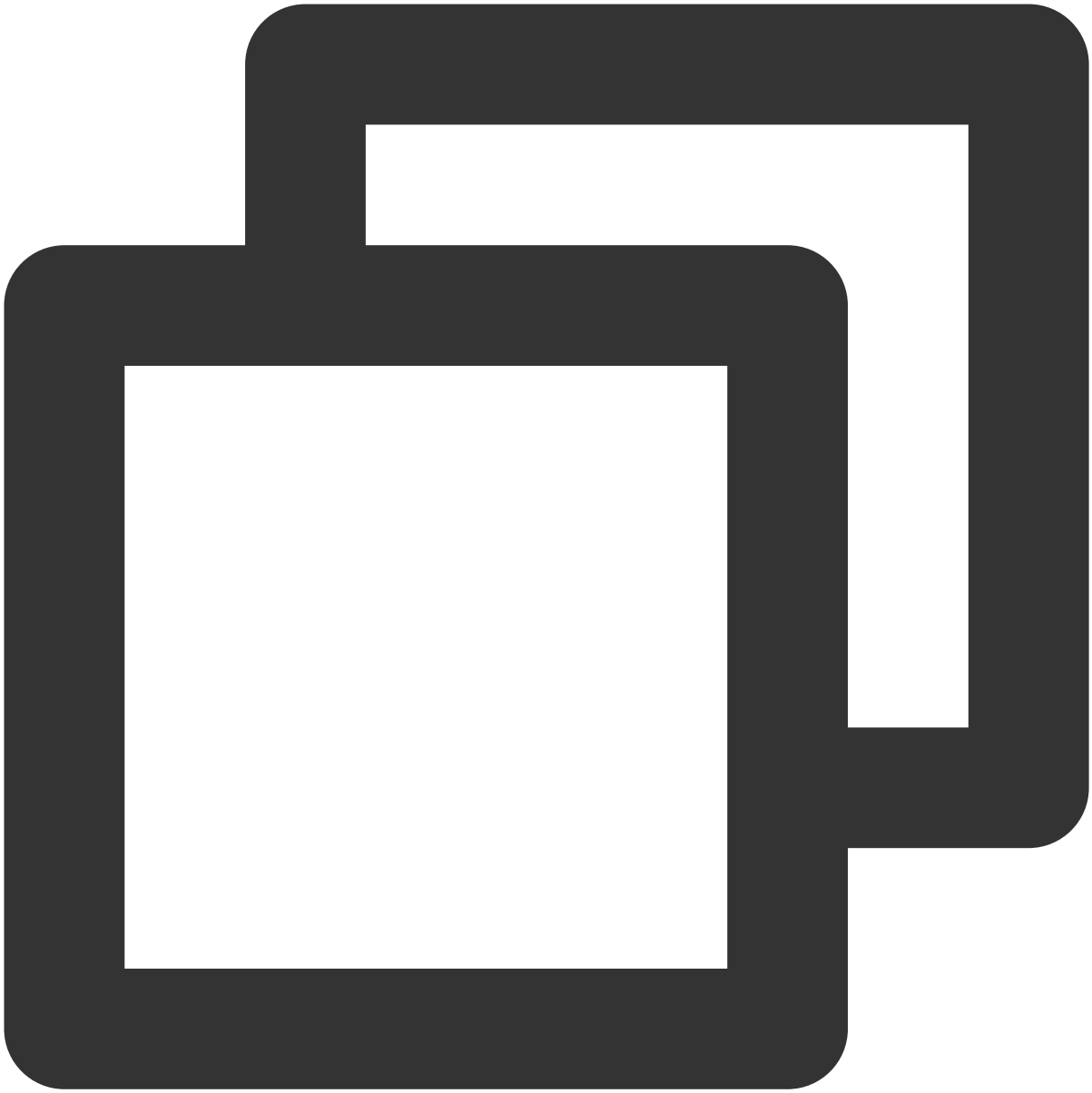
注意：

检测过程将产生少量的 基础服务费用。如果未指定推流分辨率，则默认以 640*480 的分辨率推流。

API 调用时序



代码示例



```
let uplinkTRTC = null; // 用于检测上行网络质量
let downlinkTRTC = null; // 用于检测下行网络质量
let localStream = null; // 用于测试的流
let testResult = {
  // 记录上行网络质量数据
  uplinkNetworkQualities: [],
  // 记录下行网络质量数据
  downlinkNetworkQualities: [],
  average: {
    uplinkNetworkQuality: 0,
    downlinkNetworkQuality: 0
```

```
  }
}

// 1. 检测上行网络质量
async function testUplinkNetworkQuality() {
  uplinkTRTC = TRTC.create();

  uplinkTRTC.enterRoom({
    roomId: 8080,
    sdkAppId: 0, // 填写 sdkAppId
    userId: 'user_uplink_test',
    userSig: '', // uplink_test 的 userSig
    scene: 'rtc'
  })

  uplinkTRTC.on(TRTC.EVENT.NETWORK_QUALITY, event => {
    const { uplinkNetworkQuality } = event;
    testResult.uplinkNetworkQualities.push(uplinkNetworkQuality);
  });

}

// 2. 检测下行网络质量
async function testDownlinkNetworkQuality() {
  downlinkTRTC = TRTC.create();
  downlinkTRTC.enterRoom({
    roomId: 8080,
    sdkAppId: 0, // 填写 sdkAppId
    userId: 'user_downlink_test',
    userSig: '', // userSig
    scene: 'rtc'
  });

  downlinkTRTC.on(TRTC.EVENT.NETWORK_QUALITY, event => {
      const { downlinkNetworkQuality } = event;
      testResult.downlinkNetworkQualities.push(downlinkNetworkQuality);
  })
}

// 3. 开始检测
testUplinkNetworkQuality();
testDownlinkNetworkQuality();

// 4. 15s 后停止检测，计算平均网络质量
setTimeout(() => {
  // 计算上行平均网络质量
  if (testResult.uplinkNetworkQualities.length > 0) {
```

```
    testResult.average.uplinkNetworkQuality = Math.ceil(
      testResult.uplinkNetworkQualities.reduce((value, current) => value + current,
    );
  }

  if (testResult.downlinkNetworkQualities.length > 0) {
    // 计算下行平均网络质量
    testResult.average.downlinkNetworkQuality = Math.ceil(
      testResult.downlinkNetworkQualities.reduce((value, current) => value + curren
    );
  }

  // 检测结束，清理相关状态。
  uplinkTRTC.exitRoom();
  downlinkTRTC.exitRoom();
}, 15 * 1000);
```

结果分析

经过上述步骤，可以拿到上行平均网络质量、下行平均网络质量。网络质量的枚举值如下所示：

数值	含义
0	网络状况未知，表示当前 TRTC 实例还没有建立上行/下行连接
1	网络状况极佳
2	网络状况较好
3	网络状况一般
4	网络状况差
5	网络状况极差
6	网络连接已断开 注意：若下行网络质量为此值，则表示所有下行连接都断开了

建议：当网络质量大于3时，应引导用户检查网络并尝试更换网络环境，否则难以保证正常的音视频通话。
也可通过下述策略来降低带宽消耗：
若上行网络质量大于3，则可通过 TRTC.updateLocalVideo() 接口降低码率 或 TRTC.stopLocalVideo() 方式关闭视频，以降低上行带宽消耗。
若下行网络质量大于3，则可通过订阅小流（参考：开启大小流传输）或者只订阅音频的方式，以降低下行带宽消耗。

Electron

最近更新时间：2022-07-21 16:50:46

本文档主要介绍如何感知当前网络的好与坏。

当我们在使用微信视频通话的时候，如果遇到网络环境较差的情况（比如在进入电梯以后），微信会在视频通话的界面上提示”您当前的网络质量较差“。本文档主要介绍如何通过 TRTC 完成同样的交互。



调用指引

TRTC 提供了一个叫做 **onNetworkQuality** 的回调事件，它会每隔两秒钟一次向您汇报当前的网络质量，其参数包括 localQuality 和 remoteQuality 两个部分：

- **localQuality** ：代表您当前的网络质量，分为 6 个等级，分别是 Excellent、Good、Poor、Bad、VeryBad 和 Down。
- **remoteQuality**：代表远端用户的网络质量，这是一个素组，素组中的每个元素代表一个远端用户的网络质量。

Quality	名称	说明

Quality	名称	说明
0	Unknown	未感知到
1	Excellent	当前网络非常好
2	Good	当前网络比较好
3	Poor	当前网络一般
4	Bad	当前网络较差，可能会出现明显的卡顿和通话延迟
5	VeryBad	当前网络很差，TRTC 只能勉强保持连接，但无法保证通讯质量
6	Down	当前网络不满足 TRTC 的最低要求，无法进行正常的音视频通话

您只需要监听 TRTC 的 onNetworkQuality 并在界面上做相应地提示即可：

```
import TRTCCloud, { TRTCQuality } from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
function onNetworkQuality(localQuality, remoteQuality) {
switch(localQuality.quality) {
case TRTCQuality.TRTCQuality_Unknown:
console.log('SDK has not yet sensed the current network quality.');
break;
case TRTCQuality.TRTCQuality_Excellent:
console.log('The current network is very good.');
break;
case TRTCQuality.TRTCQuality_Good:
console.log('The current network is good.');
break;
case TRTCQuality.TRTCQuality_Poor:
console.log('The current network quality barely meets the demand.');
break;
case TRTCQuality.TRTCQuality_Bad:
console.log('The current network is poor, and there may be significant freezes an
d call delays.');
break;
case TRTCQuality.TRTCQuality_Vbad:
console.log('The current network is very poor, the communication quality cannot b
e guaranteed.');
break;
case TRTCQuality.TRTCQuality_Down:
console.log('The current network does not meet the minimum requirements.');
break;
default:
```

```
break;
}
for (let i = 0; i < remoteQuality.length; i++) {
console.log(`remote user: ${remoteQuality[i].userId}, quality: ${remoteQuality
[i].quality}`);
}
}
rtcCloud.on('onNetworkQuality', onNetworkQuality);
```

Flutter

最近更新时间：2024-02-02 18:47:32

本文档主要介绍如何感知当前网络的好与坏。
当我们在使用微信视频通话的时候，如果遇到网络环境较差的情况（比如在进入电梯以后），微信会在视频通话的界面上提示“您当前的网络质量较差”。本文档主要介绍如何通过 TRTC 完成同样的交互。



调用指引

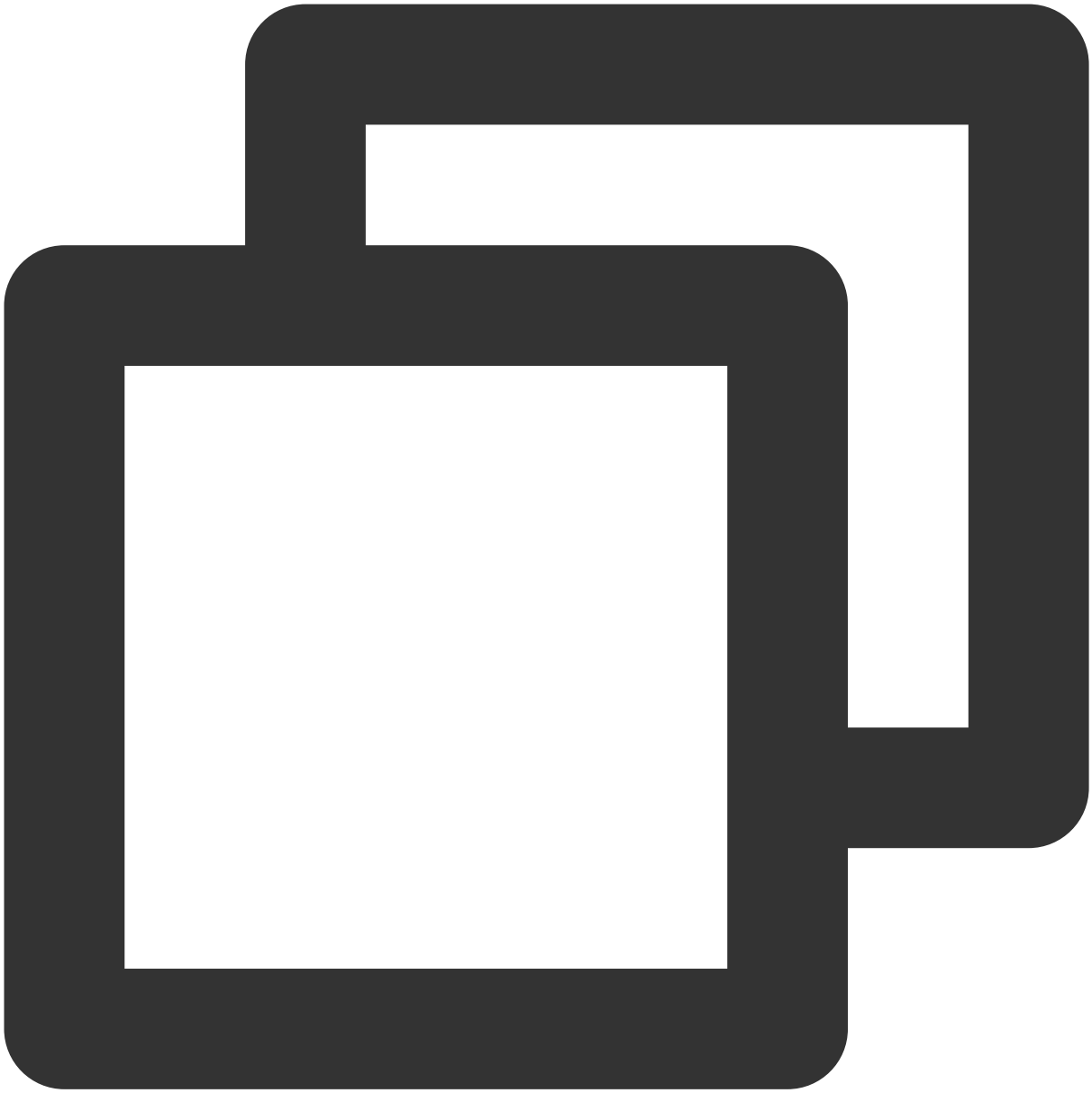
TRTC 提供了一个叫做 **onNetworkQuality** 的回调事件，它会每隔两秒钟一次向您汇报当前的网络质量，其参数包括 localQuality 和 remoteQuality 两个部分：

localQuality ：代表您当前的网络质量，分为 6 个等级，分别是 Excellent、Good、Poor、Bad、VeryBad 和 Down。

remoteQuality：代表远端用户的网络质量，这是一个素组，素组中的每个元素代表一个远端用户的网络质量。

Quality	名称	说明
0	Unknown	未感知到
1	Excellent	当前网络非常好
2	Good	当前网络比较好
3	Poor	当前网络一般
4	Bad	当前网络较差，可能会出现明显的卡顿和通话延迟
5	VeryBad	当前网络很差，TRTC 只能勉强保持连接，但无法保证通讯质量
6	Down	当前网络不满足 TRTC 的最低要求，无法进行正常的音视频通话

您只需要监听 TRTC 的 onNetworkQuality 并在界面上做相应地提示即可：



```
// 监听 onNetworkQuality 回调并感知当前网络状态的变化
if (type == TRTCCloudListener.onNetworkQuality) {
    if (type == TRTCCloudDef.TRTC_QUALITY_UNKNOWN) {
      // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Excellent) {
      // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Good) {
      // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Poor) {
      // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Bad) {
```

```
        // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Vbad) {
        // TODO
    } else if (type == TRTCCloudDef.TRTC_QUALITY_Down) {
        // TODO
    }
    // Get the network quality of remote users
    for (var info in param['remoteQuality']) {
        // TODO
    }
}
```

开启屏幕分享
iOS

最近更新时间：2024-06-07 11:20:49

腾讯云 TRTC 在 iOS 平台下支持两种不同的屏幕分享方案：

应用内分享

即只能分享当前 App 的画面，该特性需要 iOS 13 及以上版本的操作系统才能支持。由于无法分享当前 App 之外的屏幕内容，因此适用于对隐私保护要求高的场景。

跨应用分享

基于苹果的 Replaykit 方案，能够分享整个系统的屏幕内容，但需要当前 App 额外提供一个 Extension 扩展组件，因此对接步骤也相对应用内分享要多一点。

支持的平台

iOS	Android	Mac OS	Windows	Electron	Chrome 浏览器
✓	✓	✓	✓	✓	✓

应用内分享

应用内分享的方案非常简单，只需要调用 TRTC SDK 提供的接口 startScreenCaptureInApp 并传入编码参数 `TRTCVideoEncParam` 即可。该参数可以设置为 nil，此时 SDK 会沿用开始屏幕分享之前的编码参数。

我们推荐的用于 iOS 屏幕分享的编码参数是：

参数项	参数名称	常规推荐值	文字教学场景
分辨率	videoResolution	1280 × 720	1920 × 1080
帧率	videoFps	10 FPS	8 FPS
最高码率	videoBitrate	1600 kbps	2000 kbps
分辨率自适应	enableAdjustRes	NO	NO

由于屏幕分享的内容一般不会剧烈变动，所以设置较高的 FPS 并不经济，推荐10 FPS即可。

如果您要分享的屏幕内容包含大量文字，可以适当提高分辨率和码率设置。

版权所有：腾讯云计算（北京）有限责任公司

最高码率（videoBitrate）是指画面在剧烈变化时的最高输出码率，如果屏幕内容变化较少，实际编码码率会比较低。

跨应用分享

iOS 系统上的跨应用屏幕分享，需要增加 **Broadcast Upload Extension** 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 **App Group**，并在 XCode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以同主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 **Broadcast Upload Extension** 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 `TXLiteAVSDK_ReplayKitExt.framework` 。
3. 对接主 App 端的接收逻辑，让主 App 等待来自 **Broadcast Upload Extension** 的录屏数据。

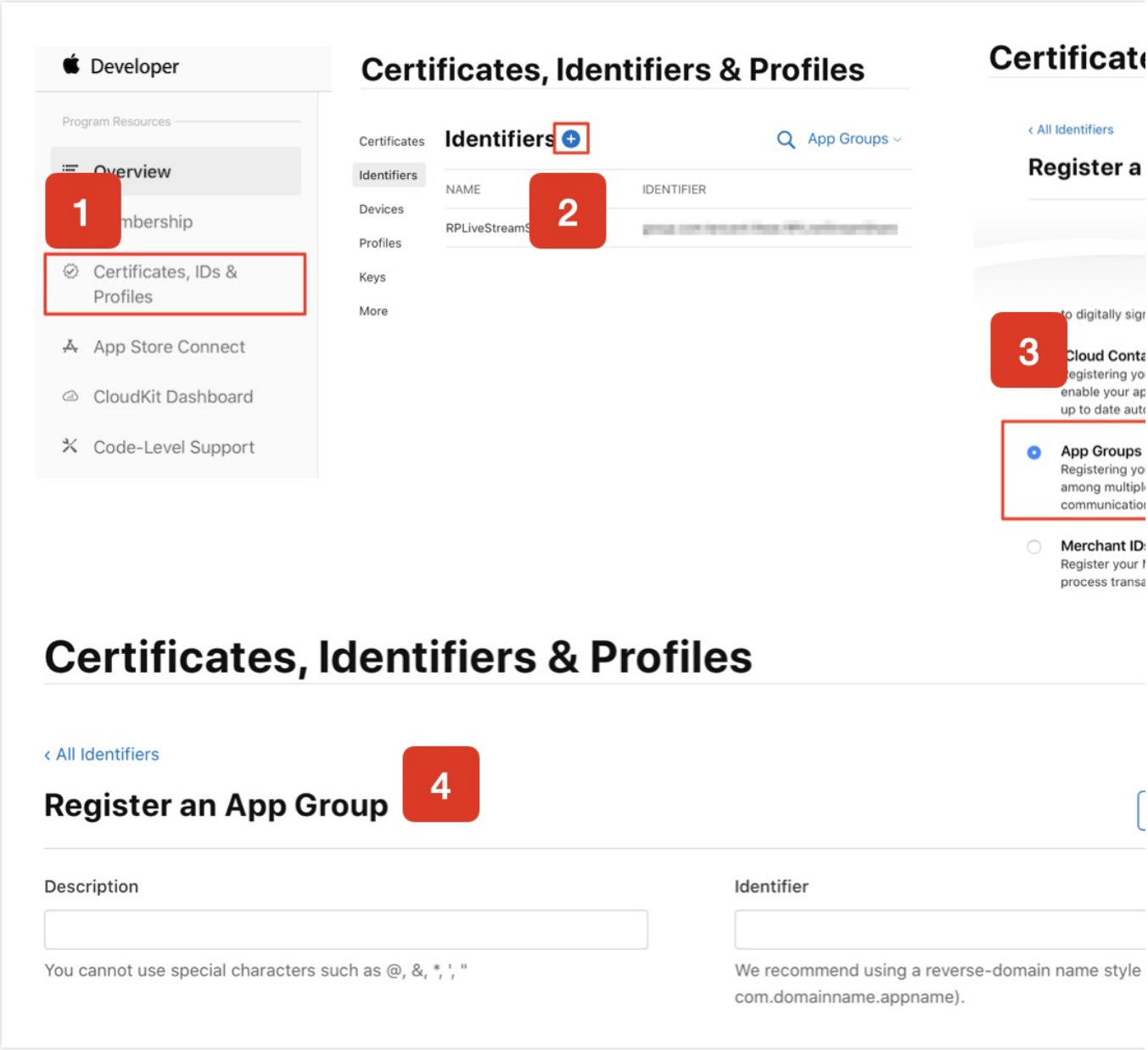
注意：

如果跳过步骤1，也就是不配置 App Group（接口传 nil），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

步骤1：创建 App Group

使用您的帐号登录 **https://developer.apple.com/** ，进行以下操作，**注意完成后需要重新下载对应的 Provisioning Profile**。

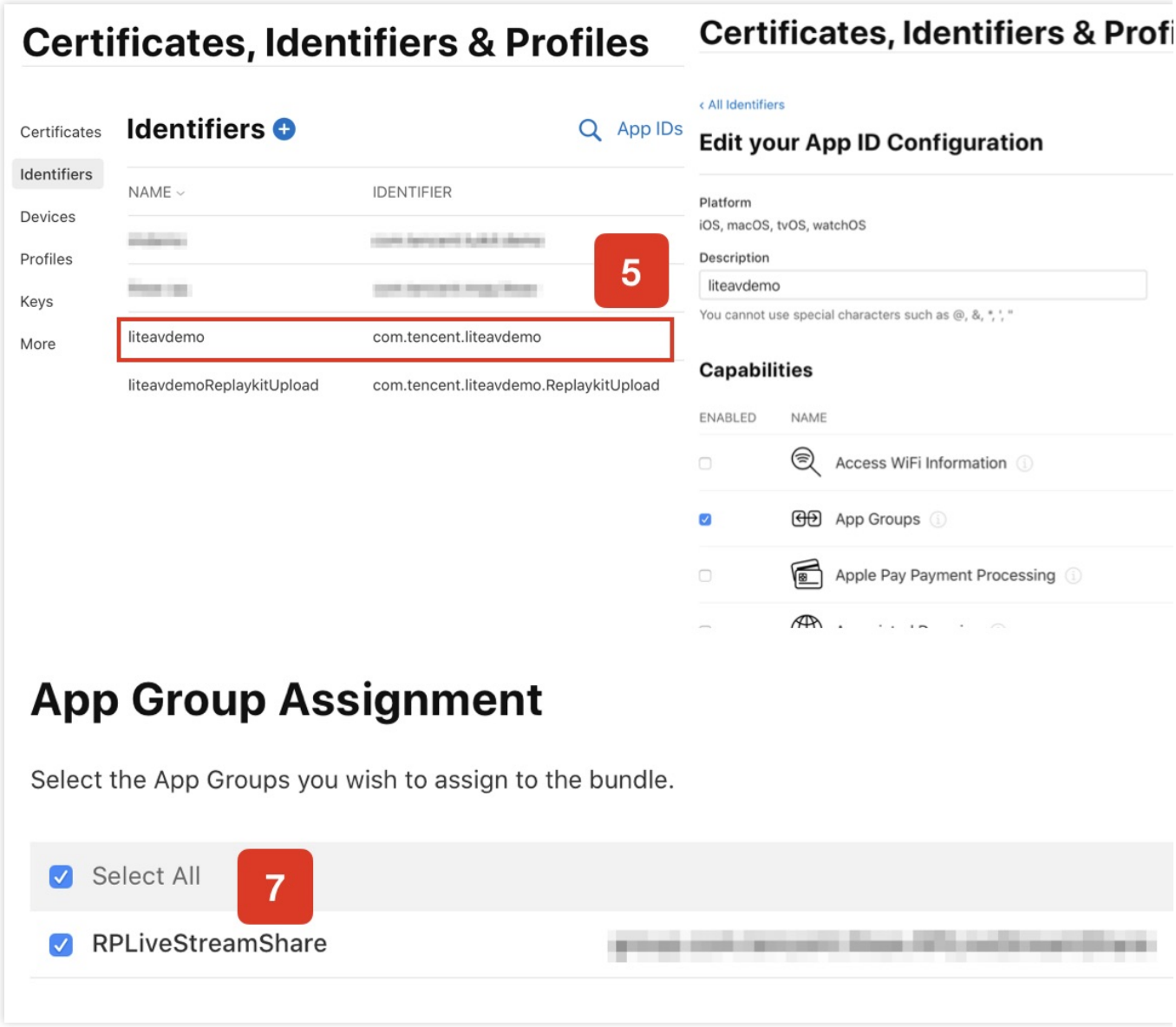
1. 单击 **Certificates, IDs & Profiles**。
2. 在右侧的界面中单击加号。
3. 选择 **App Groups**，单击 **Continue**。
4. 在弹出的表单中填写 Description 和 Identifier，其中 Identifier 需要传入接口中的对应的 AppGroup 参数。完成后单击 **Continue**。



5. 回到 Identifier 页面，左上边的菜单中选择 **App IDs**，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。

6. 选中 **App Groups** 并单击 **Edit**。

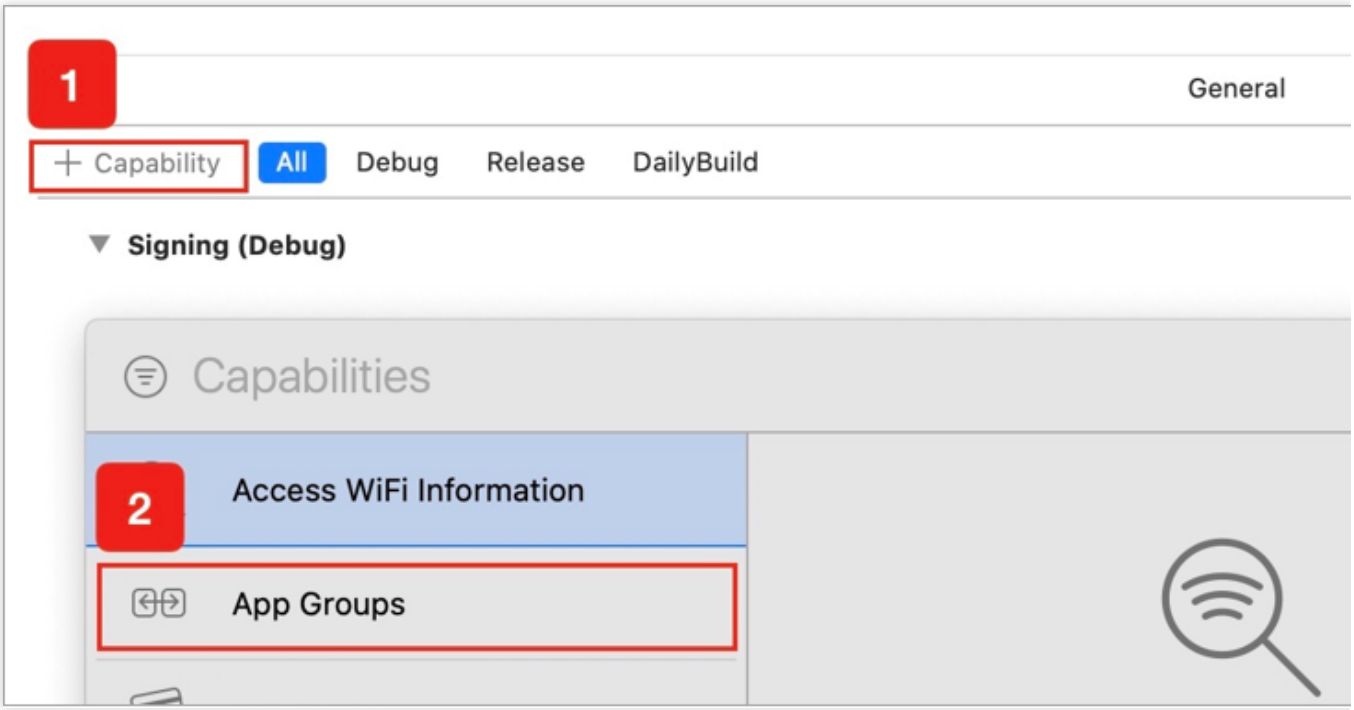
7. 在弹出的表单中选择您之前创建的 App Group，单击 **Continue** 返回编辑页，单击 **Save** 保存。



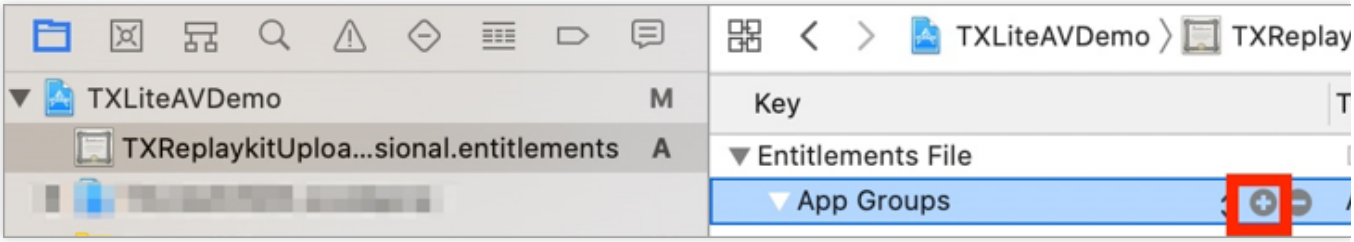
8. 重新下载 Provisioning Profile 并配置到 XCode 中。

步骤2：创建 Broadcast Upload Extension

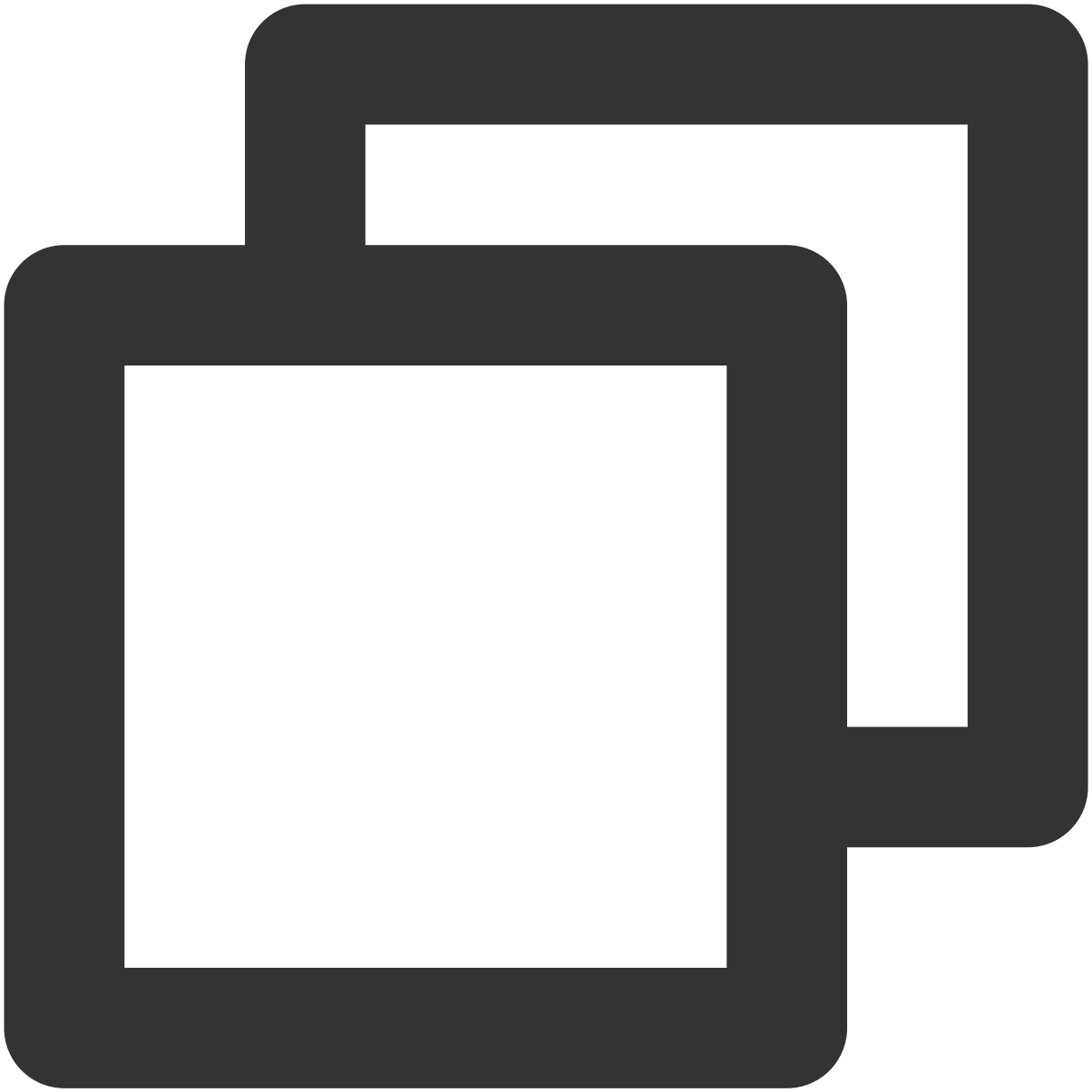
1. 在 Xcode 菜单依次单击 **File**、**New** 、**Target...**，选择 **Broadcast Upload Extension**。
2. 在弹出的对话框中填写相关信息，不用勾选 **Include UI Extension**，单击 **Finish** 完成创建。
3. 将下载到的 SDK 压缩包中的 TXLiteAVSDK_ReplayKitExt.framework 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击 **+ Capability**，双击 **App Groups**，如下图：



操作完成后，会在文件列表中生成一个名为 `Target名.entitlements` 的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



5. 选中主 App 的 Target，**并按照上述步骤对主 App 的 Target 做同样的处理。**
6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.h" 的文件，用如下代码进行替换。**需将代码中的 APPGROUP 改为上文中的创建的 App Group Identifier**。



```
#import "SampleHandler.h"
@import TXLiteAVSDK_ReplayKitExt;

#define APPGROUP @"group.com.tencent.liteav.RPLiveStreamShare"

@interface SampleHandler() <TXReplayKitExtDelegate>

@end

@implementation SampleHandler
// 注意：此处的 APPGROUP 需要改成上文中的创建的 App Group Identifier。
```

```
- (void)broadcastStartedWithSetupInfo:(NSDictionary<NSString *,NSObject *> *)setupI
    [[TXReplayKitExt sharedInstance] setupWithAppGroup:APPGROUP delegate:self];
}

- (void)broadcastPaused {
    // User has requested to pause the broadcast. Samples will stop being delivered
}

- (void)broadcastResumed {
    // User has requested to resume the broadcast. Samples delivery will resume.
}

- (void)broadcastFinished {
    [[TXReplayKitExt sharedInstance] finishBroadcast];
    // User has requested to finish the broadcast.
}

#pragma mark - TXReplayKitExtDelegate
- (void)broadcastFinished:(TXReplayKitExt *)broadcast reason:(TXReplayKitExtReason)
{
    NSString *tip = @"";
    switch (reason) {
        case TXReplayKitExtReasonRequestedByMain:
            tip = @"屏幕共享已结束";
            break;
        case TXReplayKitExtReasonDisconnected:
            tip = @"应用断开";
            break;
        case TXReplayKitExtReasonVersionMismatch:
            tip = @"集成错误（SDK 版本号不相符合）";
            break;
    }

    NSError *error = [NSError errorWithDomain:NSStringFromClass(self.class)
                                         code:0
                                     userInfo:@{
                                         NSLocalizedFailureReasonErrorKey:tip
                                     }];
    [self finishBroadcastWithError:error];
}

- (void)processSampleBuffer:(CMSampleBufferRef)sampleBuffer withType:(RPSampleBuffe
    switch (sampleBufferType) {
        case RPSampleBufferTypeVideo:
            [[TXReplayKitExt sharedInstance] sendVideoSampleBuffer:sampleBuffer];
            break;
        case RPSampleBufferTypeAudioApp:
```

```
            // Handle audio sample buffer for app audio
            break;
        case RPSampleBufferTypeAudioMic:
            // Handle audio sample buffer for mic audio
            break;

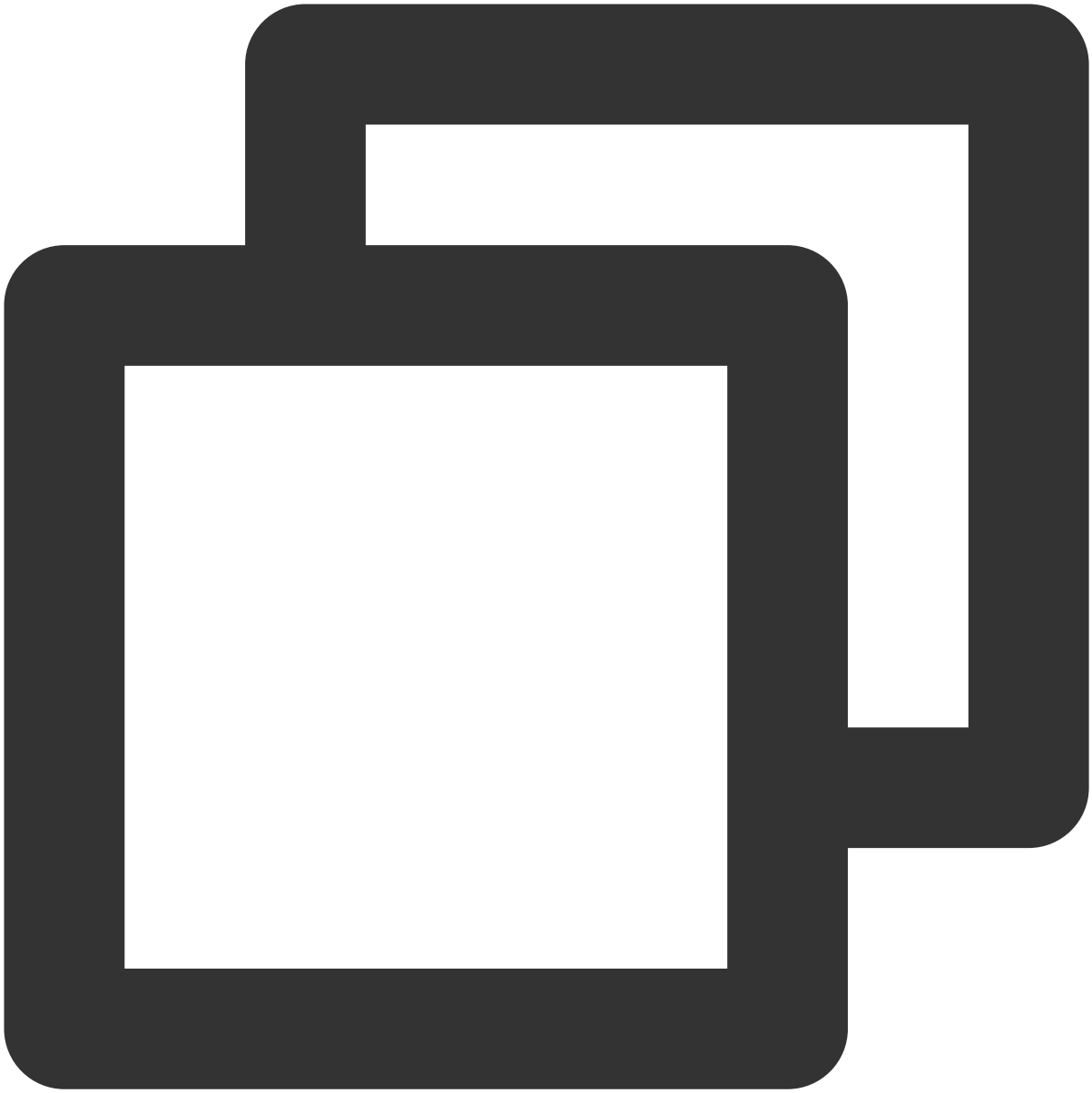
        default:
            break;
    }
}
@end
```

步骤3：对接主 App 端的接收逻辑

按照如下步骤，对接主 App 端的接收逻辑。也就是在用户触发屏幕分享之前，要让主 App 处于“等待”状态，以便随时接收来自 Broadcast Upload Extension 进程的录屏数据。

1. 确保 TRTCCloud 已经关闭了摄像头采集，如果尚未关闭，请调用 stopLocalPreview 关闭摄像头采集。
2. 调用 startScreenCaptureByReplaykit:appGroup: 方法，并传入 步骤1 中设置的 AppGroup，让 SDK 进入“等待”状态。
3. 等待用户触发屏幕分享。如果不实现 步骤4 中的“触发按钮”，屏幕分享就需要用户在 iOS 系统的控制中心，通过长按录屏按钮来触发，这一操作步骤如下图所示：
4. 通过调用 stopScreenCapture 接口可以随时中止屏幕分享。

版权所有：腾讯云计算（北京）有限责任公司



```
// 开始屏幕分享，需要将 APPGROUP 替换为上述步骤中创建的 App Group Identifier。
- (void)startScreenCapture {
    TRTCVideoEncParam *videoEncConfig = [[TRTCVideoEncParam alloc] init];
    videoEncConfig.videoResolution = TRTCVideoResolution_1280_720;
    videoEncConfig.videoFps = 10;
    videoEncConfig.videoBitrate = 2000;
    //需要将 APPGROUP 替换为上述步骤中创建的 App Group Identifier:
    [[TRTCCloud sharedInstance] startScreenCaptureByReplaykit:videoEncConfig
                                                     appGroup:APPGROUP];
}
```

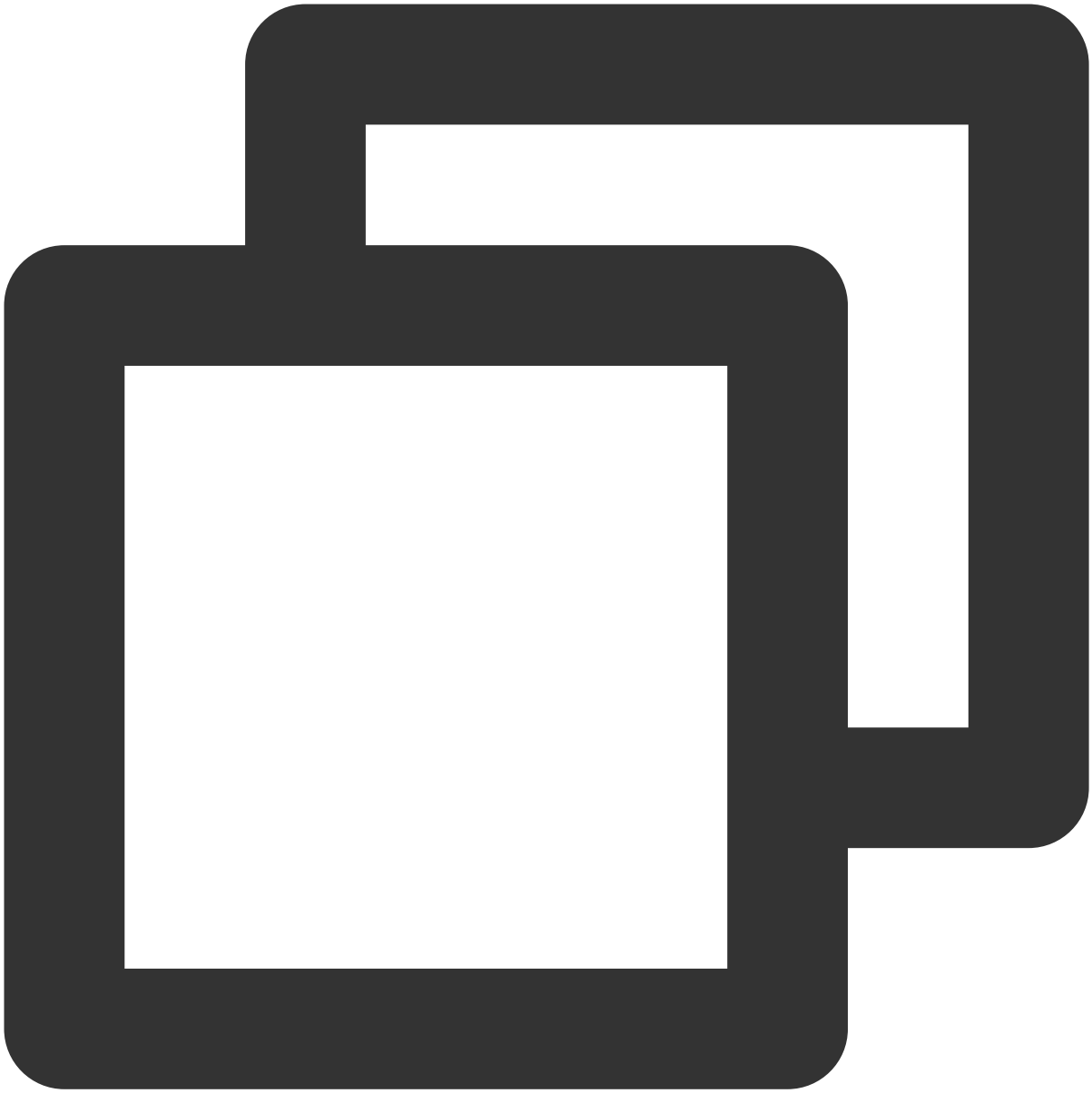
```
// 停止屏幕分享
- (void)stopScreenCapture {
    [[TRTCCloud sharedInstance] stopScreenCapture];
}

// 屏幕分享的启动事件通知，可以通过 TRTCCloudDelegate 进行接收
- (void)onScreenCaptureStarted {
    [self showTip:@"屏幕分享开始"];
}
```

步骤4：增加屏幕分享的触发按钮（可选）

截止到 步骤3 ，我们的屏幕分享还必须要用户从控制中心中长按录屏按钮来手动启动。您可通过下述方法实现类似腾讯会议的单击按钮即可触发的效果：

1. 在 Demo 中的 TRTCBroadcastExtensionLauncher 文件实现了唤起屏幕分享，找到并将其加入到您的工程中。
2. 在您的界面上放置一个按钮，并在按钮的响应函数中调用 `TRTCBroadcastExtensionLauncher` 中的 `launch` 函数，就可以唤起屏幕分享功能了。



```
// 自定义按钮响应方法
- (IBAction)onScreenButtonTapped:(id)sender {
    [TRTCBroadcastExtensionLauncher launch];
}
```

注意：

苹果在 iOS 12.0 中增加了 `RPSystemBroadcastPickerView` 可以从应用中弹出启动器供用户确认启动屏幕分享，到目前为止，`RPSystemBroadcastPickerView` 尚不支持自定义界面，也没有官方的唤起方法。

TRTCBroadcastExtensionLauncher 的原理就是遍历 `RPSystemBroadcastPickerView` 的子 View 寻找 UIButton 并触发了其点击事件。

但该方案不被苹果官方推荐，并可能在新一轮的系统更新中失效，因此 步骤4 只是一个可选方案，您需要自行承担风险来选用此方案。

观看屏幕分享

观看 Mac / Windows 屏幕分享

当房间里有一个 Mac / Windows 用户启动了屏幕分享，会通过辅流进行分享。房间里的其他用户会通过 TRTCCloudDelegate 中的 onUserSubStreamAvailable 事件获得这个通知。

希望观看屏幕分享的用户可以通过 startRemoteSubStreamView 接口来启动渲染远端用户辅流画面。

观看 Android / iOS 屏幕分享

若用户通过 Android / iOS 进行屏幕分享，会通过主流进行分享。房间里的其他用户会通过 TRTCCloudDelegate 中的 onUserVideoAvailable 事件获得这个通知。

希望观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户主流画面。

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2022-07-04 12:15:12

本文档主要介绍如何使用屏幕分享，目前一个 TRTC 音视频房间只能有一路屏幕分享。

调用指引

开启屏幕分享

步骤1：添加 Activity

在 manifest 文件中粘贴如下 activity（若项目代码中存在则不需要添加）。

```
<activity
android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantActi
vity"
android:theme="@android:style/Theme.Translucent"/>
```

步骤2： 启动屏幕分享

要开启 Android 端的屏幕分享，只需调用 `TRTCCloud` 中的 startScreenCapture() 接口即可。

通过设置 startScreenCapture() 中的首个参数 `encParams` ，您可以指定屏幕分享的编码质量。如果您指定 `encParams` 为 null，SDK 会自动使用之前设定的编码参数，我们推荐的参数设定如下：

参数项	参数名称	常规推荐值	文字教学场景
分辨率	videoResolution	1280 × 720	1920 × 1080
帧率	videoFps	10 FPS	8 FPS
最高码率	videoBitrate	1600 kbps	2000 kbps
分辨率自适应	enableAdjustRes	NO	NO

- 由于屏幕分享的内容一般不会剧烈变动，所以设置较高的 FPS 并不经济，推荐10 FPS即可。
- 如果您要分享的屏幕内容包含大量文字，可以适当提高分辨率和码率设置。
- 最高码率（videoBitrate）是指画面在剧烈变化时的最高输出码率，如果屏幕内容变化较少，实际编码码率会比较低。

步骤3：弹出悬浮窗防止应用被强杀（可选）

版权所有：腾讯云计算（北京）有限责任公司

从 Android 7.0 系统开始，切入到后台运行的普通 App 进程，但凡有 CPU 活动，都很容易会被系统强杀掉。 所以当 App 在切入到后台默默进行屏幕分享时，通过弹出悬浮窗的方案，可以避免被系统强杀掉。 同时，在手机屏幕上显示悬浮窗也有利于告知用户当前正在做屏幕分享，避免用户泄漏个人隐私。

要弹出悬浮窗，您只需要参考示例代码 FloatingView.java 中的实现即可：

```
public void showView(View view, int width, int height) {
mWindowManager = (WindowManager) mContext.getSystemService(Context.WINDOW_SERVIC
E);
int type = WindowManager.LayoutParams.TYPE_TOAST;
//TYPE_TOAST仅适用于4.4+系统，假如要支持更低版本使用TYPE_SYSTEM_ALERT(需要在manifest中声
明权限)
//7.1（包含）及以上系统对TYPE_TOAST做了限制
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
type = WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY;
} else if (Build.VERSION.SDK_INT > Build.VERSION_CODES.N) {
type = WindowManager.LayoutParams.TYPE_PHONE;
}
mLayoutParams = new WindowManager.LayoutParams(type);
mLayoutParams.flags = WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE;
mLayoutParams.flags |= WindowManager.LayoutParams.FLAG_WATCH_OUTSIDE_TOUCH;
mLayoutParams.width = width;
mLayoutParams.height = height;
mLayoutParams.format = PixelFormat.TRANSLUCENT;
mWindowManager.addView(view, mLayoutParams);
}
```

观看屏幕分享

- **观看 Mac / Windows 屏幕分享**
  当房间里有一个 Mac / Windows 用户启动了屏幕分享，会通过辅流进行分享。房间里的其他用户会通过 TRTCCloudListener 中的 onUserSubStreamAvailable 事件获得这个通知。

- **观看 Android / iOS 屏幕分享**
  若用户通过 Android / iOS 进行屏幕分享，会通过主流进行分享。房间里的其他用户会通过 TRTCCloudListener 中的 onUserVideoAvailable 事件获得这个通知。

观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户主流画面。

Mac

最近更新时间：2022-06-22 15:33:33

腾讯云 TRTC 支持屏幕分享功能，Mac 平台下的屏幕分享支持主路分享和辅路分享两种方案：

- **辅路分享**
  在 TRTC 中，我们可以单独为屏幕分享开启一路上行的视频流，并称之为“辅路（**substream**）”。辅路分享即主播同时上行摄像头画面和屏幕画面两路画面。这是腾讯会议的使用方案，您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeSub` 来启用该模式。观看该路画面需要使用专门的 `startRemoteSubStreamView` 接口。

- **主路分享**
  在 TRTC 中，我们一般把摄像头走的通道叫做“主路（**bigstream**）”，主路分享即用摄像头通道分享屏幕。该模式下，主播只有一路上行视频流，要么上行摄像头画面，要么上行屏幕画面，两者是互斥的。您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeBig` 来启用该模式。

支持的平台

iOS	Android	Mac OS	Windows	Electron	Chrome 浏览器
✓	✓	✓	✓	✓	✓

获取分享目标

通过 getScreenCaptureSourcesWithThumbnailSize 可以枚举可共享的窗口列表，每一个可共享的目标都是一个 `TRTCScreenCaptureSourceInfo` 对象。

Mac OS 里的桌面屏幕也是一个可共享目标，普通的 Mac 窗口的 type 为 `TRTCScreenCaptureSourceTypeWindow`，桌面屏幕的 type 为 `TRTCScreenCaptureSourceTypeScreen`。

除了 type，每一个 `TRTCScreenCaptureSourceInfo` 还有如下字段信息：

字段	类型	含义

字段	类型	含义
type	TRTCScreenCaptureSourceType	采集源类型：指定类型为窗口或屏幕
sourceId	NSString	采集源 ID：对于窗口，该字段指示窗口句柄； 对于屏幕，该字段指示屏幕 ID
sourceName	NSString	窗口名字，如果是屏幕则返回 Screen0 Screen1...
extInfo	NSDictionary	共享窗口的附加信息
thumbnail	NSImage	窗口缩略图
icon	NSImage	窗口图标

有了上面这些信息，您就可以实现一个简单的列表页面，将可以分享的目标罗列出来供用户选择。

选择分享目标

TRTC SDK 支持三种分享模式，您可以通过 selectScreenCaptureTarget 来指定：

- **整个屏幕分享**：
  即把整个屏幕窗口分享出去，支持多显示器分屏的情况。需要指定一个 type 为 `TRTCScreenCaptureSourceTypeScreen` 的 screenSource 参数，并将 rect 设为 { 0, 0, 0, 0 }。

- **指定区域分享**：
  即把屏幕的某个区域分享出去，需要用户圈定区域的位置坐标。需要指定一个 type 为 `TRTCScreenCaptureSourceTypeScreen` 的 screenSource 参数，并将 captureRect 设为非 NULL，例如 { 100, 100, 300, 300 }。

- **指定窗口分享**：
  即把目标窗口的内容分享出去，需要用户选择要分享的是哪一个窗口。需要指定一个 type 为 `TRTCScreenCaptureSourceTypeWindow` 的 screenSource 参数，并将 captureRect 设为 { 0, 0, 0, 0 }。

> 说明：
> 两个额外参数：
>
> - 参数 capturesCursor 用于指定是否捕获鼠标指针。

- 参数 highlight 用于指定是否高亮正在共享的窗口，以及当捕获图像被遮挡时提示用户移走遮挡。（这一分部的 UI 特效是 SDK 内部实现的）

开始屏幕分享

- 选取分享目标之后，使用 startScreenCapture 接口可以启动屏幕分享。
- 两个函数 pauseScreenCapture 和 stopScreenCapture 的区别在于 pause 会停止屏幕内容的采集，并以暂停那一刻的画面垫片，所以在远端看到一直都是最后一帧画面，直到 resumeScreenCapture。

```
/**
* 7.6 【屏幕共享】启动屏幕分享
* @param view 渲染控件所在的父控件
*/
- (void)startScreenCapture:(NSView *)view;
/**
* 7.7 【屏幕共享】停止屏幕采集
* @return 0：成功 <0:失败
*/
- (int)stopScreenCapture;
/**
* 7.8 【屏幕共享】暂停屏幕分享
* @return 0：成功 <0:失败
*/
- (int)pauseScreenCapture;
/**
* 7.9 【屏幕共享】恢复屏幕分享
*
* @return 0：成功 <0:失败
*/
- (int)resumeScreenCapture;
```

设定画面质量

您可以通过 setSubStreamEncoderParam 接口设定屏幕分享的画面质量，包括分辨率、码率和帧率，我们提供如下建议参考值：

清晰度级别	分辨率	帧率	码率
超高清（HD+）	1920 × 1080	10	800kbps

清晰度级别	分辨率	帧率	码率
高清（HD）	1280 × 720	10	600kbps
标清（SD）	960 × 720	10	400kbps

观看屏幕分享

- **观看 Mac / Windows 屏幕分享**
  当房间里有一个 Mac / Windows 用户启动了屏幕分享，会通过辅流进行分享。房间里的其他用户会通过 TRTCCloudDelegate 中的 onUserSubStreamAvailable 事件获得这个通知。
  希望观看屏幕分享的用户可以通过 startRemoteSubStreamView 接口来启动渲染远端用户辅流画面。

- **观看 Android / iOS 屏幕分享**
  若用户通过 Android / iOS 进行屏幕分享，会通过主流进行分享。房间里的其他用户会通过 TRTCCloudDelegate 中的 onUserVideoAvailable 事件获得这个通知。
  希望观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户主流画面。

```
//示例代码：观看屏幕分享的画面
- (void)onUserSubStreamAvailable:(NSString *)userId available:(BOOL)available {
if (available) {
[self.trtcCloud startRemoteSubStreamView:userId view:self.capturePreviewWindow.co
ntentView];
} else {
[self.trtcCloud stopRemoteSubStreamView:userId];
}
}
```

常见问题

一个房间里可以同时有多个人共享屏幕吗？
目前一个 TRTC 音视频房间只能有一路屏幕分享。

指定窗口分享（SourceTypeWindow），当窗口大小变化时，视频流的分辨率会不会也跟着变化？
默认情况下，SDK 内部会自动根据分享的窗口大小进行编码参数的调整。
如需固定分辨率，需调用 setSubStreamEncoderParam 接口设置屏幕分享的编码参数，或在调用 startScreenCapture 时指定对应的编码参数。

Web

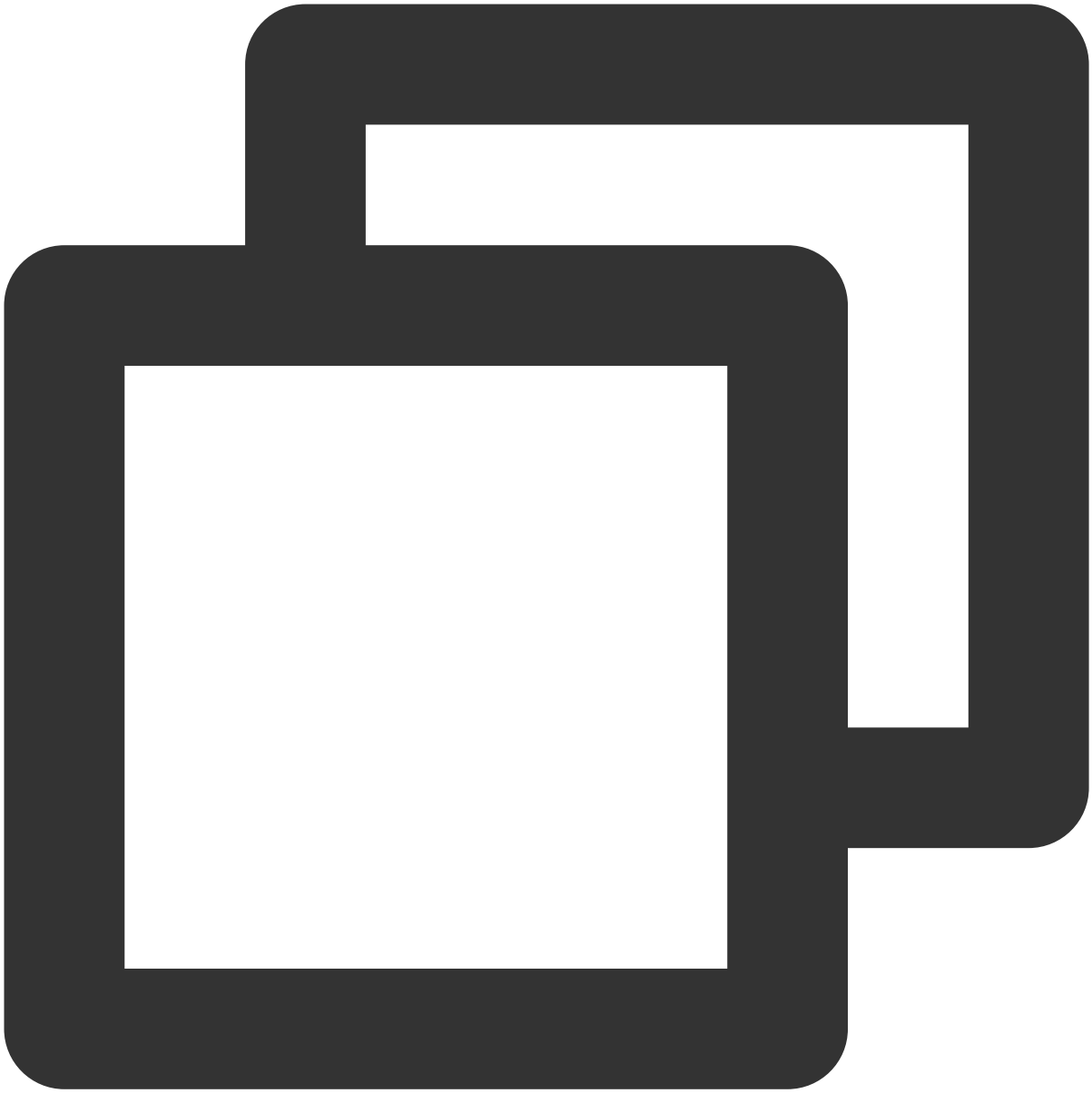
最近更新时间：2023-10-30 11:12:09

功能描述

本文主要介绍如何在 TRTC Web SDK 实现屏幕分享功能。

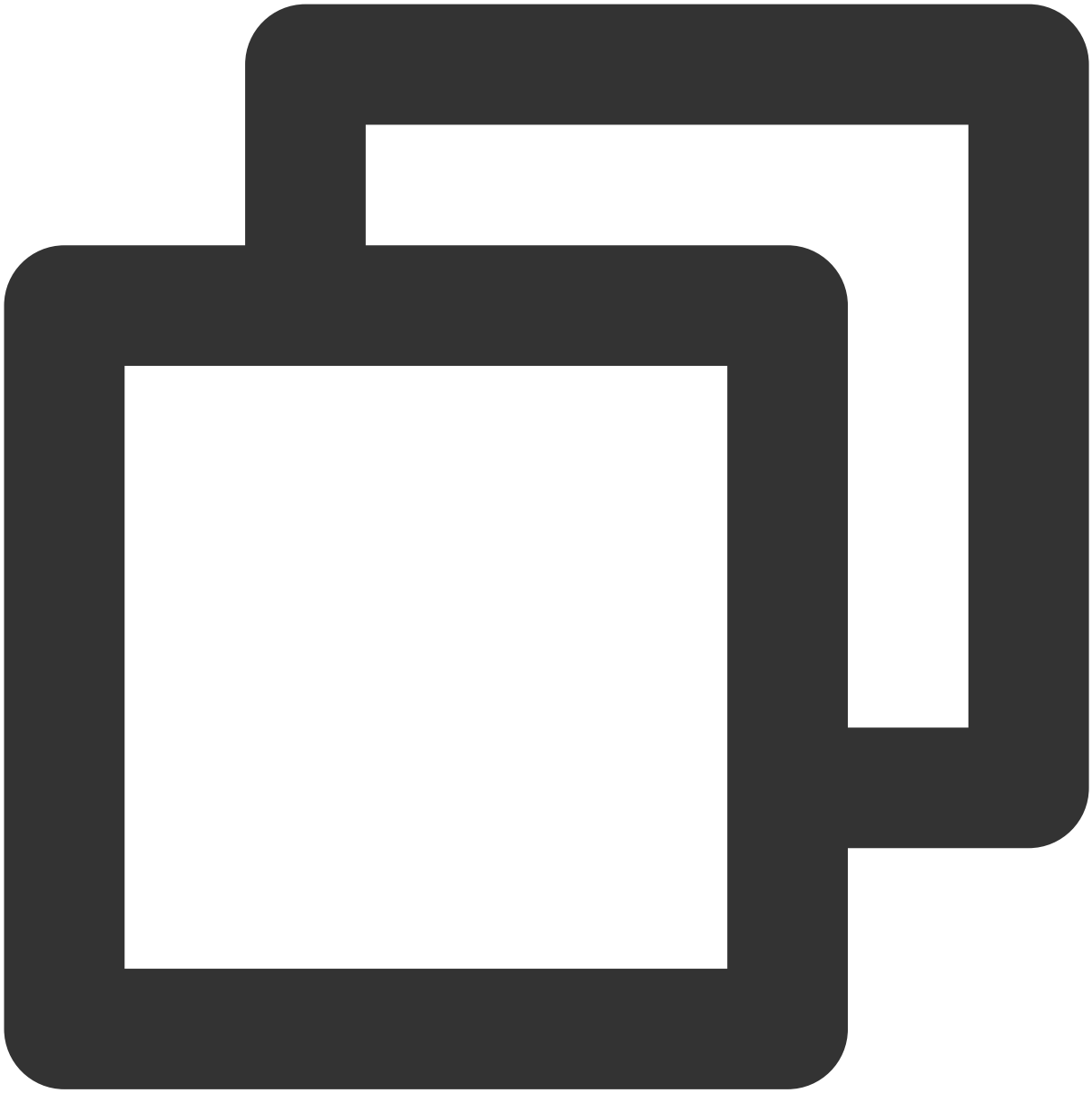
实现流程

1. 【推流端】开启屏幕分享



```
const trtcA = TRTC.create();
await trtcA.enterRoom({
  scene: 'rtc',
  sdkAppId: 140000000, // 填写您的 sdkAppId
  userId: 'userA', // 填写您的 userId
  userSig: 'userA_sig', // 填写 userId 对应的 userSig
  roomId: 6969
})
await trtcA.startScreenShare();
```

2. 【拉流端】播放屏幕分享

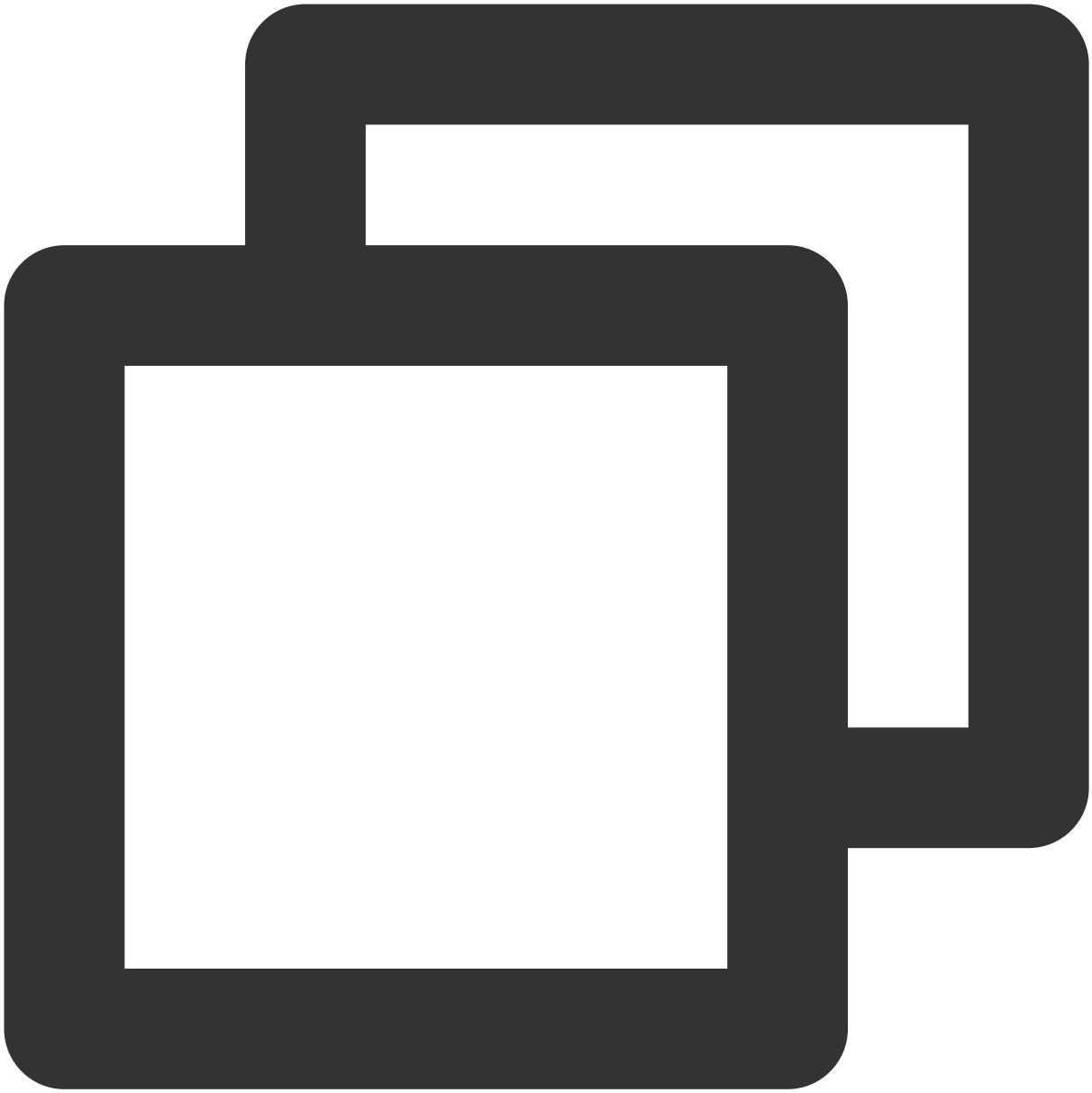


```
const trtcB = TRTC.create();
trtcB.on(TRTC.EVENT.REMOTE_VIDEO_AVAILABLE, ({ userId, streamType }) => {
  // 主路视频流，一般是推摄像头的那路流
  if (streamType === TRTC.TYPE.STREAM_TYPE_MAIN) {
    // 1. 在页面中放置一个 id 为 `${userId}_main` 的 div 标签，用于在 div 标签内播放主路流。
    // 2. 播放主路视频流
    trtcB.startRemoteVideo({ userId, streamType,  view: `${userId}_main` });
  } else {
    // 辅路视频流，一般是推屏幕分享的那路流。
    // 1. 在页面中放置一个 id 为 `${userId}_screen` 的 div 标签，用于在 div 标签内播放屏幕分
    // 2. 播放屏幕分享
```

```
    trtcB.startRemoteVideo({ userId, streamType, view: `${userId}_screen` });
  }
});

await trtcB.enterRoom({
  scene: 'rtc',
  sdkAppId: 140000000, // 填写您的 sdkAppId
  userId: 'userB', // 填写您的 userId
  userSig: 'userB_sig', // 填写 userId 对应的 userSig
  roomId: 6969
})
```

3. 同时推摄像头 + 屏幕分享



```
await trtcA.startLocalVideo();
await trtcA.startScreenShare();
```

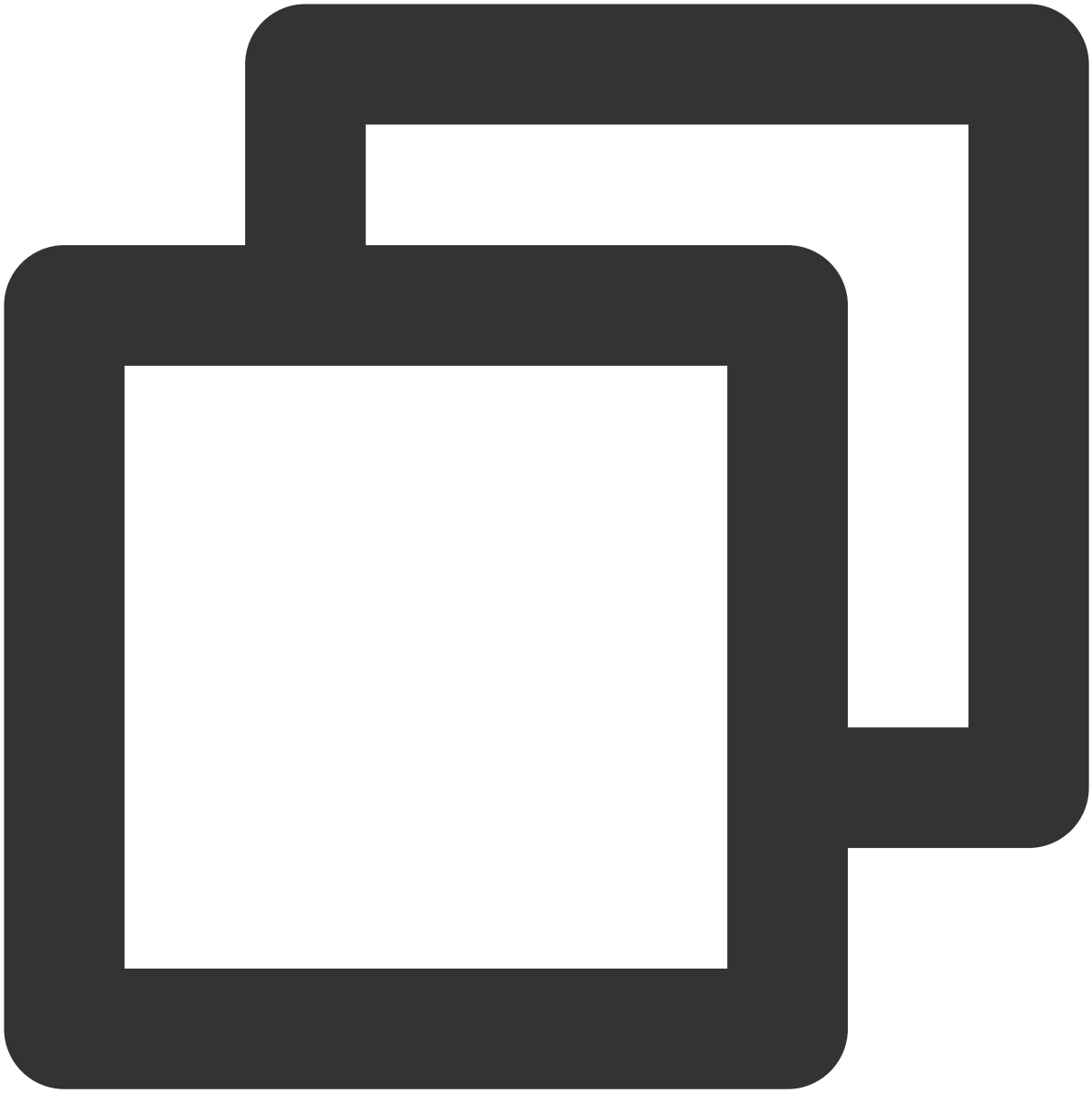
4. 屏幕分享 + 系统音频

采集系统音频支持 Chrome M74+

在 Windows 和 Chrome OS 上，可以采集整个系统的音频。

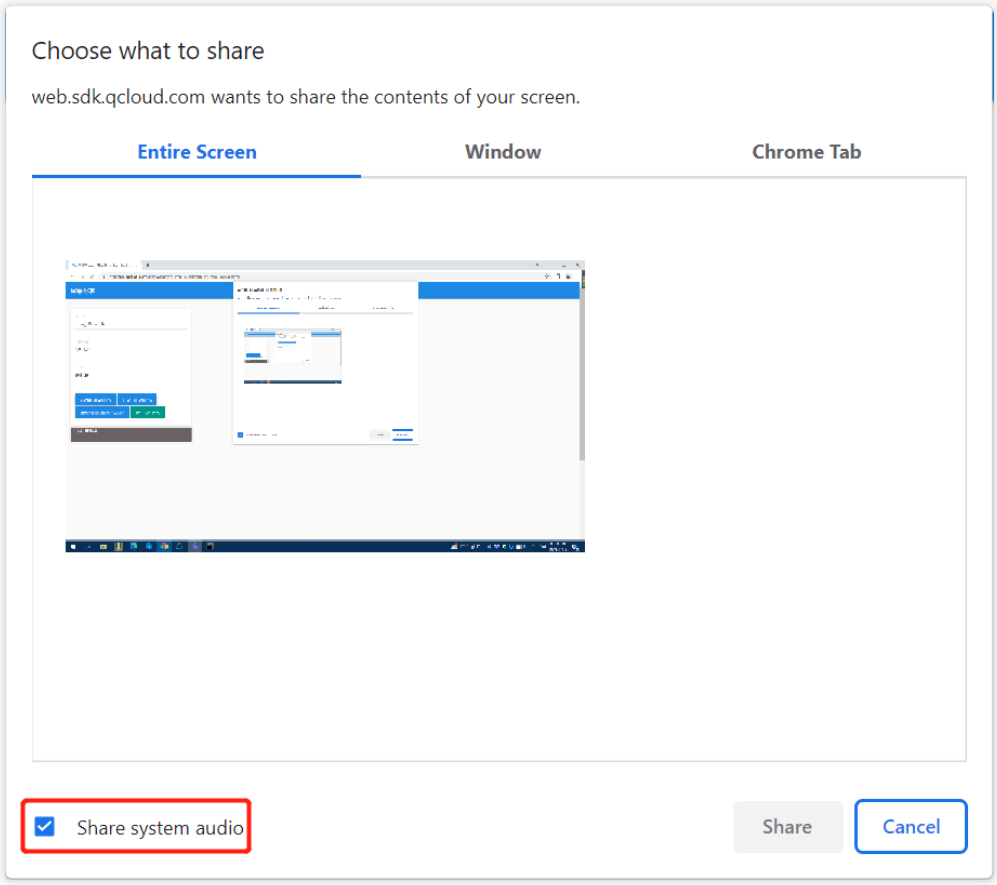
在 Linux 和 Mac 上，只能采集某个页面的音频。

其它 Chrome 版本、其它系统、其它浏览器均不支持。

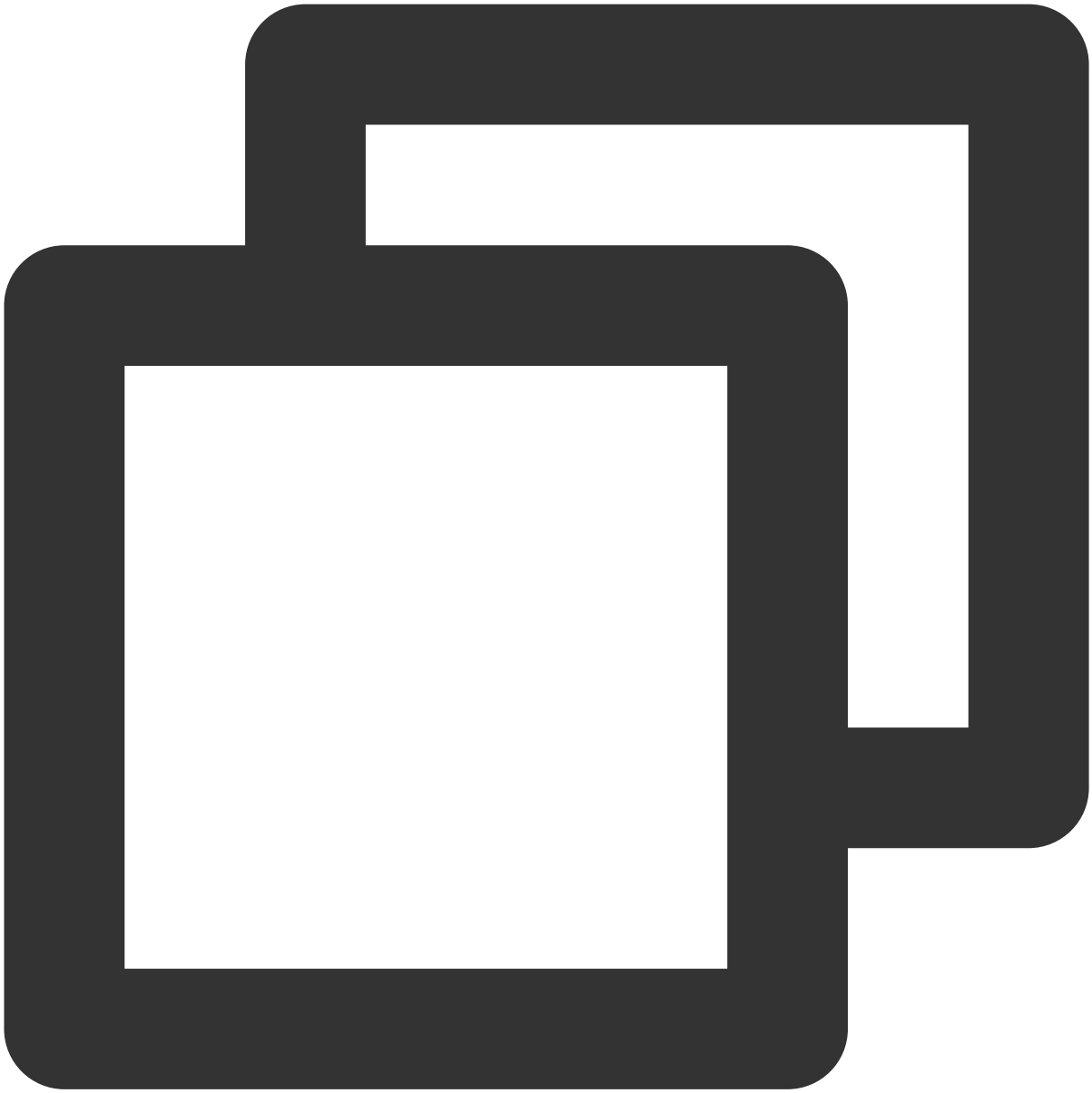


```
await trtcA.startScreenShare({ option: { systemAudio: true }});
```

在弹出的对话框中勾选分享音频，系统音频会与本地麦克风混音后发布，房间内其他用户会收到 TRTC.EVENT.REMOTE_AUDIO_AVAILABLE事件


Choose what to share
web.sdk.qcloud.com wants to share the contents of your screen.
Entire Screen
Window
Chrome Tab
Share system audio
Share
Cancel

5. 停止屏幕分享

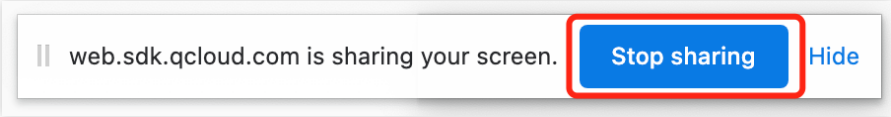


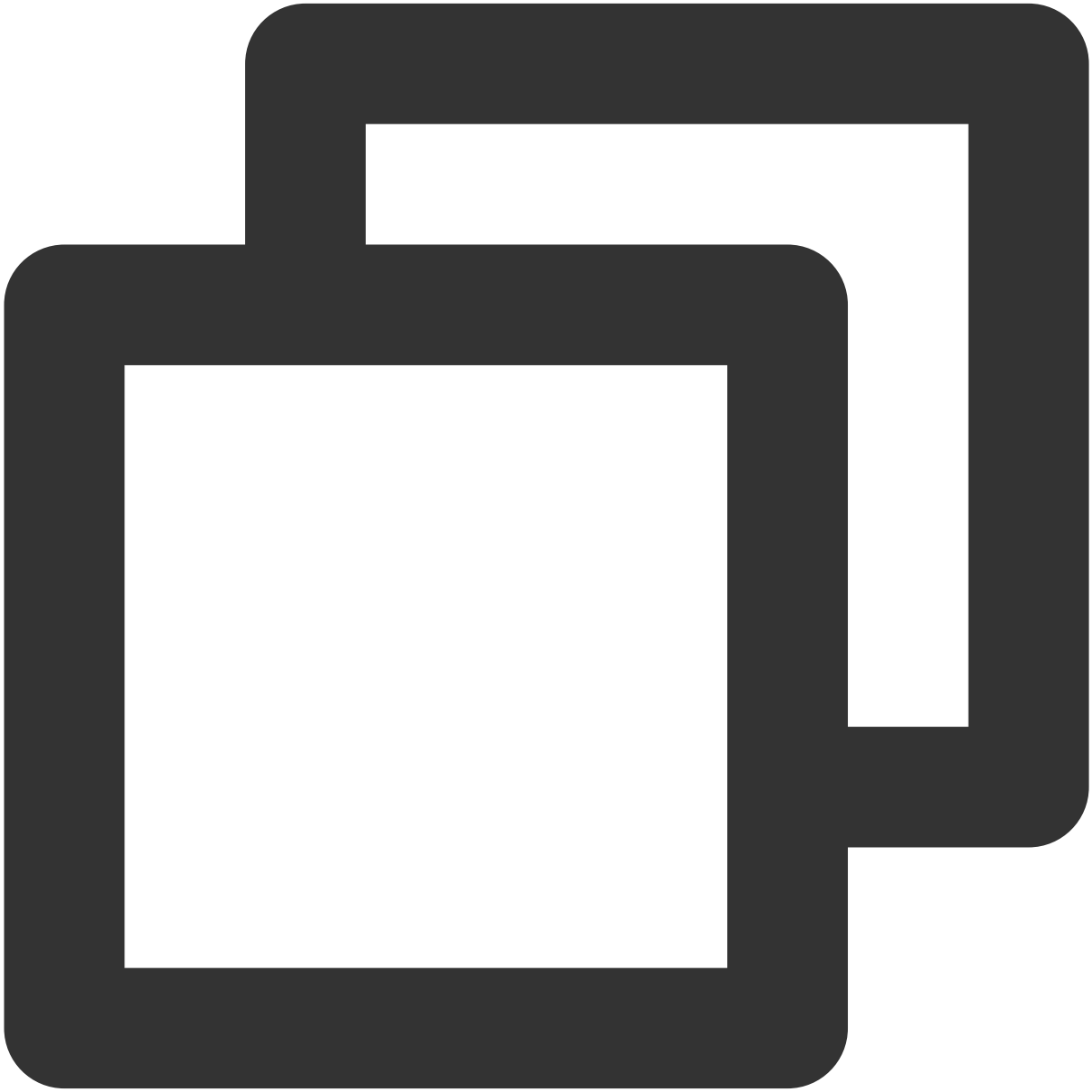
```
// 停止屏幕分享采集及发布
await trtcA.stopScreenShare();

// 房间内的其他用户会收到 TRTC.EVENT.REMOTE_VIDEO_UNAVAILABLE 事件, streamType 是 TRTC.
trtcB.on(TRTC.EVENT.REMOTE_VIDEO_UNAVAILABLE, ({ userId, streamType }) => {
    if (streamType === TRTC.TYPE.STREAM_TYPE_SUB) {

    }
})
```

另外用户还可能会通过浏览器自带的按钮停止屏幕分享，因此屏幕分享流需要监听屏幕分享停止事件，并进行相应的处理。





```
// 监听屏幕分享停止事件
```

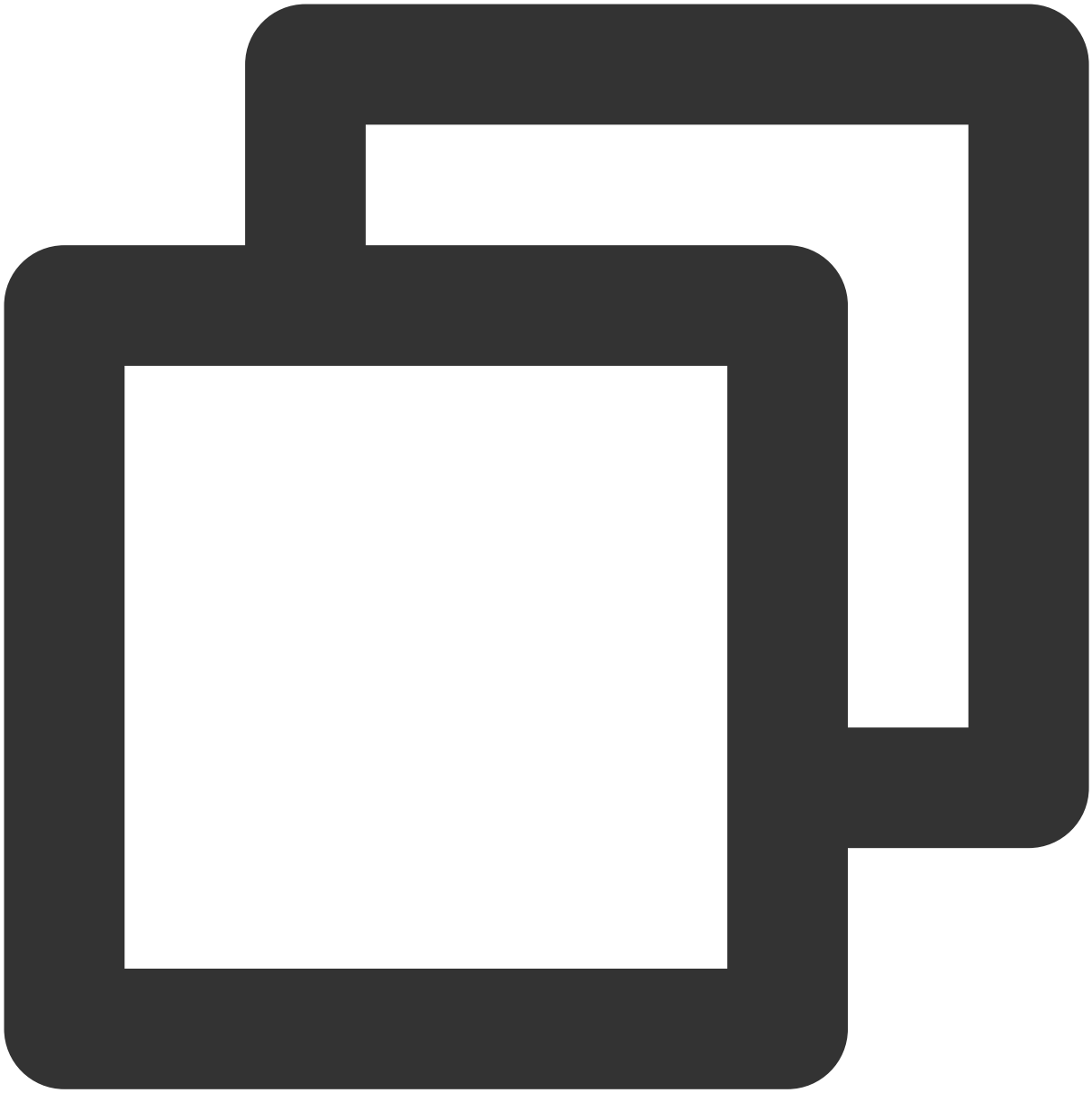
```
trtcA.on(TRTC.EVENT.SCREEN_SHARE_STOPPED, () => {
  console.log('screen sharing was stopped');
});
```

注意事项

1. 什么是主流，辅流？
2. SDK 默认使用 1080p 参数配置来采集屏幕分享，具体参考接口： `TRTC.startScreenShare` 。

常见问题

1. Safari 屏幕分享出现报错 `getDisplayMedia must be called from a user gesture handler`
这是因为 Safari 限制了 `getDisplayMedia` 屏幕采集的接口，必须在用户点击事件的回调函数执行的 1 秒内才可以调用。
参考：webkit issue。



```
// good
async function onClick() {
  // 建议在 onClick 执行时，先执行采集逻辑
  await trtcA.startScreenShare();
  await trtcA.enterRoom({
    roomId: 123123,
    sdkAppId: 140000000, // 填写您的 sdkAppId
    userId: 'userA', // 填写您的 userId
    userSig: 'userA_sig', // 填写 userId 对应的 userSig });
});
```

```
// bad
async function onClick() {
  await trtcA.enterRoom({
    roomId: 123123,
    sdkAppId: 140000000, // 填写您的 sdkAppId
    userId: 'userA', // 填写您的 userId
    userSig: 'userA_sig', // 填写 userId 对应的 userSig });
  })
  // 进房可能耗时超过 1s，可能会采集失败
   await trtcA.startScreenShare();
}
```

2. Mac Chrome 在已授权屏幕录制的情况下屏幕分享失败，出现 "NotAllowedError: Permission denied by system" 或者 "NotReadableError: Could not start video source" 错误信息，Chrome bug。解决方案：打开【设置】> 点击【安全性与隐私】> 点击【隐私】> 点击【屏幕录制】> 关闭 Chrome 屏幕录制授权 > 重新打开 Chrome 屏幕录制授权 > 关闭 Chrome 浏览器 > 重新打开 Chrome 浏览器。

3. WebRTC 屏幕分享已知问题及规避方案

Windows

最近更新时间：2022-10-14 17:17:53

本文档主要介绍如何使用屏幕分享，目前一个 TRTC 音视频房间只能有一路屏幕分享。

Windows 平台下的屏幕分享支持主路分享和辅路分享两种方案：

- **辅路分享**
  在 TRTC 中，我们可以单独为屏幕分享开启一路上行的视频流，并称之为“辅路（**substream**）”。辅路分享即主播同时上行摄像头画面和屏幕画面两路画面。这是腾讯会议的使用方案，您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeSub` 来启用该模式。

- **主路分享**
  在 TRTC 中，我们一般把摄像头走的通道叫做“主路（**bigstream**）”，主路分享即用摄像头通道分享屏幕。该模式下，主播只有一路上行视频流，要么上行摄像头画面，要么上行屏幕画面，两者是互斥的。您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeBig` 来启用该模式。

依赖的 API

API 功能	C++ 版本	C# 版本	Electron 版本
选择分享目标	selectScreenCaptureTarget	selectScreenCaptureTarget	selectScreenCaptureTarget
开始屏幕分享	startScreenCapture	startScreenCapture	startScreenCapture
暂停屏幕分享	pauseScreenCapture	pauseScreenCapture	pauseScreenCapture
恢复屏幕分享	resumeScreenCapture	resumeScreenCapture	resumeScreenCapture
结束屏幕分享	stopScreenCapture	stopScreenCapture	stopScreenCapture

获取分享目标

通过 `getScreenCaptureSources` 可以枚举可共享的窗口列表，列表通过出参 sourceInfoList 返回。

版权所有：腾讯云计算（北京）有限责任公司

> 说明：
> Windows 里的桌面屏幕也是一个窗口，叫桌面窗口（Desktop），有两台显示器时，每一台显示器都有一个对应的桌面窗口。所以，getScreenCaptureSources 返回的窗口列表里也会有 Desktop 窗口。

根据获取到的窗口信息，您可以实现一个简单的列表页面，将可以分享的目标罗列出来供用户选择。

开始屏幕分享

- 选取分享目标后，使用 `startScreenCapture` 接口可以启动屏幕分享。
- 分享过程中，您依然可以通过调用 `selectScreenCaptureTarget` 更换分享目标。
- `pauseScreenCapture` 和 `stopScreenCapture` 的区别在于 pause 会停止屏幕内容的采集，并以暂停那一刻的画面垫片，所以在远端看到一直都是最后一帧画面，直到 resume。

设定画面质量

您可以通过 `setSubStreamEncoderParam` 接口设定屏幕分享的画面质量，包括分辨率、码率和帧率，我们提供如下建议参考值：

清晰度级别	分辨率	帧率	码率
超高清（HD+）	1920 × 1080	10	800kbps
高清（HD）	1280 × 720	10	600kbps
标清（SD）	960 × 720	10	400kbps

观看屏幕分享

当房间里有一个用户启动了屏幕分享，会通过辅流进行分享。房间里的其他用户会通过 ITRTCCloudCallback 中的 onUserSubStreamAvailable 事件获得这个通知。
希望观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户辅流画面。

```
//示例代码：观看屏幕分享的画面
void CTRTCCloudSDK::onUserSubStreamAvailable(const char * userId, bool available)
{
LINFO(L"onUserSubStreamAvailable userId[%s] available[%d]\n", UTF82Wide(userId).c
```

```
_str(), available);
liteav::ITRTCCloud* trtc_cloud_ = getTRTCShareInstance();
if (available) {
trtc_cloud_->startRemoteView(userId, liteav::TRTCVideoStreamTypeSub, hWnd);
} else {
trtc_cloud_->stopRemoteView(userId, liteav::TRTCVideoStreamTypeSub);
}
}
```

常见问题

1. 一个房间里可以同时有多路屏幕分享吗？

目前一个 TRTC 音视频房间只能有一路屏幕分享。

2. 指定窗口分享（SourceTypeWindow），当窗口大小变化时，视频流的分辨率会不会也跟着变化？

默认情况下，SDK 内部会自动根据分享的窗口大小进行编码参数的调整。
如需固定分辨率，需调用 setSubStreamEncoderParam 接口设置屏幕分享的编码参数，或在调用 startScreenCapture 时指定对应的编码参数。

Electron

最近更新时间：2022-07-21 16:57:07

本文档主要介绍如何使用屏幕分享，目前一个 TRTC 音视频房间只能有一路屏幕分享。

Electron 平台下的屏幕分享支持主路分享和辅路分享两种方案：

- **辅路分享**
  在 TRTC 中，我们可以单独为屏幕分享开启一路上行的视频流，并称之为“辅路（**substream**）”。辅路分享即主播同时上行摄像头画面和屏幕画面两路画面。这是腾讯会议的使用方案，您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeSub` 来启用该模式。

- **主路分享**
  在 TRTC 中，我们一般把摄像头走的通道叫做“主路（**bigstream**）”，主路分享即用摄像头通道分享屏幕。该模式下，主播只有一路上行视频流，要么上行摄像头画面，要么上行屏幕画面，两者是互斥的。您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeBig` 来启用该模式。

步骤1：获取分享目标

通过 `getScreenCaptureSources` 可以枚举可共享的窗口列表，列表通过出参 sourceInfoList 返回。

> 说明：
> Electron 里的桌面屏幕也是一个窗口，叫桌面窗口（Desktop），有两台显示器时，每一台显示器都有一个对应的桌面窗口。所以，getScreenCaptureSources 返回的窗口列表里也会有 Desktop 窗口。

根据获取到的窗口信息，您可以实现一个简单的列表页面，将可以分享的目标罗列出来供用户选择。

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
// https://web.sdk.qcloud.com/trtc/electron/doc/en-us/trtc_electron_sdk/TRTCCloud.html#getScreenCaptureSources
const screenList = rtcCloud.getScreenCaptureSources();
```

步骤2：开始屏幕分享

- 可以通过调用 selectScreenCaptureTarget 选取分享目标。
- 选取分享目标后，使用 startScreenCapture 接口可以启动屏幕分享。
- 分享过程中，您依然可以通过调用 selectScreenCaptureTarget 更换分享目标。
- pauseScreenCapture 和 stopScreenCapture 的区别在于 pause 会停止屏幕内容的采集，并以暂停那一刻的画面垫片，所以在远端看到一直都是最后一帧画面，直到 resume。

```
import TRTCCloud, {
Rect, TRTCScreenCaptureProperty, TRTCVideoStreamType, TRTCVideoEncParam,
TRTCVideoResolution, TRTCVideoResolutionMode
} from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCCloud.html#getScreenCaptureSources
const screenList = rtcCloud.getScreenCaptureSources();
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/Rect.html
const captureRect = new Rect(0, 0, 0, 0);
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCScreenCaptureProperty.html
const property = new TRTCScreenCaptureProperty(
true, true, true, 0, 0, false
);
if (screenList.length > 0) {
rtcCloud.selectScreenCaptureTarget(screenList[0], captureRect, property)
}
const screenshareDom = document.querySelector('screen-dom');
// https://web.sdk.qcloud.com/trtc/electron/doc/zh-cn/trtc_electron_sdk/TRTCVideoEncParam.html
const encParam = new TRTCVideoEncParam(
TRTCVideoResolution.TRTCVideoResolution_1920_1080,
TRTCVideoResolutionMode.TRTCVideoResolutionModeLandscape,
15,
2000,
0,
false
);
rtcCloud.startScreenCapture(screenshareDom, TRTCVideoStreamType.TRTCVideoStreamTypeSub, encParam);
```

步骤3：设定画面质量

您可以通过 `startScreenCapture` 接口的第三个参数 `encParam` 设定屏幕分享的画面质量(参见 步骤2)，包括分辨率、码率和帧率，我们提供如下建议参考值：

清晰度级别	分辨率	帧率	码率
超高清（HD+）	1920 × 1080	10	2000kbps
高清（HD）	1280 × 720	10	600kbps
标清（SD）	960 × 720	10	400kbps

步骤4：观看屏幕分享

当房间里有一个用户启动了屏幕分享，会通过辅流进行分享。房间里的其他用户会通过 onUserSubStreamAvailable 事件获得这个通知。
希望观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户辅流画面。

```
import TRTCCloud, {
TRTCVideoStreamType
} from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
const remoteDom = document.querySelector('.remote-user');
function onUserSubStreamAvailable(userId, available) {
if (available === 1) {
rtcCloud.startRemoteView(userId, remoteDom, TRTCVideoStreamType.TRTCVideoStreamTy
peSub);
} else {
rtcCloud.stopRemoteView(userId, TRTCVideoStreamType.TRTCVideoStreamTypeSub);
}
}
rtcCloud.on('onUserSubStreamAvailable', onUserSubStreamAvailable);
```

常见问题

1. 一个房间里可以同时有多路屏幕分享吗？

目前一个 TRTC 音视频房间只能有一路屏幕分享。

2. 指定窗口分享（SourceTypeWindow），当窗口大小变化时，视频流的分辨率会不会也跟着变化？

默认情况下，SDK 内部会自动根据分享的窗口大小进行编码参数的调整。
如需固定分辨率，需调用 setSubStreamEncoderParam 接口设置屏幕分享的编码参数，或在调用

startScreenCapture 时指定对应的编码参数。

Flutter

最近更新时间：2024-05-31 17:08:26

基于 Android 平台

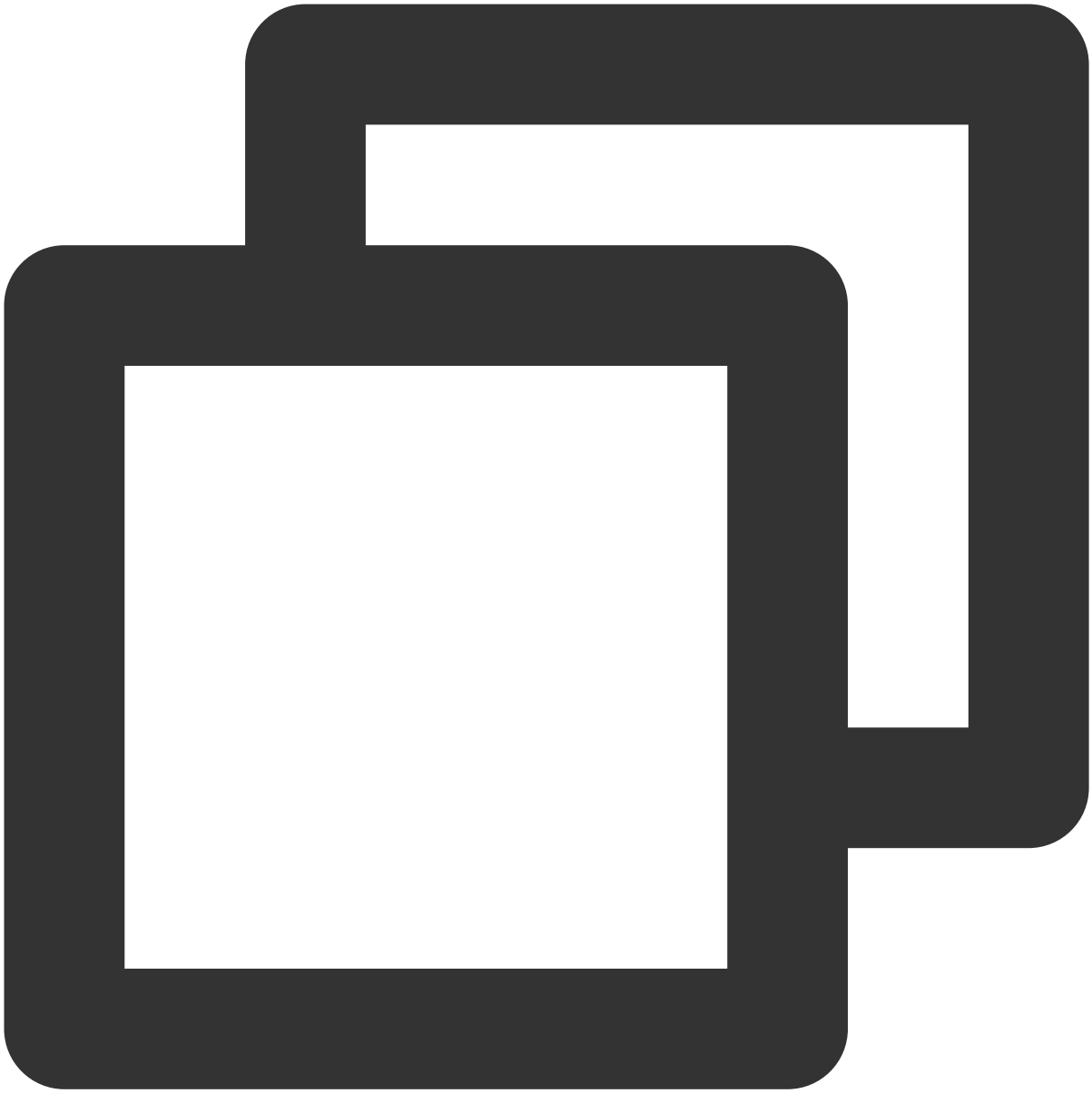
腾讯云 TRTC 在 Android 系统上支持屏幕分享，即将当前系统的屏幕内容通过 TRTC SDK 分享给房间里的其他用户。关于此功能，有两点需要注意：
移动端 TRTC Android 8.6 之前的版本屏幕分享并不像桌面端版本一样支持“辅路分享”，因此在启动屏幕分享时，摄像头的采集需要先被停止，否则会相互冲突；8.6 及之后的版本支持“辅路分享”，则不需要停止摄像头的采集。
当一个 Android 系统上的后台 App 在持续使用 CPU 时，很容易会被系统强行杀掉，而且屏幕分享本身又必然会消耗 CPU。要解决这个看似矛盾的冲突，我们需要在 App 启动屏幕分享的同时，在 Android 系统上弹出悬浮窗。由于 Android 不会强杀包含前台 UI 的 App 进程，因此该种方案可以让您的 App 可以持续进行屏幕分享而不被系统自动回收。

启动屏幕分享

要开启 Android 端的屏幕分享，只需调用 `TRTCCloud` 中的 startScreenCapture() 接口即可。但如果要达到稳定和清晰的分享效果，您需要关注如下三个问题：

添加 Activity

在 manifest 文件中粘贴如下 activity（若项目代码中存在则不需要添加）。



```
<activity
    android:name="com.tencent.rtmp.video.TXScreenCapture$TXScreenCaptureAssistantAc
    android:theme="@android:style/Theme.Translucent"/>
```

设定视频编码参数

通过设置 startScreenCapture() 中的首个参数 `encParams` ，您可以指定屏幕分享的编码质量。如果您指定 `encParams` 为 null，SDK 会自动使用之前设定的编码参数，我们推荐的参数设定如下：

参数项	参数名称	常规推荐值	文字教学场景

分辨率	videoResolution	1280 × 720	1920 × 1080
帧率	videoFps	10 FPS	8 FPS
最高码率	videoBitrate	1600 kbps	2000 kbps
分辨率自适应	enableAdjustRes	NO	NO

说明：

由于屏幕分享的内容一般不会剧烈变动，所以设置较高的 FPS 并不经济，推荐10 FPS即可。

如果您要分享的屏幕内容包含大量文字，可以适当提高分辨率和码率设置。

最高码率（videoBitrate）是指画面在剧烈变化时的最高输出码率，如果屏幕内容变化较少，实际编码码率会比较低。

基于 iOS 平台

应用内分享

即只能分享当前 App 的画面，该特性需要 iOS 13 及以上版本的操作系统才能支持。由于无法分享当前 App 之外的屏幕内容，因此适用于对隐私保护要求高的场景。

跨应用分享

基于苹果的 Replaykit 方案，能够分享整个系统的屏幕内容，但需要当前 App 额外提供一个 Extension 扩展组件，因此对接步骤也相对应用内分享要多一点。

方案1：iOS 平台应用内分享

应用内分享的方案非常简单，只需要调用 TRTC SDK 提供的接口 startScreenCapture 并传入编码参数 `TRTCVideoEncParam` 和 参数 `appGroup` 设置为 `''` 。 `TRTCVideoEncParam` 参数可以设置为 null，此时 SDK 会沿用开始屏幕分享之前的编码参数。

我们推荐的用于 iOS 屏幕分享的编码参数是：

参数项	参数名称	常规推荐值	文字教学场景
分辨率	videoResolution	1280 × 720	1920 × 1080
帧率	videoFps	10 FPS	8 FPS
最高码率	videoBitrate	1600 kbps	2000 kbps
分辨率自适应	enableAdjustRes	NO	NO

说明：

由于屏幕分享的内容一般不会剧烈变动，所以设置较高的 FPS 并不经济，推荐10 FPS即可。

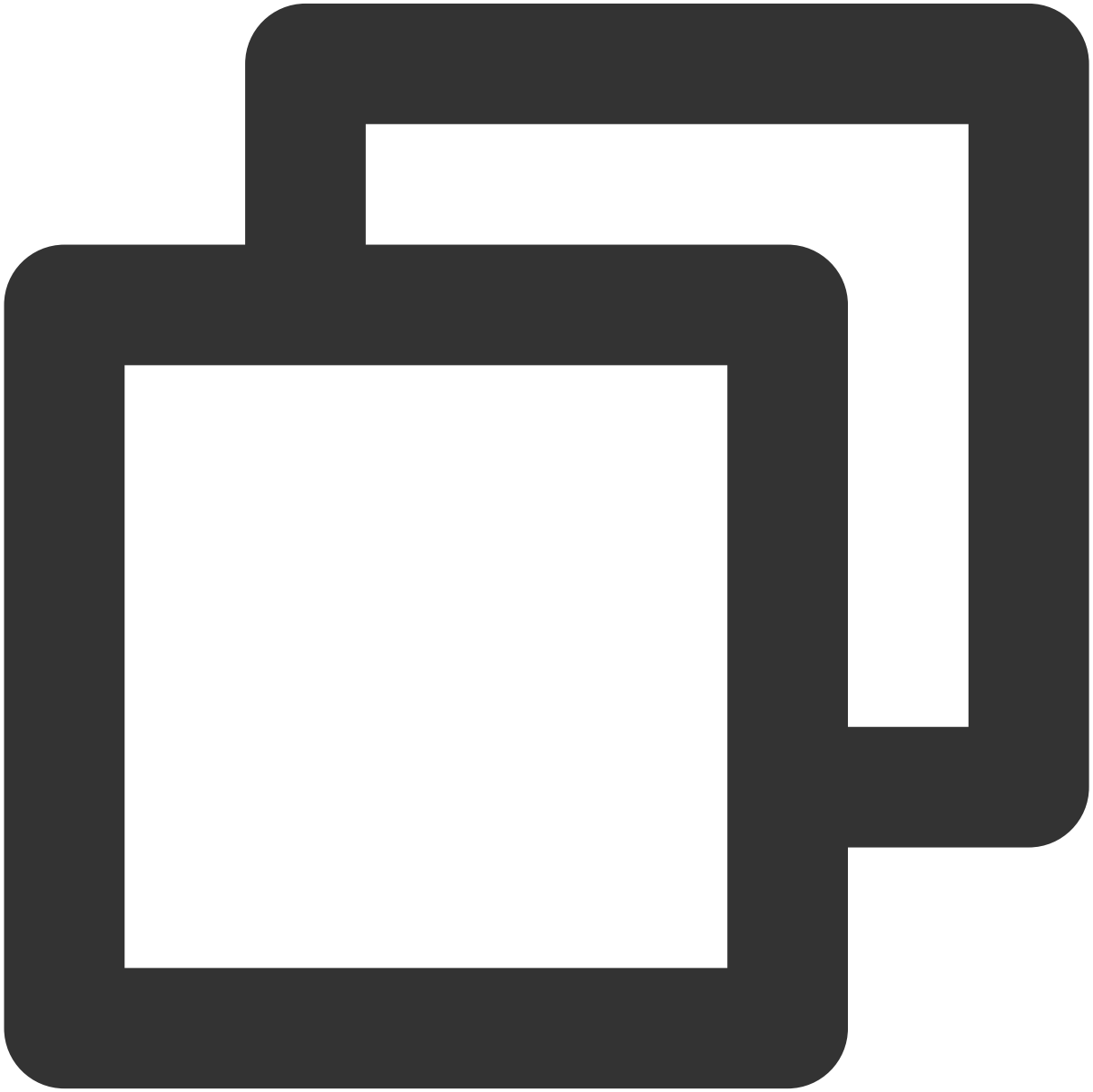
版权所有：腾讯云计算（北京）有限责任公司

如果您要分享的屏幕内容包含大量文字，可以适当提高分辨率和码率设置。
最高码率（videoBitrate）是指画面在剧烈变化时的最高输出码率，如果屏幕内容变化较少，实际编码码率会比较低。

方案2：iOS 平台跨应用分享

示例代码

我们在 Github 中的 **trtc_demo/ios** 目录下放置了一份跨应用分享的示例代码，其包含如下一些文件：



```
├── Broadcast.Upload          //录屏进程 Broadcast Upload Extension 代码详见步骤2
```

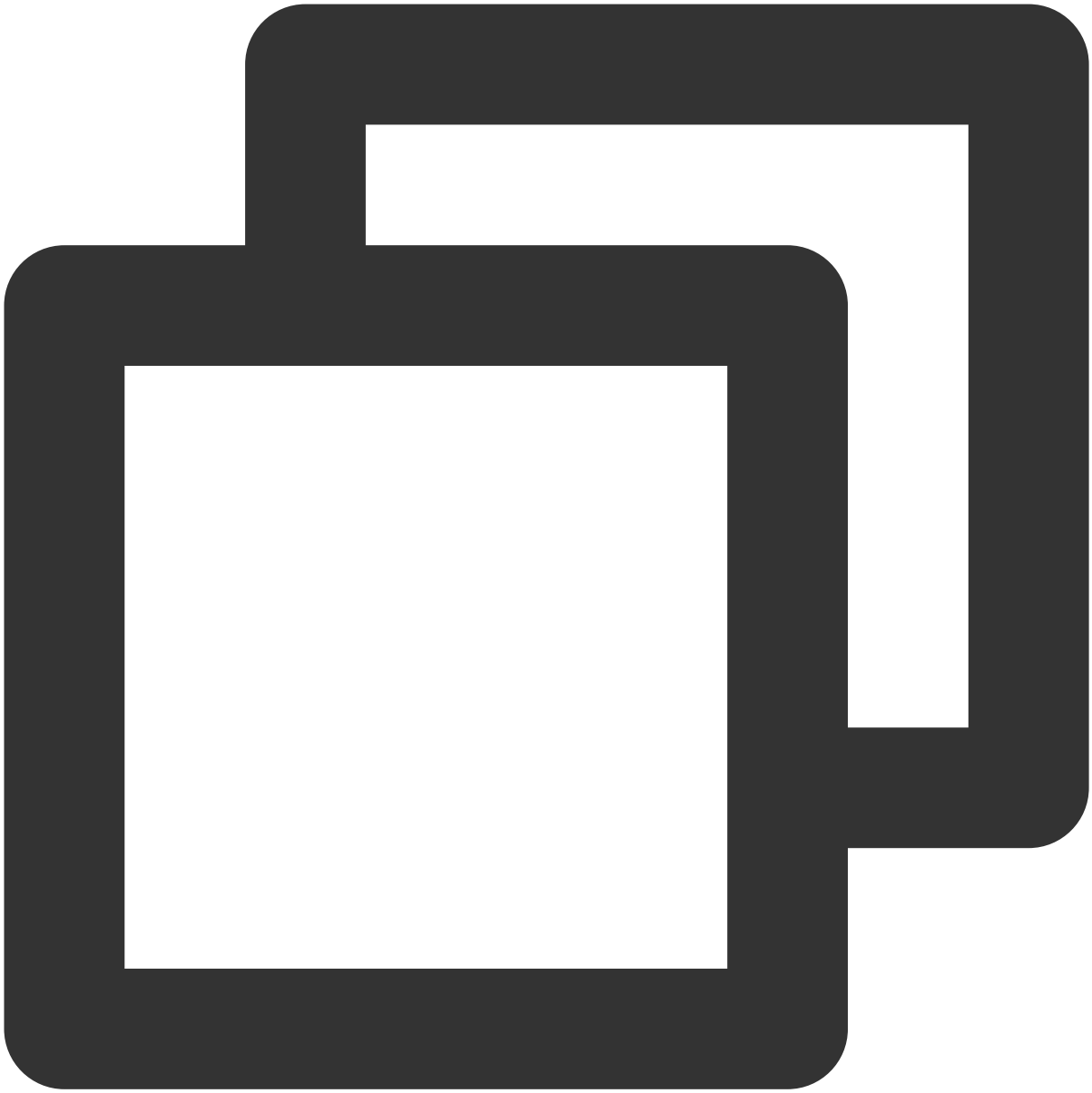
```
|   ├── Broadcast.Upload.entitlements        //用于设置进程间通信的 AppGroup 信息
|   ├── Broadcast.UploadDebug.entitlements   //用于设置进程间通信的 AppGroup 信息（debug
|   ├── Info.plist
|   └── SampleHandler.swift      // 用于接收来自系统的录屏数据
├── Resource                     // 资源文件
├── Runner                       // TRTC 精简化 Demo
├── TXLiteAVSDK_ReplayKitExt.framework      //TXLiteAVSDK_ReplayKitExt SDK
```

您可以通过 README 中的指引跑通该示例 Demo。

对接步骤

iOS 系统上的跨应用屏幕分享，需要增加 Extension 录屏进程以配合主 App 进程进行推流。Extension 录屏进程由系统在需要录屏的时候创建，并负责接收系统采集到屏幕图像。因此需要：

1. 创建 App Group，并在 XCode 中进行配置（可选）。这一步的目的是让 Extension 录屏进程可以同主 App 进程进行跨进程通信。
2. 在您的工程中，新建一个 Broadcast Upload Extension 的 Target，并在其中集成 SDK 压缩包中专门为扩展模块定制的 `TXLiteAVSDK_ReplayKitExt.framework` 。
3. 对接主 App 端的接收逻辑，让主 App 等待来自 Broadcast Upload Extension 的录屏数据。
4. 编辑 `pubspec.yaml` 文件引入 `replay_kit_launcher` 插件，实现类似TRTC Demo Screen中点击一个按钮即可唤起屏幕分享的效果（可选）。



```
# 引入 trtc sdk和replay_kit_launcher
dependencies:
  tencent_trtc_cloud: ^0.2.1
  replay_kit_launcher: ^0.2.0+1
```

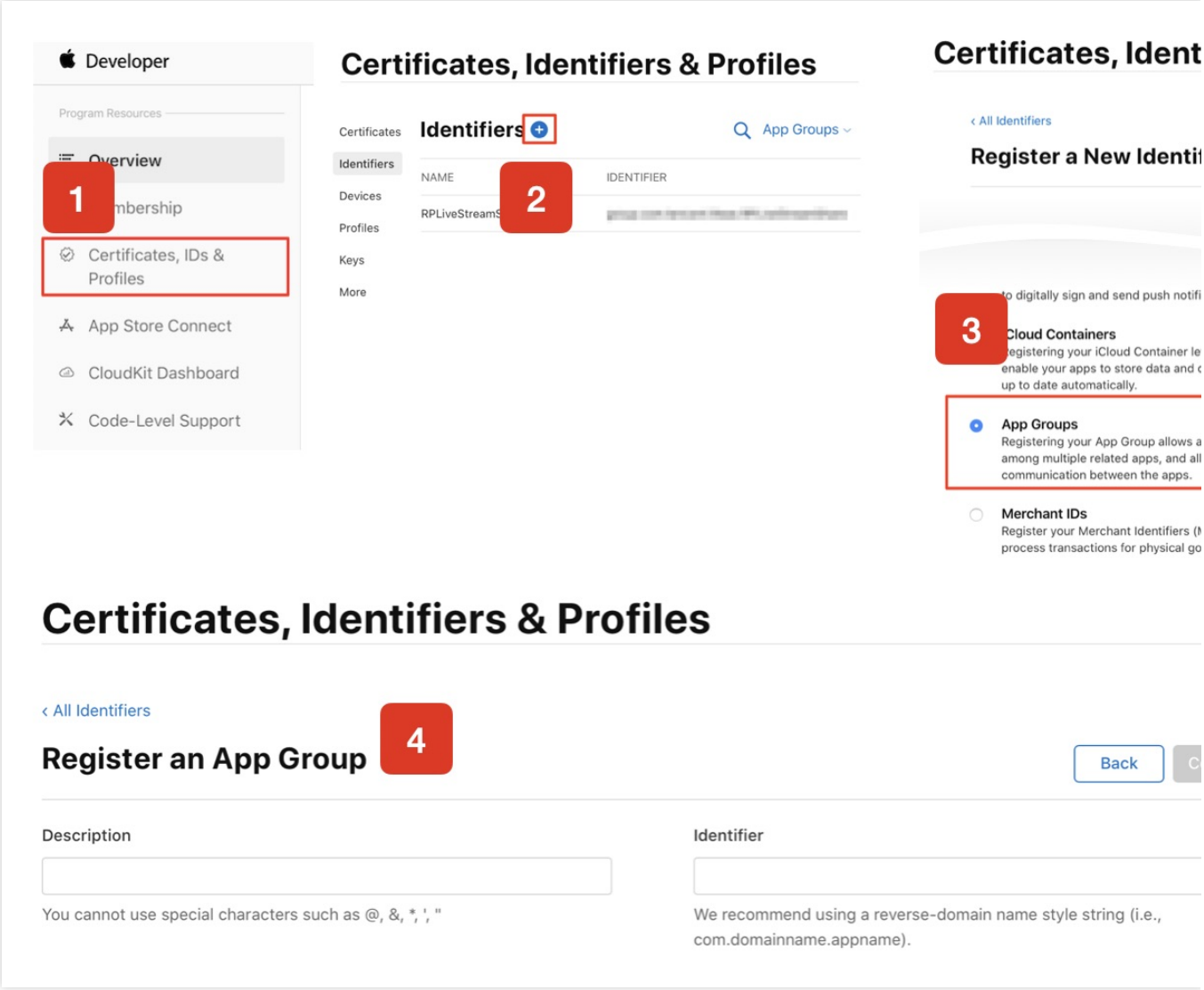
注意：

如果跳过 步骤1，也就是不配置 App Group（接口传 null），屏幕分享依然可以运行，但稳定性要打折扣，故虽然步骤较多，但请尽量配置正确的 App Group 以保障屏幕分享功能的稳定性。

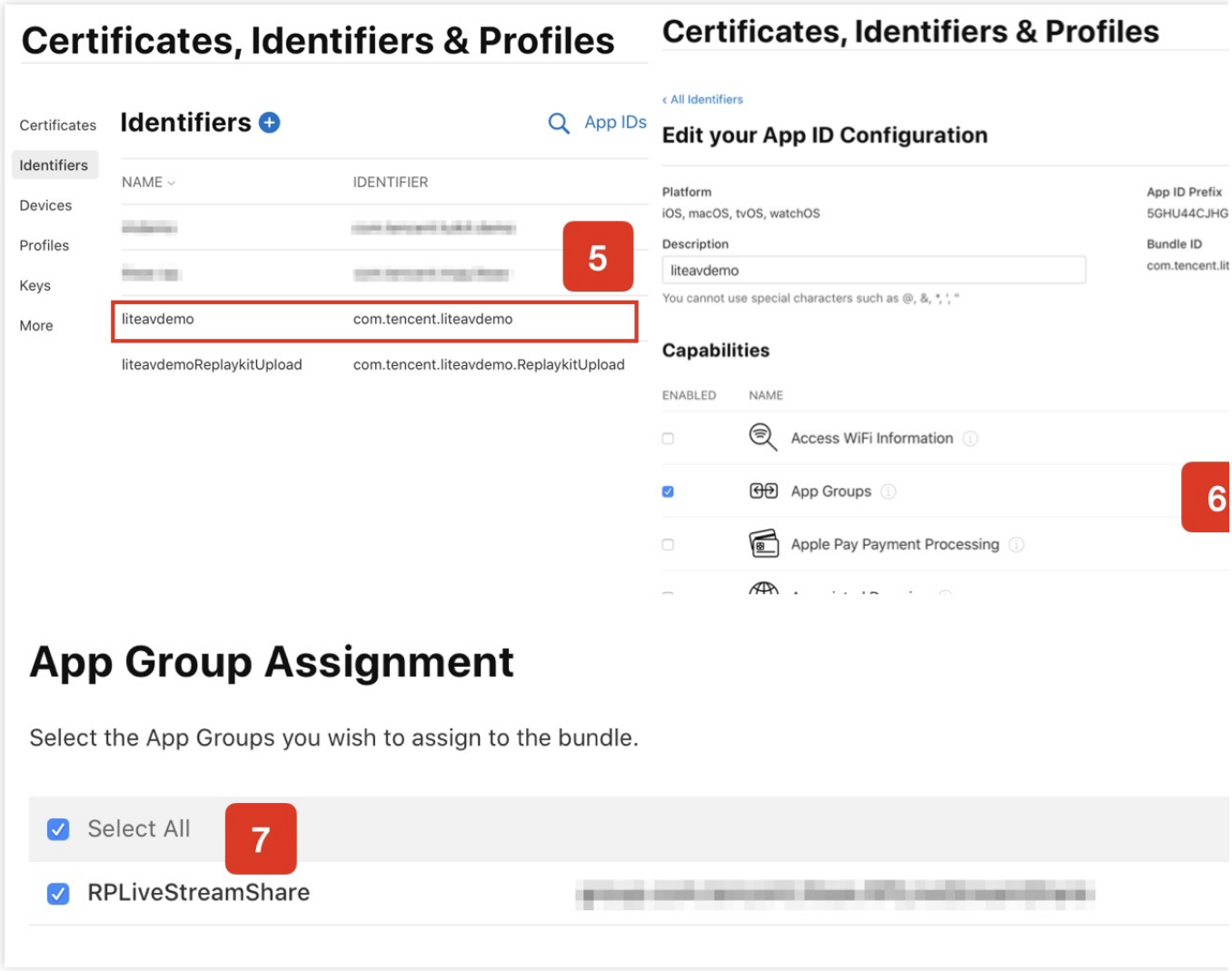
步骤1：创建 App Group

使用您的帐号登录 https://developer.apple.com/，进行以下操作，**注意完成后需要重新下载对应的 Provisioning Profile**。

1. 单击 **Certificates, IDs & Profiles**。
2. 在右侧的界面中单击加号。
3. 选择 **App Groups**，单击 **Continue**。
4. 在弹出的表单中填写 Description 和 Identifier, 其中 Identifier 需要传入接口中的对应的 AppGroup 参数。完成后单击 **Continue**。



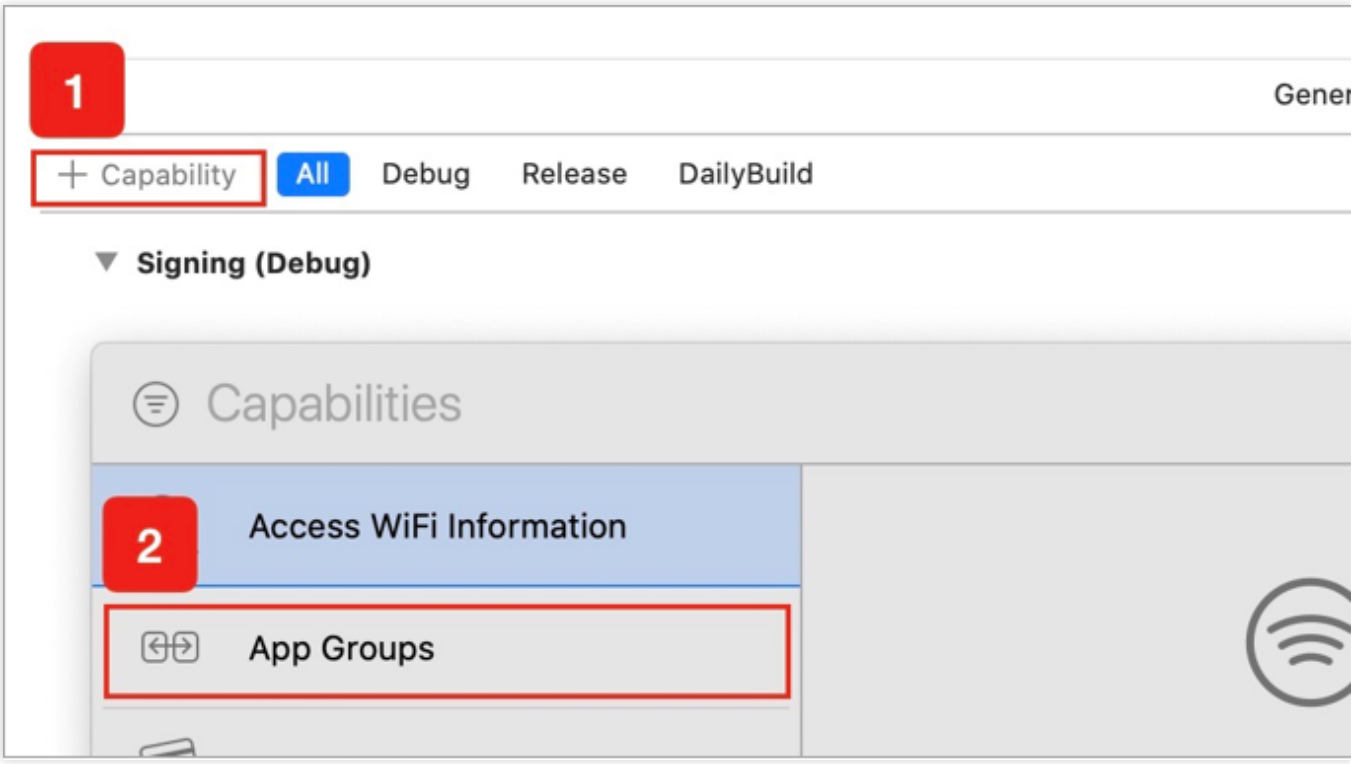
5. 回到 Identifier 页面，左上边的菜单中选择 **App IDs**，然后单击您的 App ID（主 App 与 Extension 的 AppID 需要进行同样的配置）。
6. 选中 **App Groups** 并单击 **Edit**。
7. 在弹出的表单中选择您之前创建的 App Group，单击 **Continue** 返回编辑页，单击 **Save** 保存。



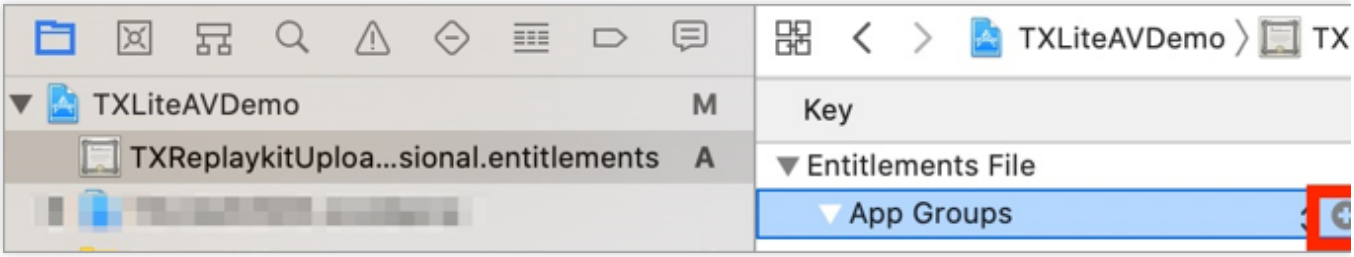
8. 重新下载 Provisioning Profile 并配置到 XCode 中。

步骤2：创建 Broadcast Upload Extension

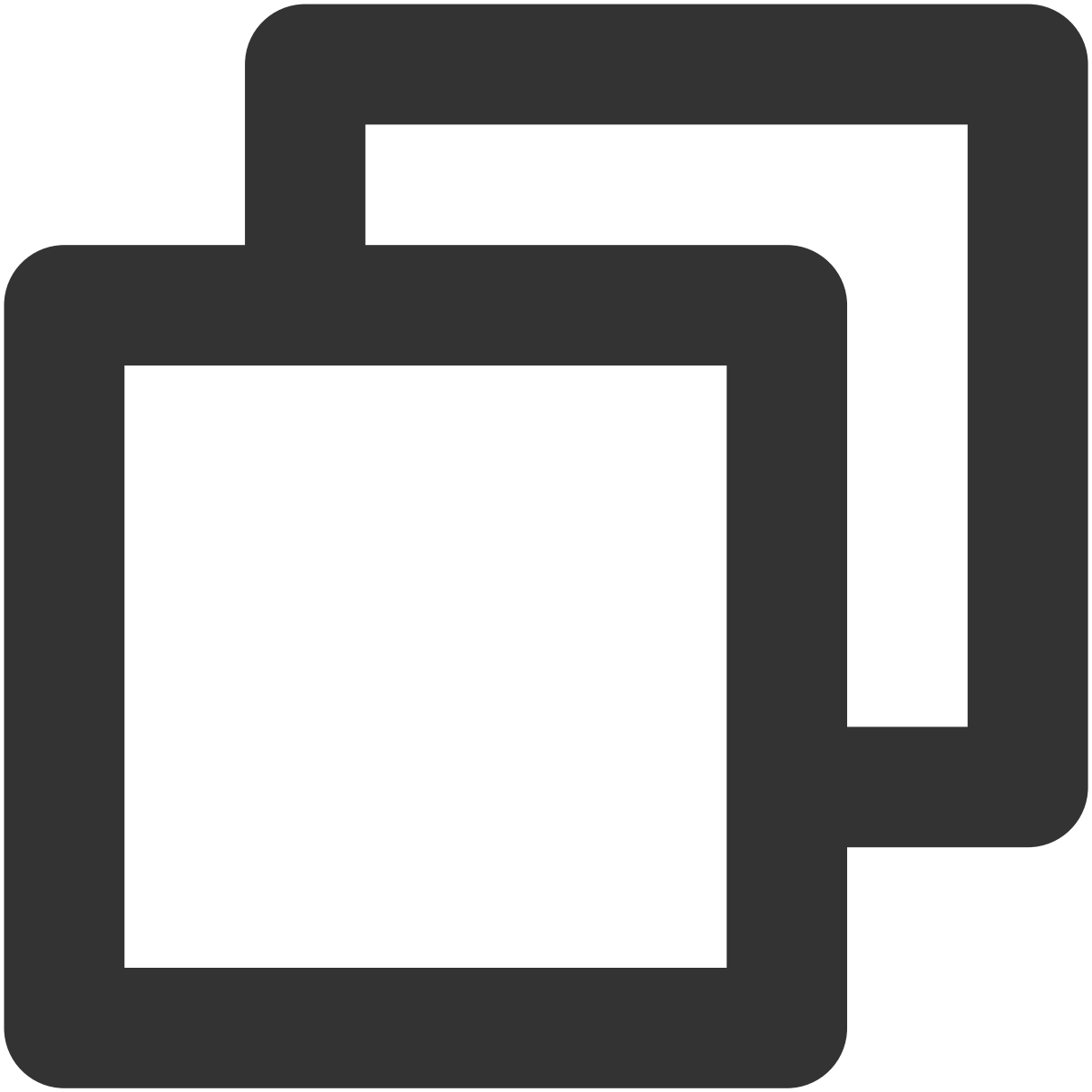
1. 在 Xcode 菜单依次单击 **File** > **New** > **Target...**，选择 **Broadcast Upload Extension**。
2. 在弹出的对话框中填写相关信息，**不用**勾选 **Include UI Extension**，单击 **Finish** 完成创建。
3. 将下载到的 SDK 压缩包中的 TXLiteAVSDK_ReplayKitExt.framework 拖动到工程中，勾选刚创建的 Target。
4. 选中新增加的 Target，依次单击**+ Capability**，双击 **App Groups**，如下图：



操作完成后，会在文件列表中生成一个名为 `Target名.entitlements` 的文件，如下图所示，选中该文件并单击 + 号填写上述步骤中的 App Group 即可。



5. 选中主 App 的 Target，**并按照上述步骤对主 App 的 Target 做同样的处理。**
6. 在新创建的 Target 中，Xcode 会自动创建一个名为 "SampleHandler.swift" 的文件，用如下代码进行替换。**需将代码中的 APPGROUP 改为上文中的创建的 App Group Identifier**。



```
import ReplayKit
import TXLiteAVSDK_ReplayKitExt


let APPGROUP = "group.com.tencent.comm.trtc.demo"

class SampleHandler: RPBroadcastSampleHandler, TXReplayKitExtDelegate {

    let recordScreenKey = Notification.Name.init("TRTCRecordScreenKey")
```

```
    override func broadcastStarted(withSetupInfo setupInfo: [String : NSObject]?) {
        // User has requested to start the broadcast. Setup info from the UI extens
        TXReplayKitExt.sharedInstance().setup(withAppGroup: APPGROUP, delegate: sel
    }

    override func broadcastPaused() {
        // User has requested to pause the broadcast. Samples will stop being deliv
    }

    override func broadcastResumed() {
        // User has requested to resume the broadcast. Samples delivery will resume
    }

    override func broadcastFinished() {
        // User has requested to finish the broadcast.
        TXReplayKitExt.sharedInstance() .finishBroadcast()
    }

    func broadcastFinished(_ broadcast: TXReplayKitExt, reason: TXReplayKitExtReaso
        var tip = ""
        switch reason {
        case TXReplayKitExtReason.requestedByMain:
            tip = "屏幕共享已结束"
            break
        case TXReplayKitExtReason.disconnected:
            tip = "应用断开"
            break
        case TXReplayKitExtReason.versionMismatch:
            tip = "集成错误（SDK 版本号不相符合）"
            break
        default:
            break
        }

        let error = NSError(domain: NSStringFromClass(self.classForCoder), code: 0,
        finishBroadcastWithError(error)
    }

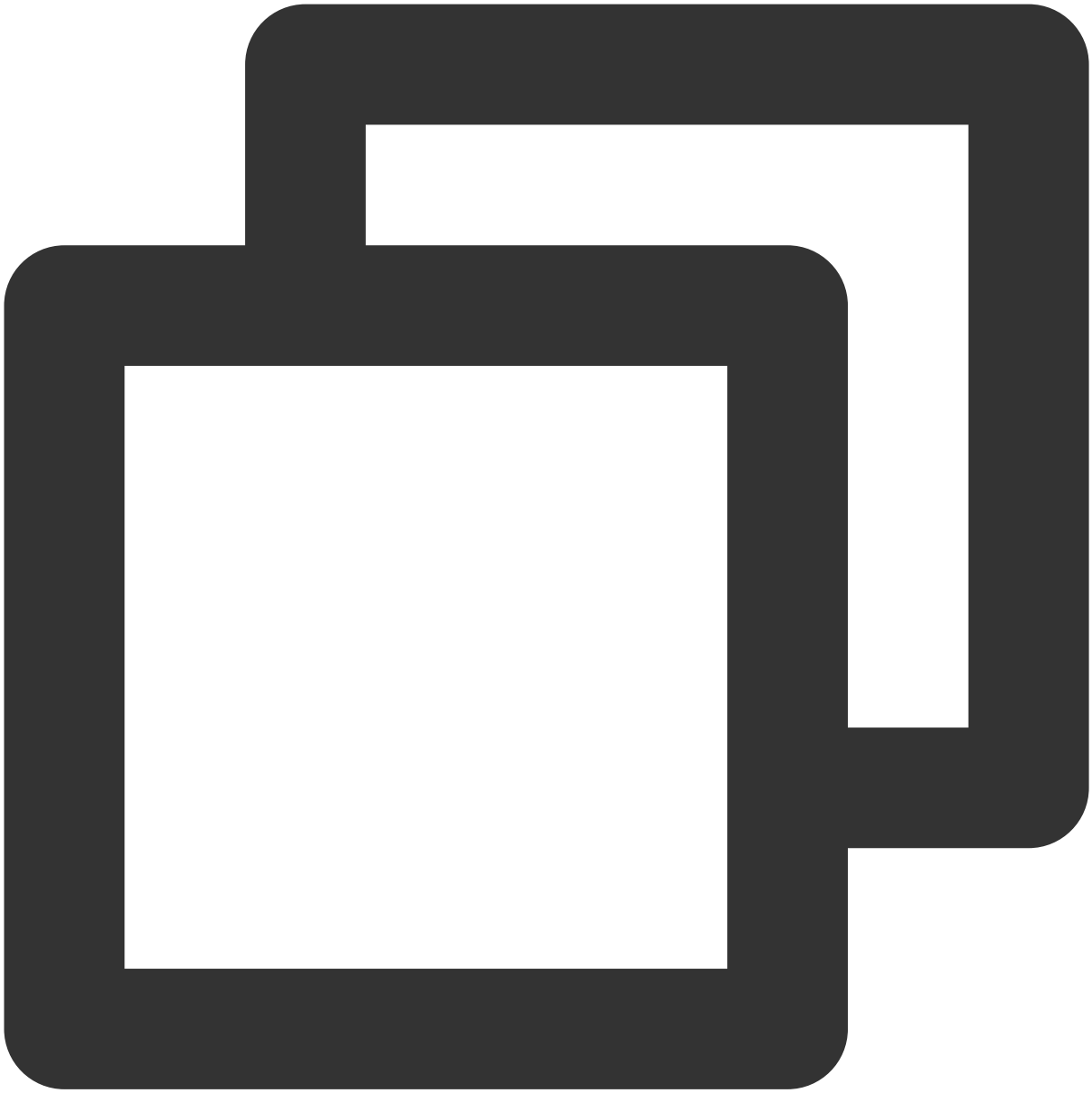
    override func processSampleBuffer(_ sampleBuffer: CMSampleBuffer, with sampleBu
        switch sampleBufferType {
        case RPSampleBufferType.video:
            // Handle video sample buffer
            TXReplayKitExt.sharedInstance() .sendVideoSampleBuffer(sampleBuffer)
            break
        case RPSampleBufferType.audioApp:
            // Handle audio sample buffer for app audio
            break
```

```
            case RPSampleBufferType.audioMic:
                // Handle audio sample buffer for mic audio
                break
            @unknown default:
                // Handle other sample buffer types
                fatalError("Unknown type of sample buffer")
            }
        }
    }
```

步骤3：对接主 App 端的接收逻辑

按照如下步骤，对接主 App 端的接收逻辑。也就是在用户触发屏幕分享之前，要让主 App 处于“等待”状态，以便随时接收来自 Broadcast Upload Extension 进程的录屏数据。

1. 确保 TRTCCloud 已经关闭了摄像头采集，如果尚未关闭，请调用 stopLocalPreview 关闭摄像头采集。
2. 调用 startScreenCapture 方法，并传入 步骤1 中设置的 AppGroup，让 SDK 进入“等待”状态。
3. 等待用户触发屏幕分享。如果不实现 步骤4 中的“触发按钮”，屏幕分享就需要用户在 iOS 系统的控制中心，通过长按录屏按钮来触发。
4. 通过调用 stopScreenCapture 接口可以随时中止屏幕分享。



```
// 开始屏幕分享，需要将 APPGROUP 替换为上述步骤中创建的 App Group
trtcCloud.startScreenCapture(
    TRTCVideoEncParam(
        videoFps: 10,
        videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_1280_720,
        videoBitrate: 1600,
        videoResolutionMode: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT,
    ),
    iosAppGroup,
);
```

版权所有：腾讯云计算（北京）有限责任公司

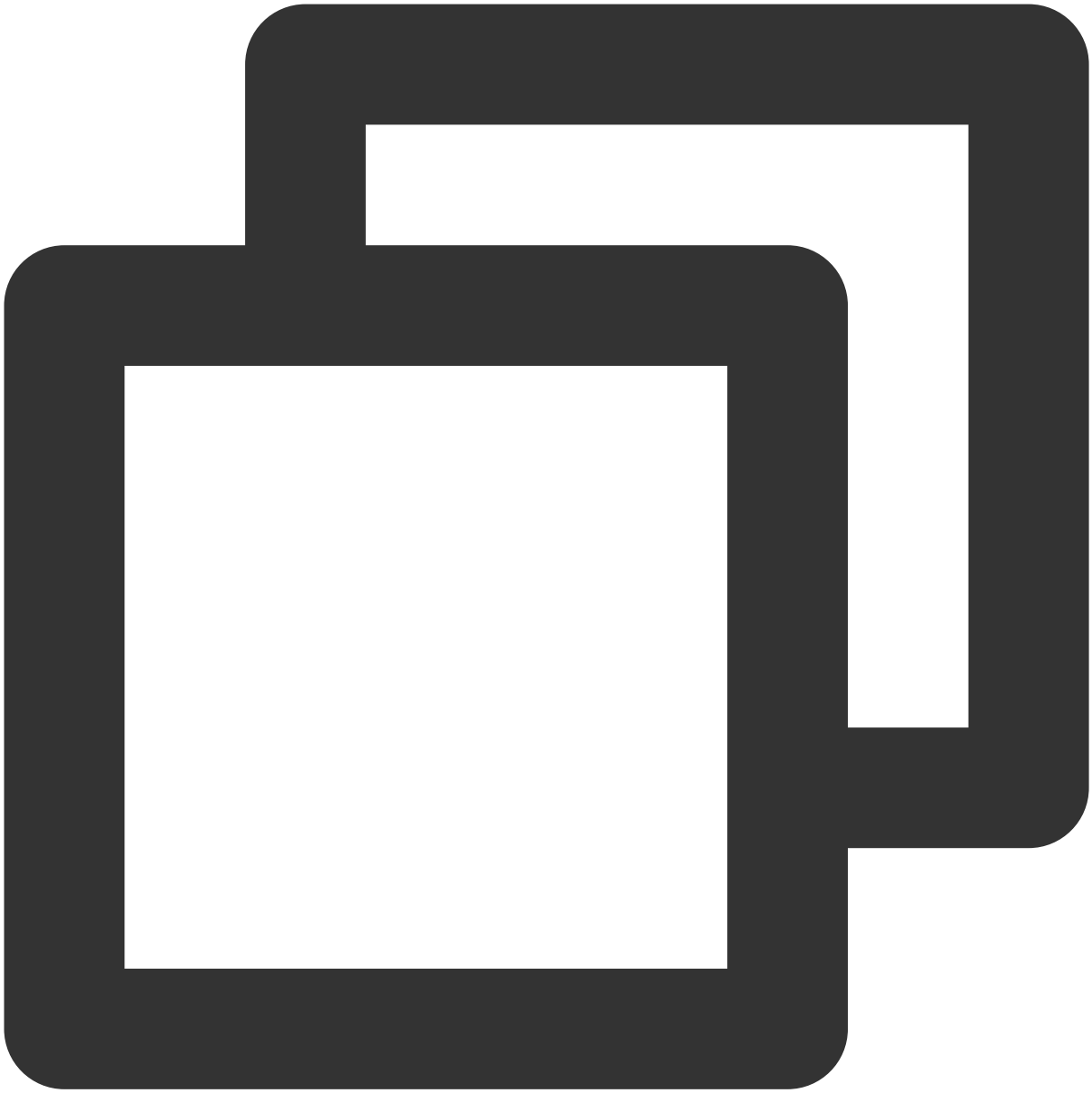
```
// 停止屏幕分享
await trtcCloud.stopScreenCapture();


// 屏幕分享的启动事件通知，可以通过 TRTCCloudListener 进行接收
onRtcListener(type, param){
    if (type == TRTCCloudListener.onScreenCaptureStarted) {
      //屏幕分享开始
    }
}
```

步骤4：增加屏幕分享的触发按钮（可选）

截止到 步骤3，我们的屏幕分享还必须要用户从控制中心中长按录屏按钮来手动启动。您可通过下述方法实现类似 TRTC Demo Screen 的单击按钮即可触发的效果。

1. 将 `replay_kit_launcher` 插件加入到您的工程中。
2. 在您的界面上放置一个按钮，并在按钮的响应函数中调用 `ReplayKitLauncher.launchReplayKitBroadcast(iosExtensionName);` 函数，就可以唤起屏幕分享功能了。



```
// 自定义按钮响应方法
onShareClick() async {
if (Platform.isAndroid) {
   if (await SystemAlertWindow.requestPermissions) {
     MeetingTool.showOverlayWindow();
   }
 } else {
   //屏幕分享功能只能在真机测试
   ReplayKitLauncher.launchReplayKitBroadcast(iosExtensionName);
 }
}
```

基于 Windows 平台

Windows 平台下的屏幕分享支持主路分享和辅路分享两种方案：

辅路分享

在 TRTC 中，我们可以单独为屏幕分享开启一路上行的视频流，并称之为“辅路（**substream**）”。辅路分享即主播同时上行摄像头画面和屏幕画面两路画面。这是腾讯会议的使用方案，您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeSub` 来启用该模式。

主路分享

在 TRTC 中，我们一般把摄像头走的通道叫做“主路（**bigstream**）”，主路分享即用摄像头通道分享屏幕。该模式下，主播只有一路上行视频流，要么上行摄像头画面，要么上行屏幕画面，两者是互斥的。您可以在调用 `startScreenCapture` 接口时，通过将 `TRTCVideoStreamType` 参数指定为 `TRTCVideoStreamTypeBig` 来启用该模式。

步骤一：获取分享目标

通过getScreenCaptureSources可以枚举可共享的窗口列表，列表通过出参 sourceInfoList 返回。

说明：

Windows 里的桌面屏幕也是一个窗口，叫桌面窗口（Desktop），有两台显示器时，每一台显示器都有一个对应的桌面窗口。所以，getScreenCaptureSources 返回的窗口列表里也会有 Desktop 窗口。

根据获取到的窗口信息，您可以实现一个简单的列表页面，将可以分享的目标罗列出来供用户选择。

步骤二：选择分享目标

当您通过 getScreenCaptureSources 获取到可以分享的屏幕和窗口之后，您可以调用 selectScreenCaptureTarget 接口选定期望分享的目标屏幕或目标窗口。

步骤三：开始屏幕分享

选取分享目标后，使用startScreenCapture 接口可以启动屏幕分享。

分享过程中，您依然可以通过调用selectScreenCaptureTarget更换分享目标。

pauseScreenCapture和 stopScreenCapture的区别在于 pause 会停止屏幕内容的采集，并以暂停那一刻的画面垫片，所以在远端看到一直都是最后一帧画面，直到 resume。

设定视频编码参数

通过设置 startScreenCapture() 中的首个参数 `encParams` ，您可以指定屏幕分享的编码质量,包括分辨率、码率和帧率，我们提供如下建议参考值：

清晰度级别	分辨率	帧率	码率
超高清（HD+）	1920 × 1080	10	800kbps

高清（HD）	1280 × 720	10	600kbps
标清（SD）	960 × 720	10	400kbps

如果您指定 `encParams` 为 null，SDK 会自动使用之前设定的编码参数。

观看屏幕分享

观看 Android / iOS 屏幕分享

若用户通过 Android / iOS 进行屏幕分享，会通过主流进行分享。房间里的其他用户会通过 TRTCCloudListener 中的 onUserVideoAvailable 事件获得这个通知。

希望观看屏幕分享的用户可以通过 startRemoteView 接口来启动渲染远端用户主流画面。

常见问题

一个房间里可以同时有多路屏幕分享吗？

目前一个 TRTC 音视频房间只能有一路屏幕分享。

分享系统声音

Web

最近更新时间：2023-10-30 11:12:58

功能描述

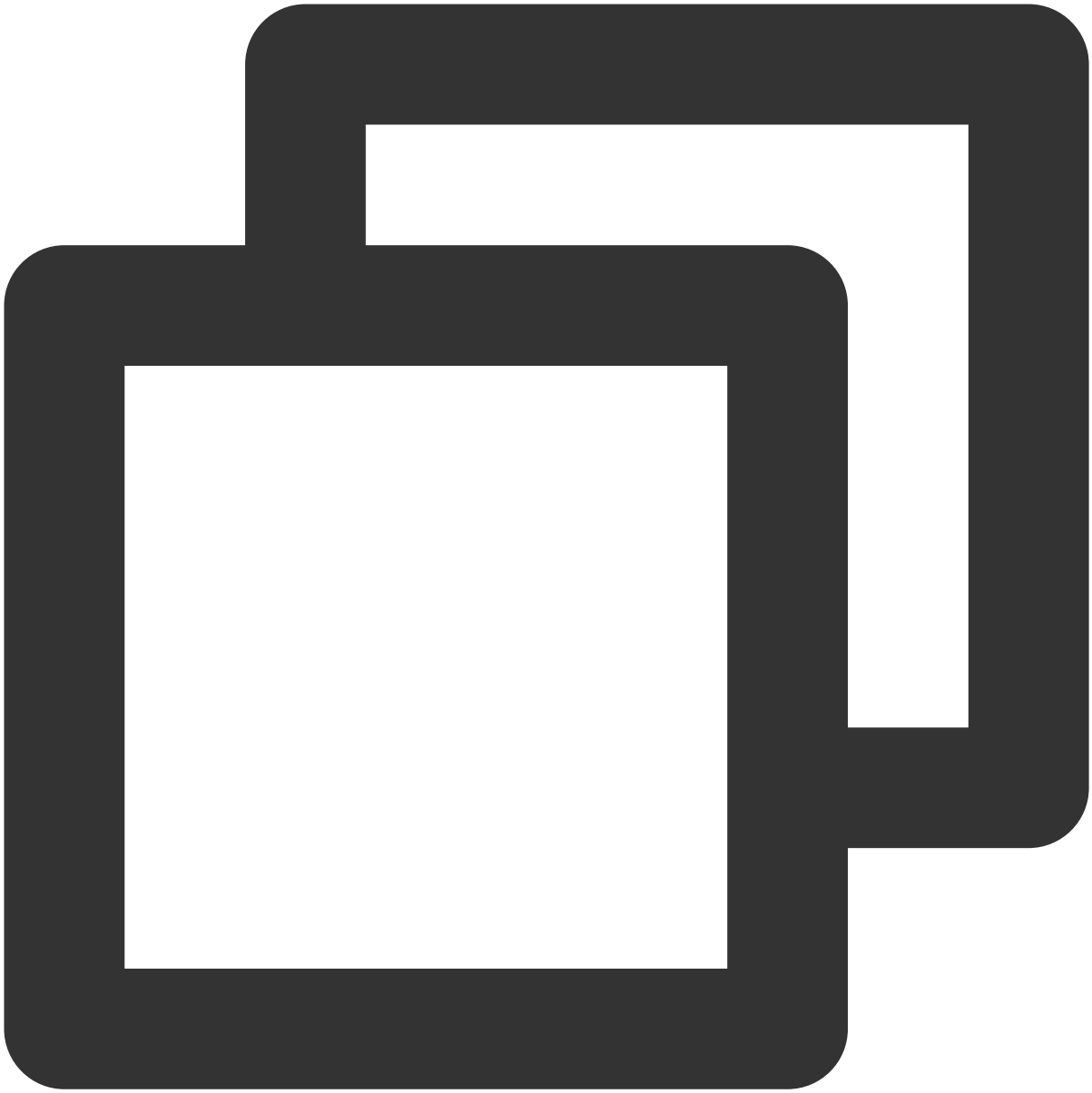
本文主要介绍如何在 TRTC Web SDK 实现分享系统音频功能。

实现流程

Web 端实现分享系统音频，需要搭配屏幕分享一起使用。无法脱离屏幕分享，单独实现分享系统音频功能。

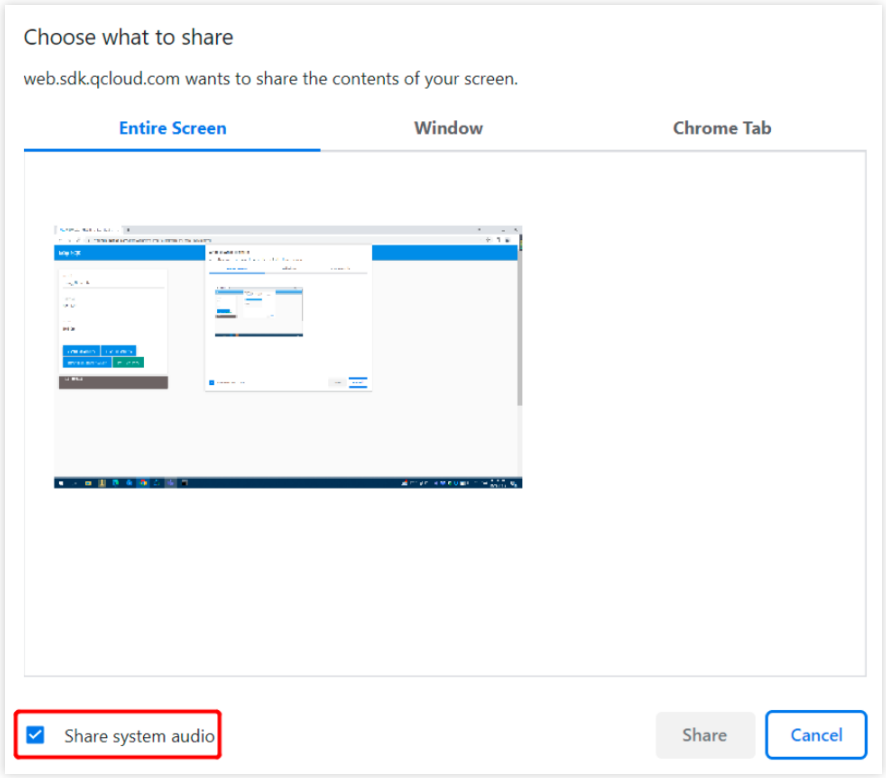
代码示例

版权所有：腾讯云计算（北京）有限责任公司



```
await trtcA.startScreenShare({ option: { systemAudio: true }});
```

在弹出的屏幕分享选择框中勾选 **Share system audio**，点击 **Share** 即可。在进房发布后，房间内其他用户会收到 TRTC.EVENT.REMOTE_AUDIO_AVAILABLE 事件。



注意：

若分享系统音频的同时采集了麦克风，则系统音频会与本地麦克风混音后发布。

兼容性说明

分享系统音频只支持基于 Chromium 74+ 版本的浏览器，例如 Chrome、Edge、Opera 等。其他浏览器暂不支持，例如：Safari、Firefox。

Windows & Chrome OS 的 Chrome 支持分享系统音频 + 某个浏览器页面音频。

MacOS & Linux 的 Chrome 只支持分享某个浏览器页面音频。

Android & iOS 的 Chrome 不支持。

版权所有：腾讯云计算（北京）有限责任公司

Mac

最近更新时间：2024-06-07 11:20:49

场景痛点及解决方案

在屏幕分享等应用场景中，常需要共享系统音频给对方，而 Mac 电脑默认声卡不支持采集系统音频，所以在 Mac 电脑上共享系统音频比较困难。基于此，TRTC 提供了在 Mac 端录制系统音频的功能来满足该场景需求，具体接入步骤见下文。

集成说明

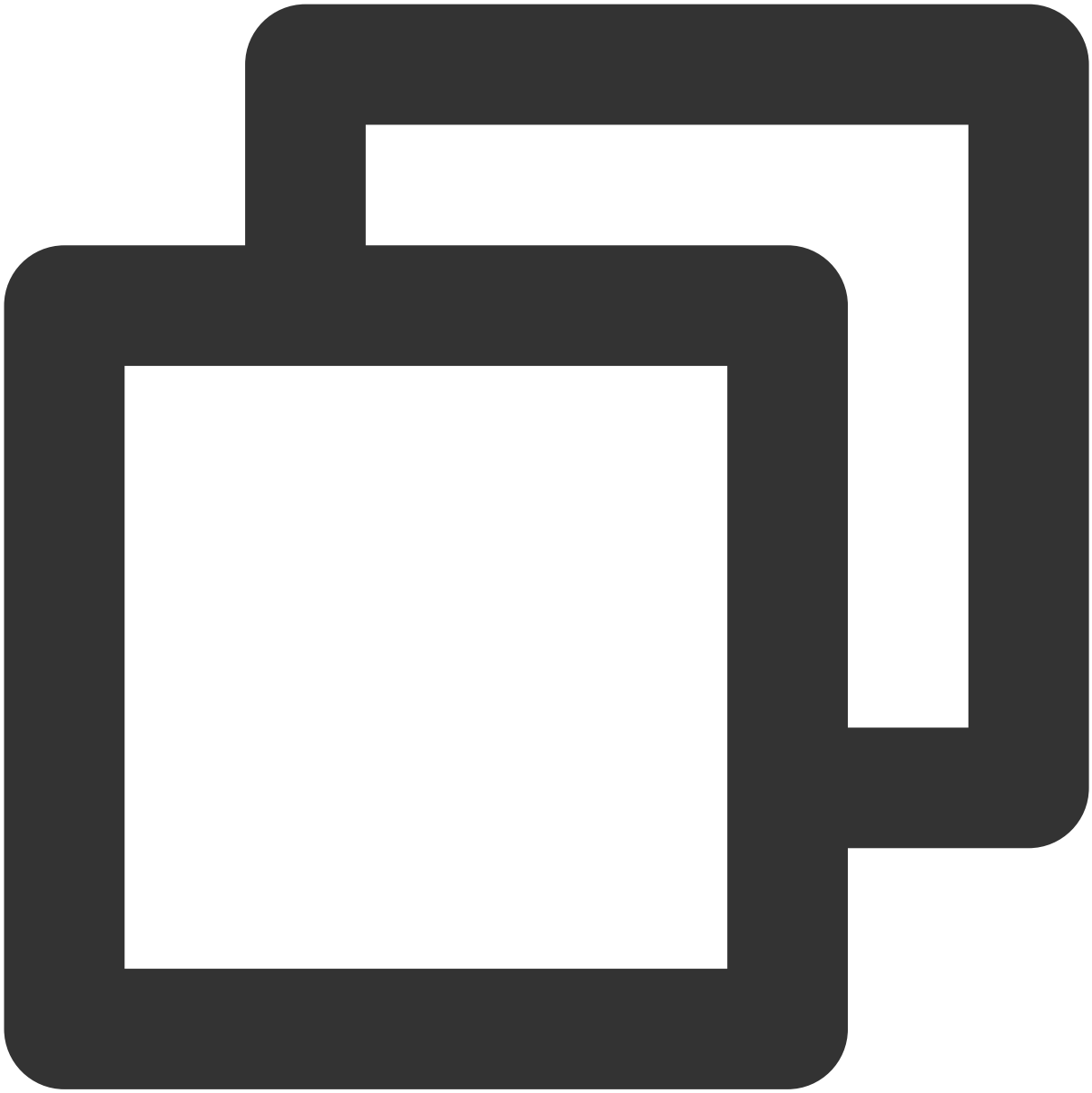
步骤1：集成 TRTCPrivilegedTask 库

SDK 需要使用 TRTCPrivilegedTask 库来获取 root 权限，从而将虚拟声卡插件 TRTCAudioPlugin.driver 安装至系统目录 `/Library/Audio/Plug-Ins/HAL` 。

使用CocoaPods集成

手动集成

1. 打开您当前项目根目录下的 `Podfile` 文件，添加下面的内容：



```
platform :osx, '10.10'

target 'Your Target' do
    pod 'TRTCPrivilegedTask', :podspec => 'https://pod-1252463788.cos.ap-guangzhou.
end
```

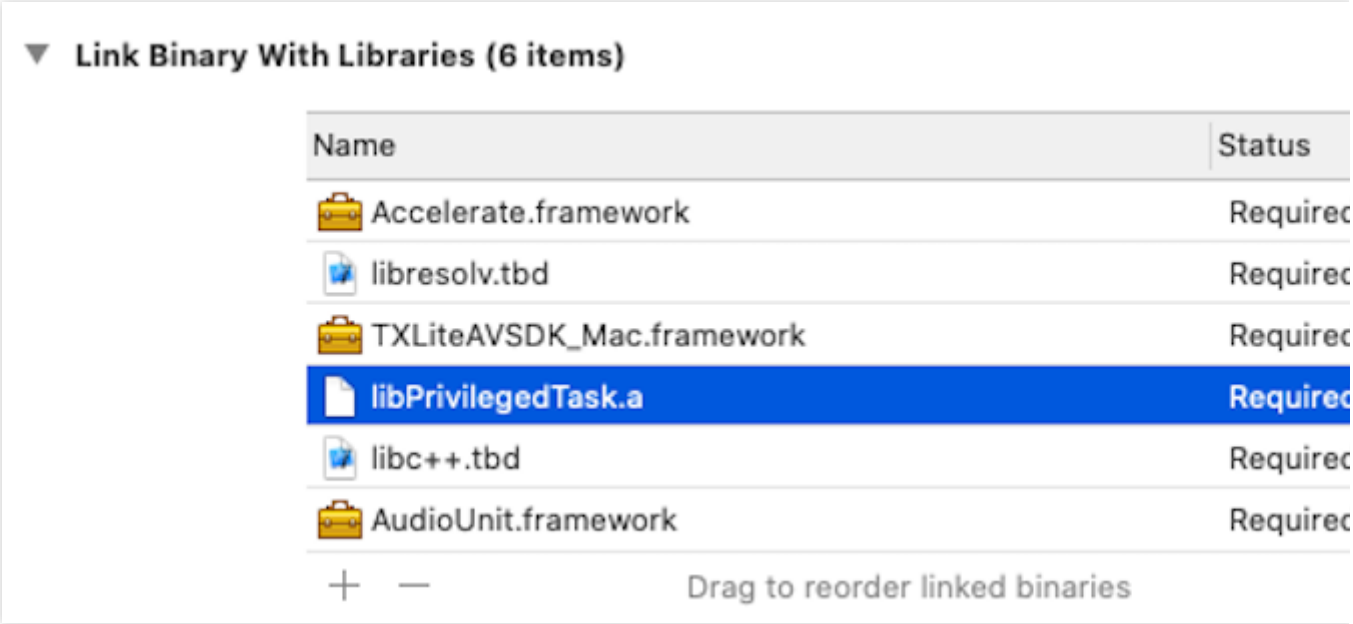
2. 执行 `pod install` 命令安装 **TRTCPrivilegedTask** 库。

说明：

如果项目根目录下没有 Podfile 文件，请先执行 `pod init` 命令新建文件再添加以下内容。

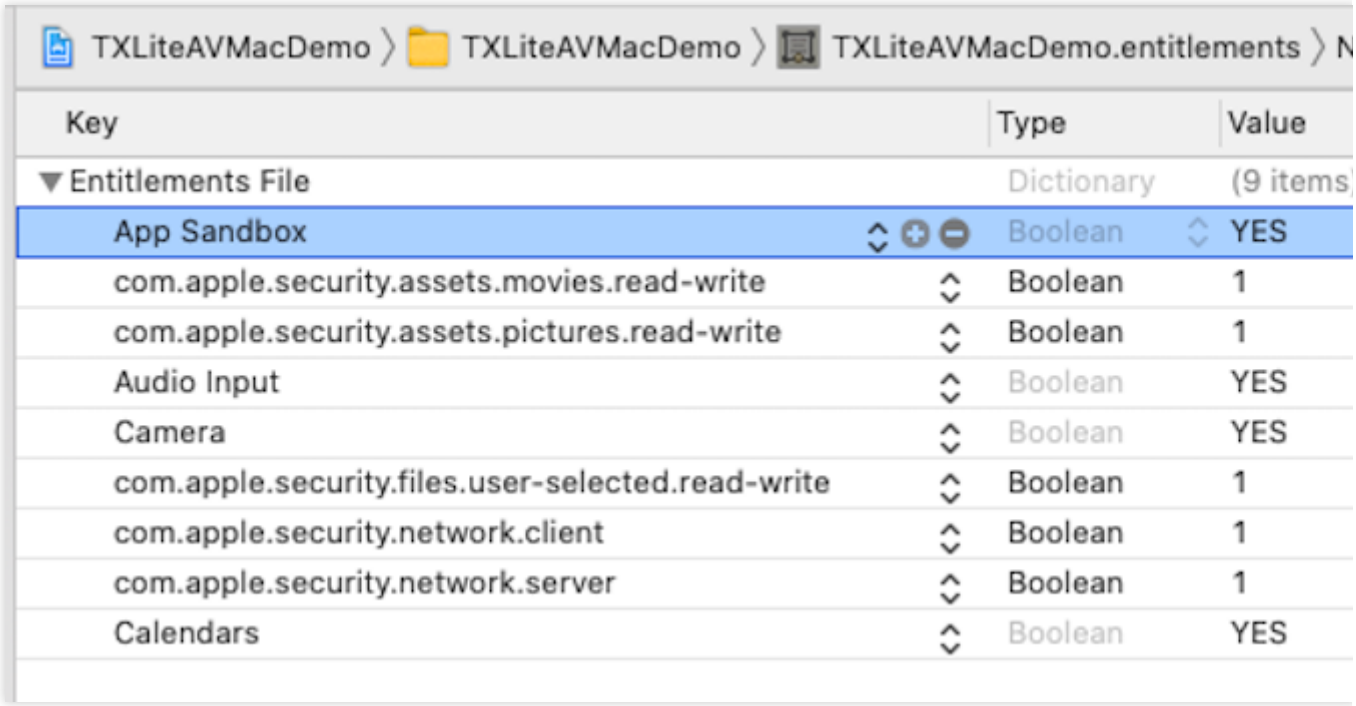
CocoaPods 的安装方法，请参见 CocoaPods 官网安装说明。

1. 下载 TRTCPrivilegedTask 库。
2. 打开您的 Xcode 工程项目，导入解压后的文件 libPrivilegedTask.a 到您的工程。
3. 选择要运行的 target，选中 Build Phases 项，展开Link Binary with Libraries 项，单击底下的**+**，添加依赖库 `libPrivilegedTask.a` 。



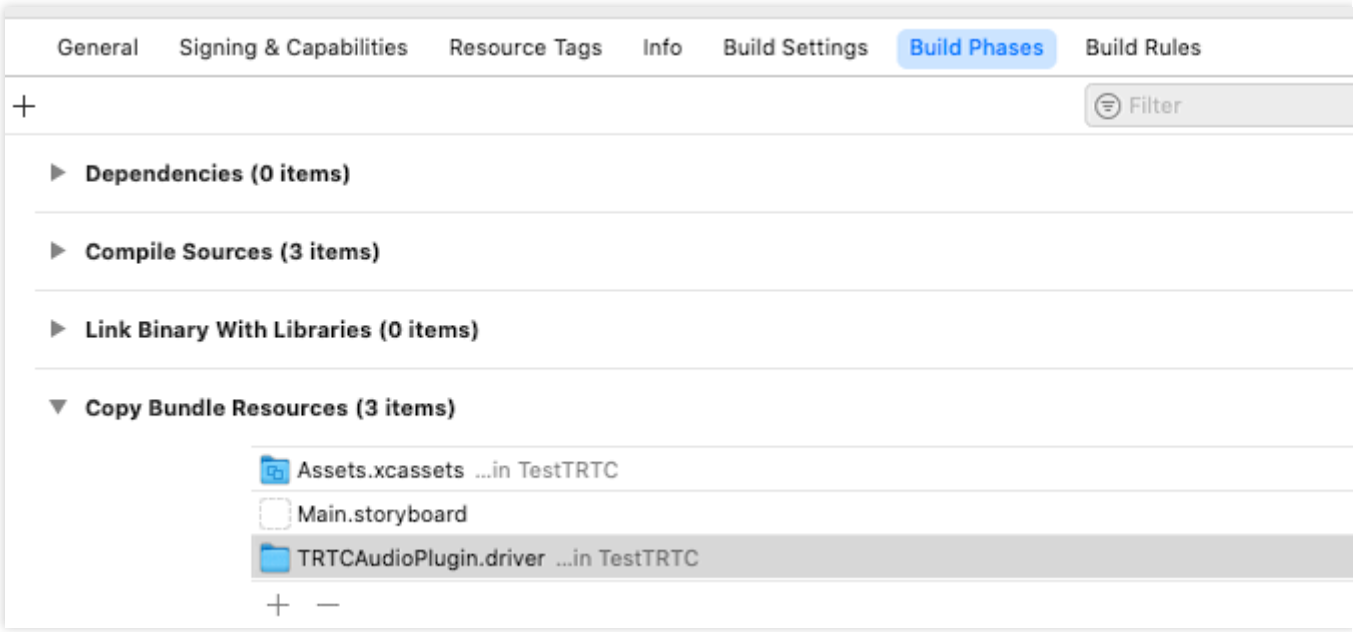
步骤2：取消 App Sandbox 功能

在 App 的 entitlements 描述文件中，删除 **App Sandbox** 条目。

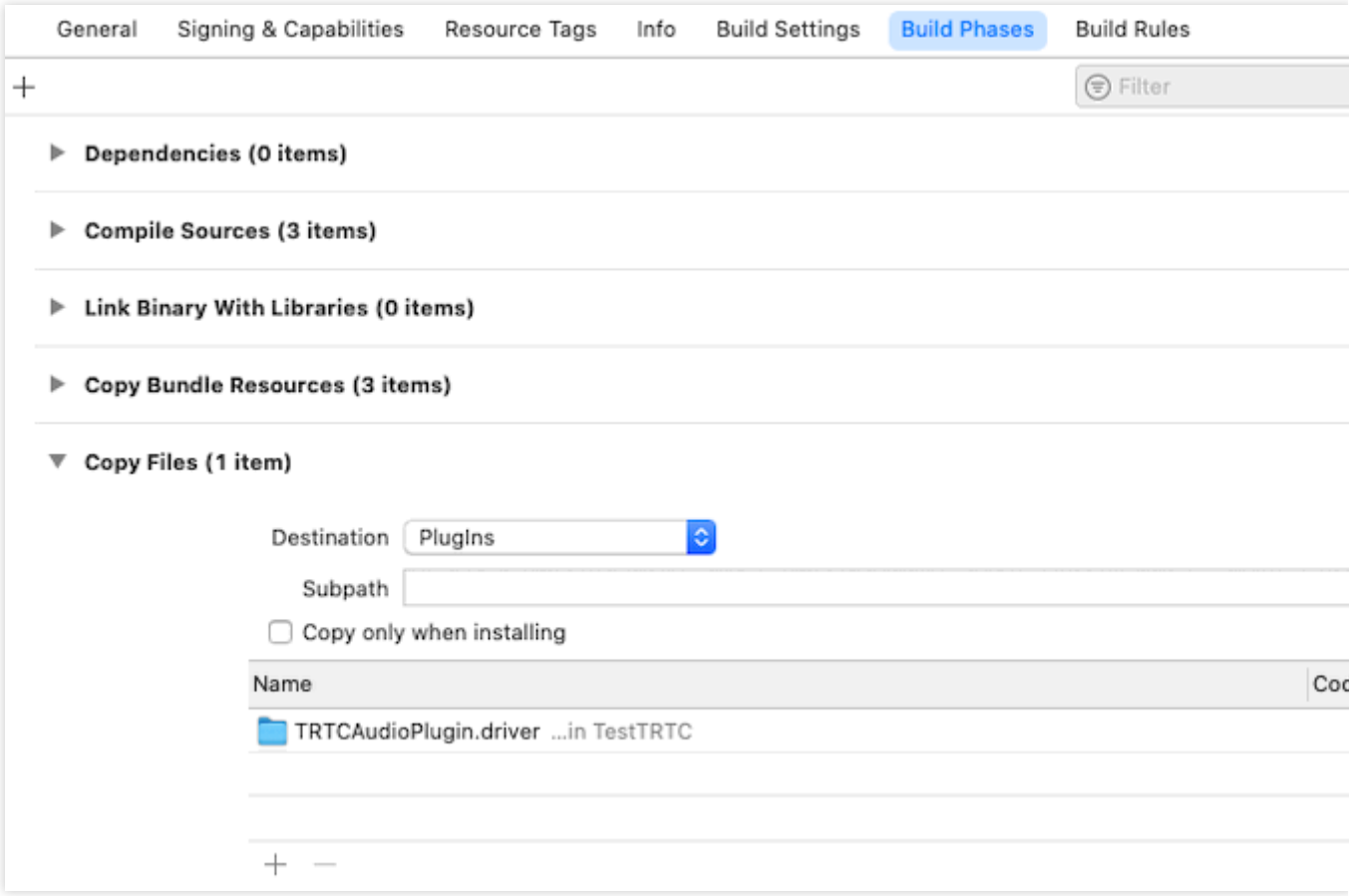


步骤3：虚拟声卡插件打包

在 集成 TRTCPrivilegedTask 库 和 取消 App Sandbox 功能 后，首次使用系统音频录制功能时，SDK 会从网络下载虚拟声卡插件并安装。如果想加速这个过程，可以将放置在 `TXLiteAVSDK_TRTC_Mac.framework` 的 PlugIns 目录下的虚拟声卡插件 `TRTCAudioPlugin.driver` 打包至 App Bundle 的 Resources 目录。如下图：

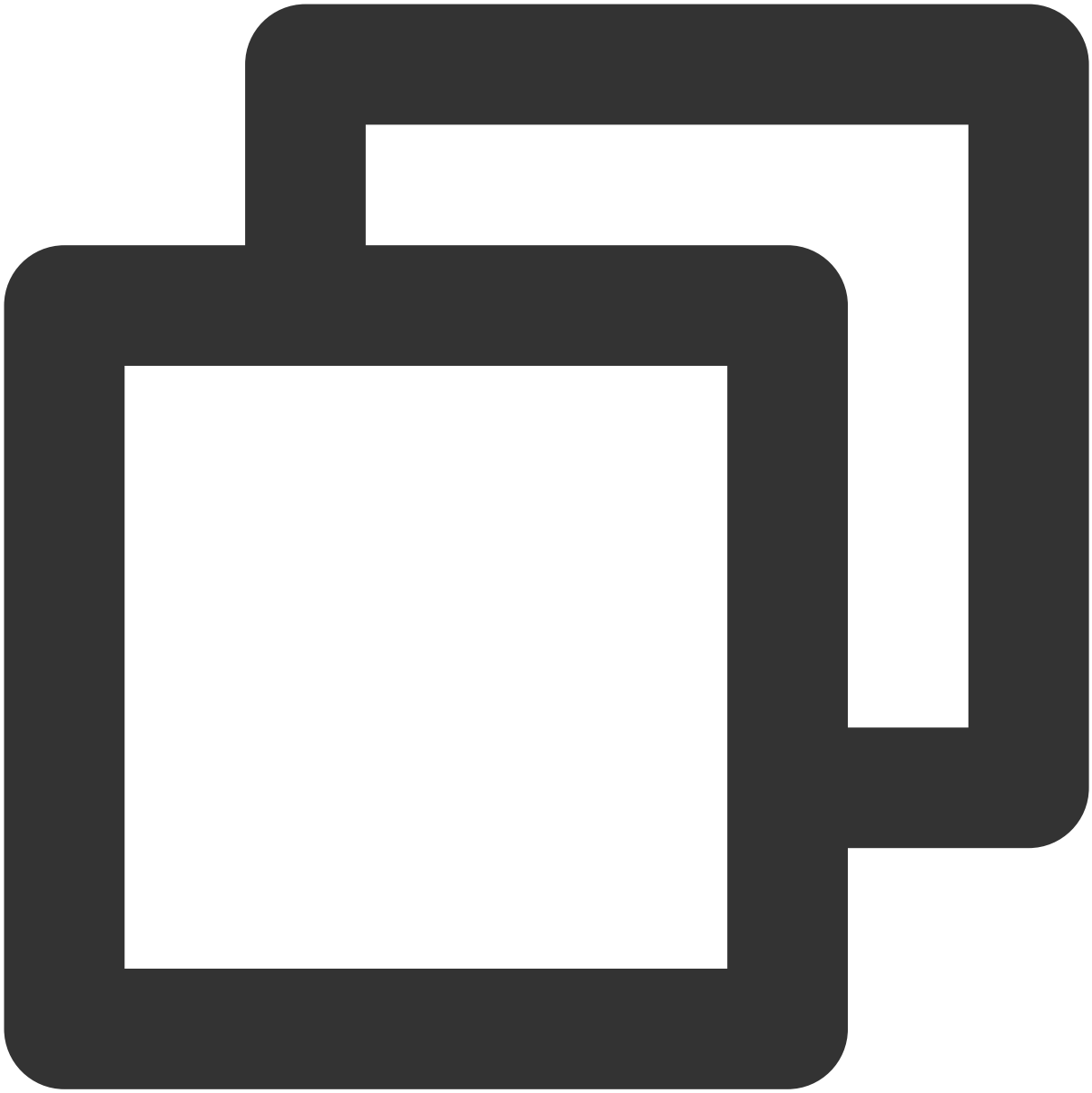


或者拷贝至 App Bundle 的 PlugIns 目录。如下图：



步骤4：开始系统声音采集

调用 startSystemAudioLoopback 接口开始系统声音采集，并将其混入上行音频流中，接口执行完成会通过 onSystemAudioLoopbackError 回调成功或失败的结果。



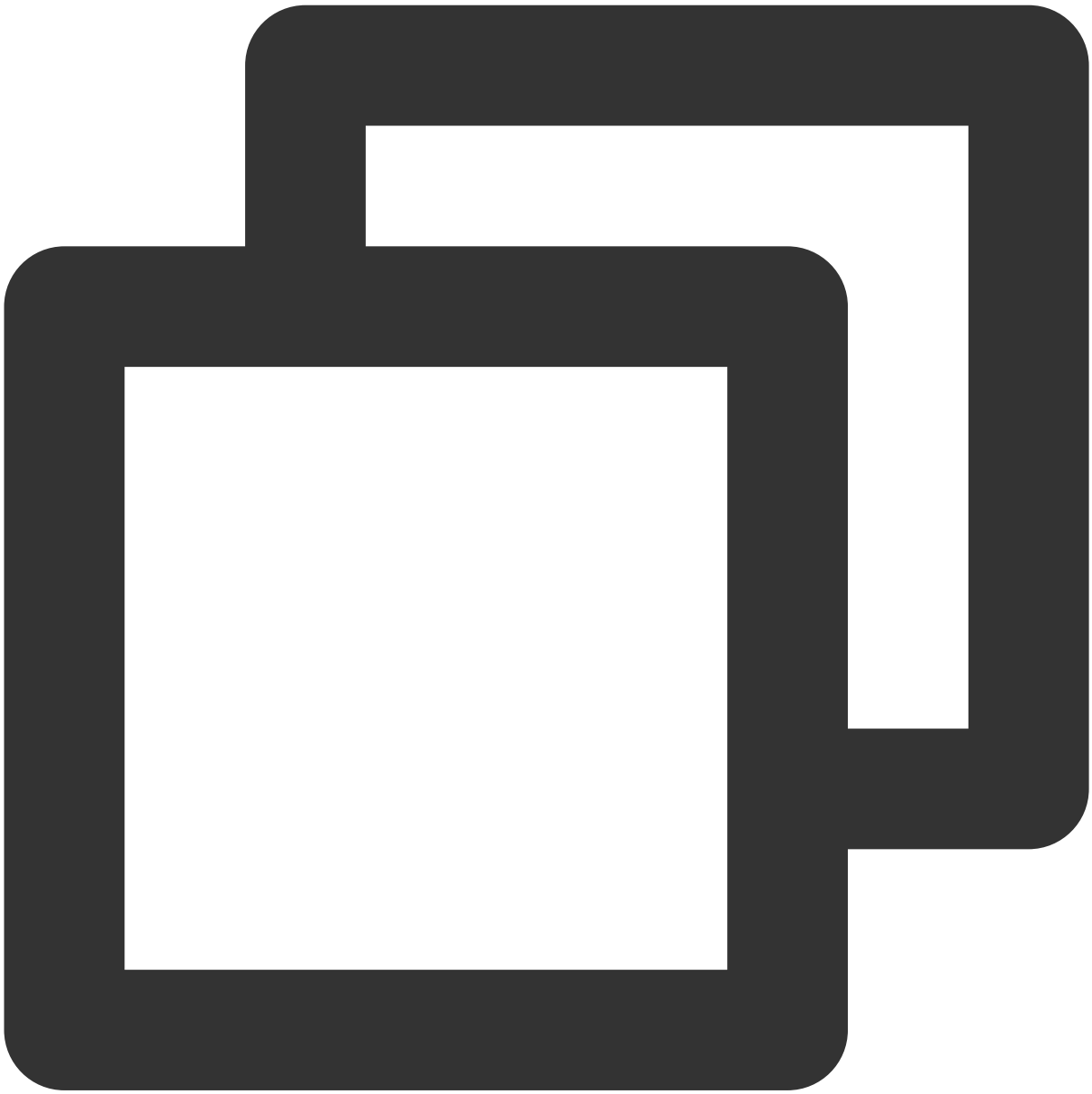
```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud startLocalAudio];
[trtcCloud startSystemAudioLoopback];
```

注意：

集成 TRTCPrivilegedTask 库和取消 App Sandbox 功能后，首次调用 startSystemAudioLoopback 会获取 root 权限。在用户单击**好**后，开始自动安装虚拟声卡插件。

步骤5：停止系统声音采集

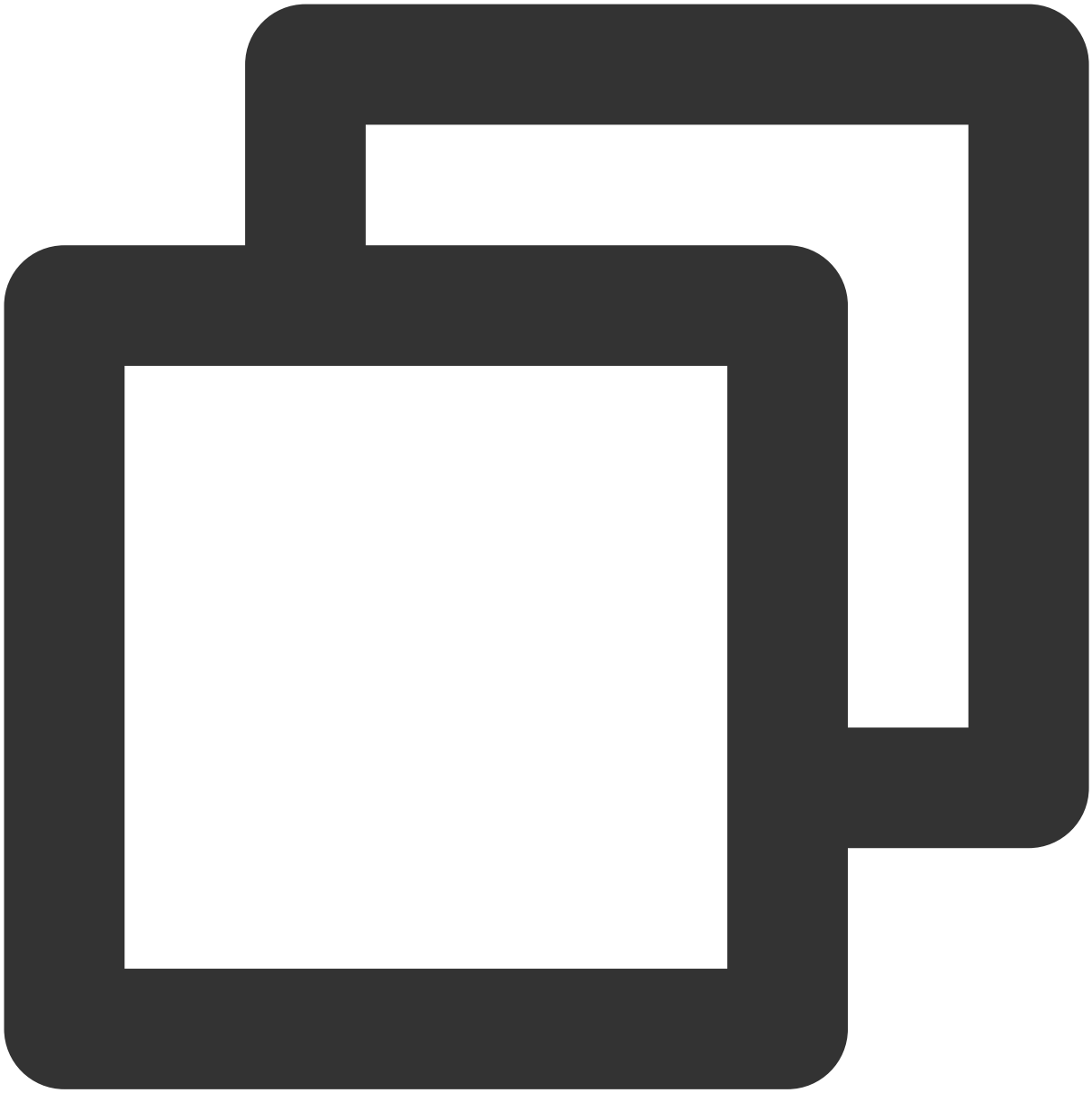
调用 stopSystemAudioLoopback 接口停止系统声音采集。



```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud stopSystemAudioLoopback];
```

步骤6：设置系统声音采集音量

调用 setSystemAudioLoopbackVolume 接口设置系统声音的采集音量。



```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud setSystemAudioLoopbackVolume:80];
```

集成小结

TRTC 在 Mac 端是通过虚拟声卡插件 `TRTCAudioPlugin.driver` 来录制系统音频。这个虚拟声卡插件需要拷贝至系统目录 `/Library/Audio/Plug-Ins/HAL` 并重启音频服务后生效。 可以通过 `启动台` 的 `其他` 文件

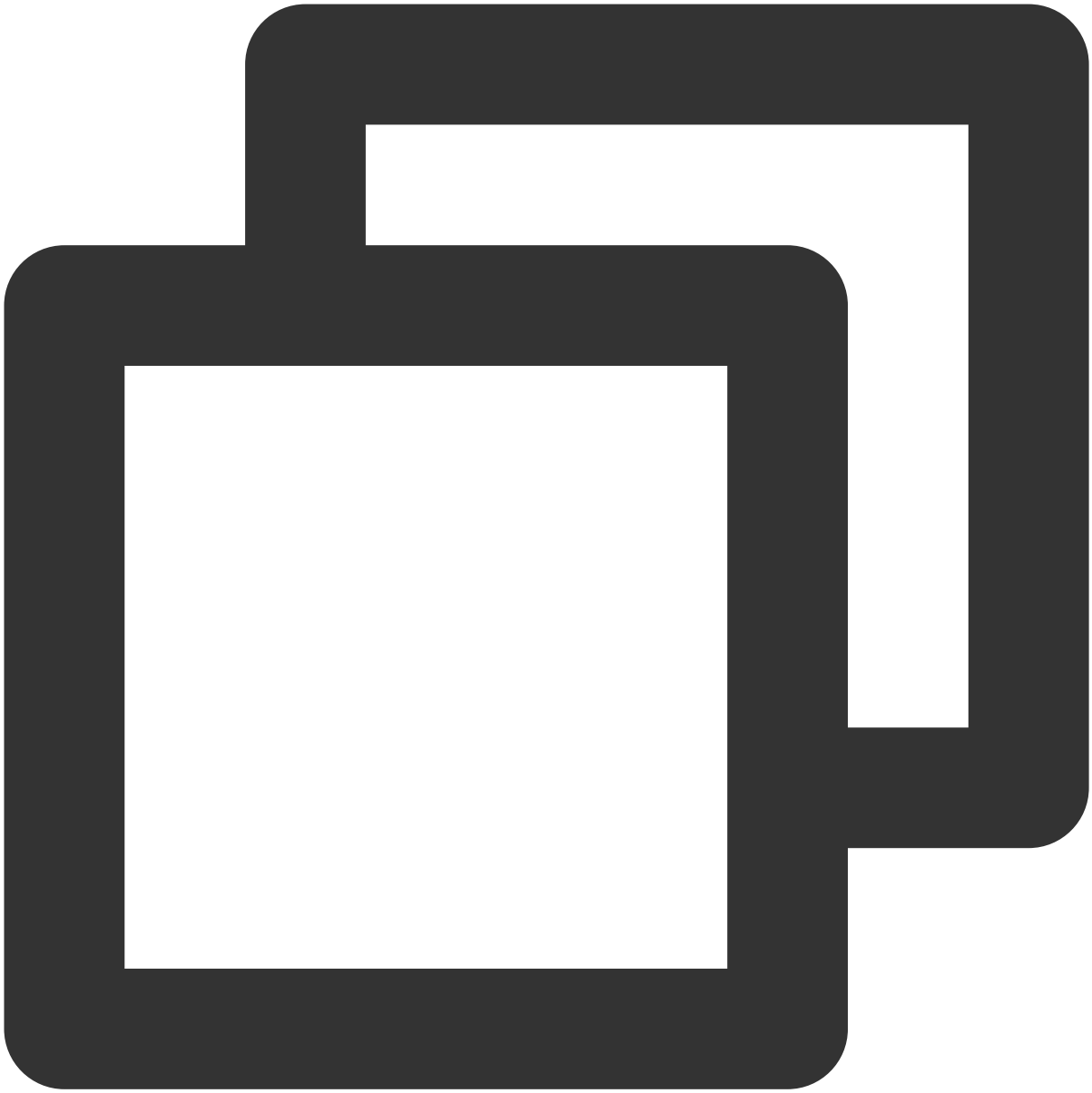
夹中 `音频 MIDI 设置` 应用来检查虚拟声卡插件是否安装成功。在该应用的设备列表中，有名称为 “TRTC Audio Device” 的设备即表明 TRTC 的虚拟声卡插件安装成功。

上述步骤中的 集成 TRTCPrivilegedTask 库 和 取消 App Sandbox 功能 是为 TRTC SDK 自动安装虚拟声卡插件提供 root 权限。如不集成 TRTCPrivilegedTask 库并保留 App Sandbox 功能， SDK 不会自动安装自动安装虚拟声卡插件，但如果系统中已安装好虚拟声卡插件，录制系统音频的功能仍然可以正常使用。

说明：

除上述方案外，也可以手动安装虚拟声卡插件来集成该功能：

1. 将放置在 `TXLiteAVSDK_TRTC_Mac.framework` 的 PlugIns 目录下的 `TRTCAudioPlugin.driver` 拷贝至系统目录 `/Library/Audio/Plug-Ins/HAL` 。
2. 重启系统的音频服务。



```
sudo cp -R TXLiteAVSDK_TRTC_Mac.framework/PlugIns/TRTCAudioPlugin.driver /Library/
sudo kill -9 `ps ax|grep 'coreaudio[a-z]' |awk '{print $1}'`
```

注意事项

App Sandbox 功能取消后，App 内获取到的用户路径会发生变化。通过 NSSearchPathForDirectoriesInDomains 等系统方法获取到的 `~/Documents` 、 `~/Library` 等目录会从沙盒目录切换成用户目录 `/Users/用户`

`名/Documents` 、 `/Users/用户名/Library` 。

集成 TRTCPrivilegedTask 库，可能会使 App 无法上架到 Mac App Store。SDK 自动安装虚拟声卡插件时需要关闭 App Sandbox 功能，并获取 root 权限，App 可能会无法上架到 Mac App Store。详情请参见 App Store Review Guidelines 。如需上架 App Store 或者使用 Sandbox 功能，建议选择手动安装虚拟音频插件的方案。

Electron

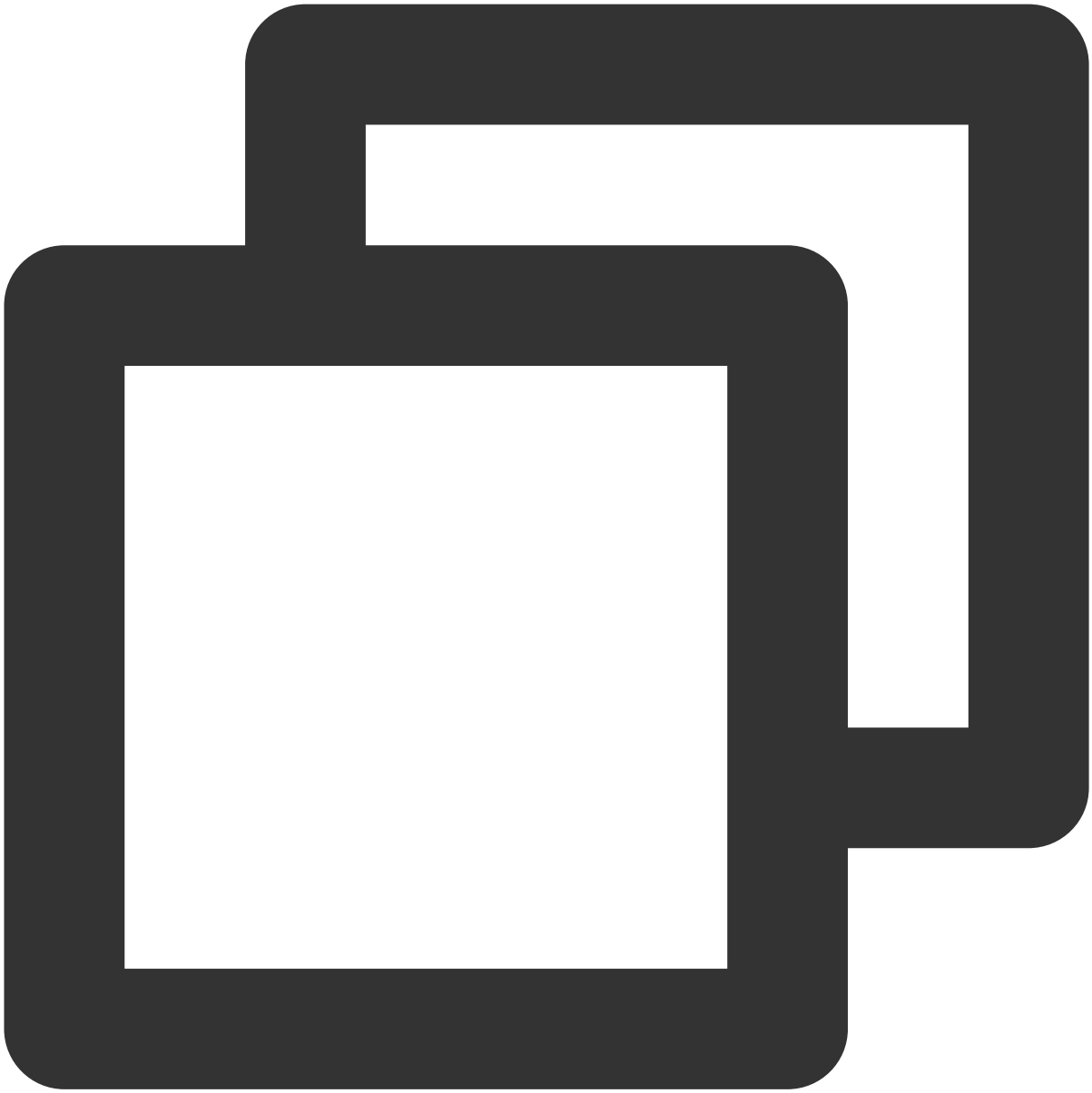
最近更新时间：2024-06-07 11:20:49

场景痛点及解决方案

在屏幕分享等应用场景中，常需要共享系统音频给对方，而采用 Electron 打包 Mac 应用时，Mac 电脑默认声卡不支持采集系统音频，所以在 Mac 电脑上共享系统音频比较困难。基于此，TRTC 提供了在 Mac 端录制系统音频的功能来满足该场景需求，具体接入步骤见下文。

步骤1：开始系统声音采集

调用 startSystemAudioLoopback 接口开始系统声音采集，并将其混入上行音频流中，接口执行完成会通过 onSystemAudioLoopbackError 回调成功或失败的结果。



```
import TRTCCloud, { TRTCAudioQuality } from 'trtc-electron-sdk';
const rtcCloud = new TRTCCloud();
function onSystemAudioLoopbackError(errCode) {
  if (errCode === 0) {
    console.log('启动成功');
  }
  if (errCode === -1330) {
    console.log('开启系统声音录制失败，例如音频驱动插件不可用');
  }
  if (errCode === -1331) {
    console.log('未授权安装音频驱动插件');
```

```
    }
    if (errCode === -1332) {
      console.log('安装音频驱动插件失败');
    }
}

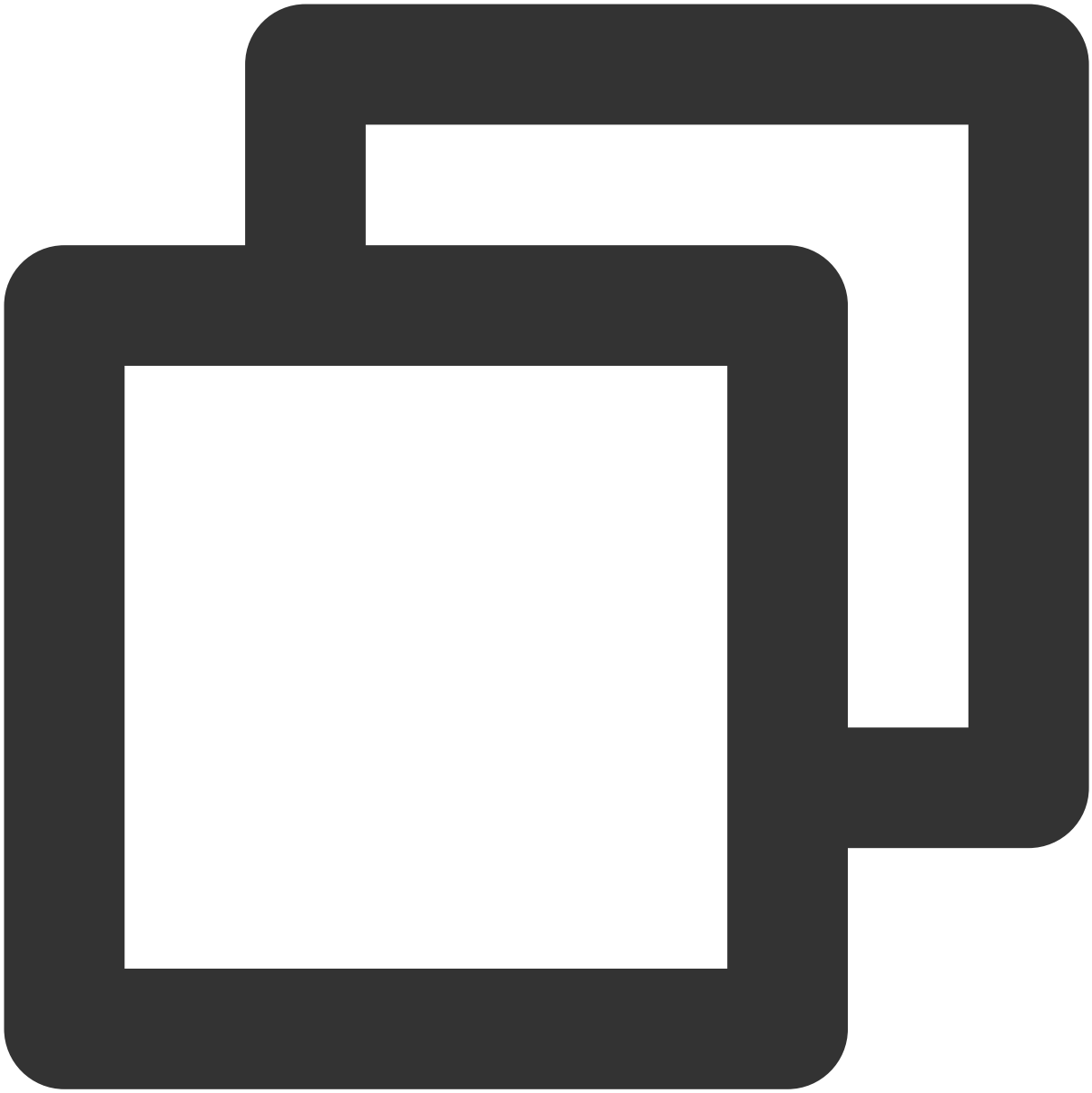
trtcCloud.on('onSystemAudioLoopbackError', onSystemAudioLoopbackError);
trtcCloud.startLocalAudio(TRTCAudioQuality.TRTCAudioQualityDefault);
trtcCloud.startSystemAudioLoopback();
```

注意：

首次调用 startSystemAudioLoopback 会获取 root 权限（如下图），在用户单击**好**后，开始自动安装虚拟声卡插件。

步骤2：停止系统声音采集

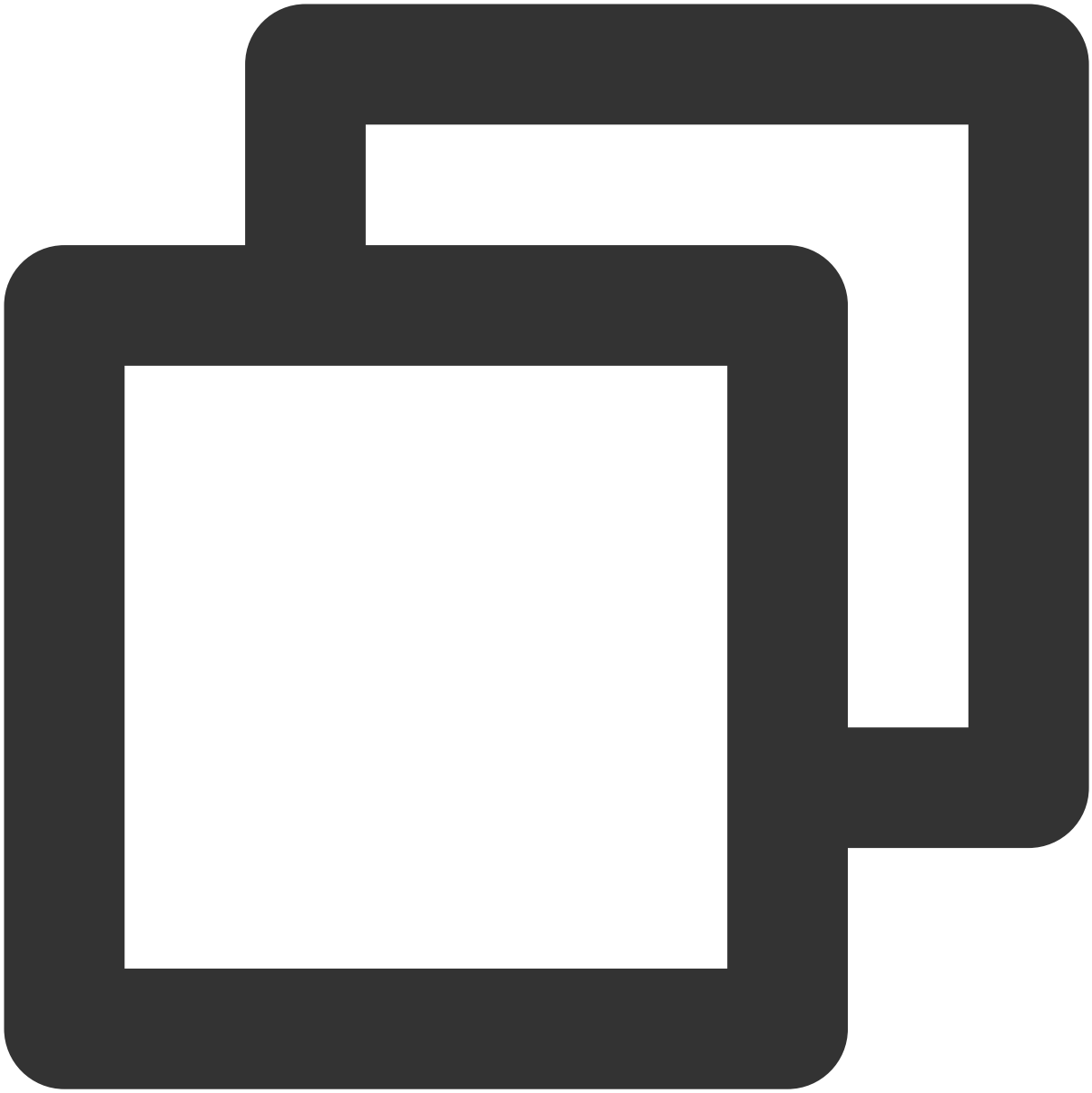
调用 stopSystemAudioLoopback 接口停止系统声音采集。



```
trtcCloud.stopSystemAudioLoopback();
```

步骤3：设置系统声音采集音量

调用 setSystemAudioLoopbackVolume 接口设置系统声音的采集音量。



```
trtcCloud.setSystemAudioLoopbackVolume(60);
```

集成小结

TRTC 在 Mac 端是通过虚拟声卡插件 `TRTCAudioPlugin.driver` 来录制系统音频。这个虚拟声卡插件需要拷贝至系统目录 `/Library/Audio/Plug-Ins/HAL` 并重启音频服务后生效。 可以通过 `启动台` 的 `其他` 文件

夹中 `音频 MIDI 设置` 应用来检查虚拟声卡插件是否安装成功。在该应用的设备列表中，有名称为 “TRTC Audio Device” 的设备即表明 TRTC 的虚拟声卡插件安装成功。

Flutter

最近更新时间：2024-02-02 18:48:25

本文档主要介绍如何分享系统声音，目前 TRTC 默认不采集所在应用的音频。

调用指引

Android
iOS
本文档主要介绍如何分享系统声音，目前 TRTC 默认不采集所在应用的音频。
注意：
仅 Android 10.0 及以上版本支持分享系统声音。

开启分享系统声音

步骤1: 开启屏幕分享

按照开启屏幕分享 - 基于Android平台中的步骤开启屏幕分享。

步骤2: 开启分享系统声音

调用 `TRTCCloud` 中的 startSystemAudioLoopback 接口即可，这时候采集的系统声音会自动混入上行流中。

步骤3: 关闭分享系统声音

调用 `TRTCCloud` 中的 stopSystemAudioLoopback 接口即可。

步骤1: 开启麦克风

在 App 中，调用 startLocalAudio 开启麦克风采集，推荐使用 TRTCAudioQualityDefault 音质。
注意：
这一步是必要的，通过开启麦克风采集，可以让 App 在后台时依然保持运行。

步骤2: 开启屏幕分享

由于 iOS 限制，只有在屏幕录制时才能采集系统声音。因此，为实现此功能，需要先对接iOS屏幕分享功能。
按照 开启屏幕分享 - 基于iOS平台 中的步骤，开启屏幕录制，系统声音将自动进行采集。
注意：
开启屏幕录制时，不要点亮麦克风图标，人声采集已在 App 中启动。

版权所有：腾讯云计算（北京）有限责任公司



步骤3: 中途开启和关闭系统声音

系统声音采集与屏幕录制同时进行，只能随录制开启而自动启动，随录制关闭而一起停止。无法单独对系统声音进行开关或静音操作。

TRTCCloud 提供了 setSystemAudioLoopbackVolume 方法对系统声音进行音量调节。当您不希望将系统声音输出时，可以将音量设置为0。

设定画面质量
Android&iOS&Windows&Mac

最近更新时间：2022-09-06 10:41:17

内容介绍

在 TRTCCloud 中，您可以通过以下方式调整画质：

- TRTCCloud.enterRoom 中的 TRTCAppScene 参数：用于选择您的应用场景。
- TRTCCloud.setVideoEncoderParam：用于设置编码参数。
- TRTCCloud.setNetworkQosParam：用于设置网络调控策略。

本文主要介绍如何配置上述参数，使 TRTC SDK 的画质效果符合您的项目需要。
您也可以参考以下 Demo：

- iOS：SetVideoQualityViewController.m
- Android：SetVideoQualityActivity.java
- Windows：TRTCMainViewController.cpp

支持的平台

iOS	Android	Mac OS	Windows	Web	Electron	Flutter
✓	✓	✓	✓	✓	✓	✓

Web端设定画面质量的详细操作，请参见 设定指引。

TRTCAppScene

- **VideoCall**
  对应视频通话场景，即绝大多数时间都是两人或两人以上视频通话的场景，内部编码器和网络协议优化侧重流畅性，降低通话延迟和卡顿率。

- **LIVE**
  对应直播场景，即绝大多数时间都是一人直播，偶尔有多人视频互动的场景，内部编码器和网络协议优化侧重性

能和兼容性，性能和清晰度表现更佳。

TRTCVideoEncParam

推荐的配置

应用场景	videoResolution	videoFps	videoBitrate
视频通话（手机）	640x360	15	550kbps
视频会议（主画面 @ Mac Win）	1280x720	15	1200kbps
视频会议（主画面 @ 手机）	640x360	15	900kbps
视频会议（小画面）	320x180	15	250kbps
在线教育（老师 @ Mac Win）	960x540	15	850kbps
在线教育（老师 @ iPad）	640x360	15	550kbps
在线教育（学生）	320x180	15	250kbps

各字段详解

- **(TRTCVideoResolution) videoResolution**
  编码分辨率，例如 640 x 360 是指编码出的画面的宽（像素） x 高（像素），我们在 TRTCVideoResolution 枚举定义里只定义了宽 >= 高的横屏（Landscape）分辨率，如果想要使用竖屏分辨率，需要将 resMode 设置为 Portrait。

  > 注意：
  > 由于很多硬件编解码器只支持能被 16 整除的像素宽度，所以 SDK 实际编码出的分辨率并不一定完全按照参数自定，而是会自动进行 16 整除修正。例如 640 x 360 的分辨率，在 SDK 内部有可能会适配为 640 x 368。

- **(TRTCVideoResolutionMode) resMode**
  指横屏或竖屏分辨率，由于 TRTCVideoResolution 中只定义了横屏分辨率，如果您希望使用 360 x 640 这样的竖屏分辨率，就需要指定 resMode 为 TRTCVideoResolutionModePortrait。一般 PC 和 Mac 采用横屏（Landscape）分辨率，手机采用竖屏（Portrait）分辨率。

版权所有：腾讯云计算（北京）有限责任公司

- **(int) videoFps**
  帧率（FPS），也就是每秒钟要编码多少帧画面。推荐设置为 15 FPS，这样既能保证画面足够流畅，又不会因为每秒帧数太多而拉低单幅画面的清晰度。

  如果您对流畅度要求比较高，可以设置为 20 FPS 或 25 FPS。但请不要设置 25 FPS 以上的数值，因为电影的常规帧率也只有 24 FPS。

- **(int) videoBitrate**
  视频码率（Bitrate），即每秒钟编码器输出多少 Kbit 的编码后的二进制数据。如果您将 videoBitrate 设置为 800kbps，那么每秒钟编码器会产生 800kbit 的视频数据，这些数据如果存储成一个文件，那么文件大小就是 800kbit，也就是100KB，也就是 0.1M。

  视频码率并不是越高越好，它跟分辨率之间要有比较恰当的映射关系，如下表所示。

分辨率码率参照表

分辨率定义	宽高比	建议码率（VideoCall）	建议码率（LIVE）
TRTCVideoResolution_120_120	1:1	80kbps	120kbps
TRTCVideoResolution_160_160	1:1	100kbps	150kbps
TRTCVideoResolution_270_270	1:1	200kbps	300kbps
TRTCVideoResolution_480_480	1:1	350kbps	525kbps
TRTCVideoResolution_160_120	4:3	100kbps	150kbps
TRTCVideoResolution_240_180	4:3	150kbps	225kbps
TRTCVideoResolution_280_210	4:3	200kbps	300kbps
TRTCVideoResolution_320_240	4:3	250kbps	375kbps
TRTCVideoResolution_400_300	4:3	300kbps	450kbps
TRTCVideoResolution_480_360	4:3	400kbps	600kbps
TRTCVideoResolution_640_480	4:3	600kbps	900kbps
TRTCVideoResolution_960_720	4:3	1000kbps	1500kbps
TRTCVideoResolution_160_90	16:9	150kbps	250kbps
TRTCVideoResolution_256_144	16:9	200kbps	300kbps

版权所有：腾讯云计算（北京）有限责任公司

分辨率定义	宽高比	建议码率（VideoCall）	建议码率（LIVE）
TRTCVideoResolution_320_180	16:9	250kbps	400kbps
TRTCVideoResolution_480_270	16:9	350kbps	550kbps
TRTCVideoResolution_640_360	16:9	550kbps	900kbps
TRTCVideoResolution_960_540	16:9	850kbps	1300kbps
TRTCVideoResolution_1280_720	16:9	1200kbps	1800kbps
TRTCVideoResolution_1920_1080	16:9	2000kbps	3000kbps

TRTCNetworkQosParam

QosPreference

在网络带宽比较充裕的情况下，清晰和流畅是可以兼顾的，但当用户的网络并不理想时，究竟是优先保证清晰还是优先保证流畅？您可以通过指定 TRTCNetworkQosParam 中的 preference 参数来做出选择。

- **流畅优先（TRTCVideoQosPreferenceSmooth）**
  在用户遭遇弱网环境时，画面会变得模糊，且会有较多马赛克，但可以保持流畅或轻微卡顿。

- **清晰优先（TRTCVideoQosPreferenceClear）**
  在用户遭遇弱网环境时，画面会尽可能保持清晰，但可能会更容易出现卡顿。

ControlMode

controlMode 参数选择 **TRTCQosControlModeServer** 即可，TRTCQosControlModeClient 是腾讯云研发团队做内部调试用的，请勿关注。

常见的误区

1. 分辨率越高越好？

较高的分辨率也需要较高的码率来支撑，如果分辨率选择 1280 x 720，但码率却指定为 200kbps，画面就会有大量的马赛克。推荐参考 分辨率码率参照表 进行设置。

2. 帧率越高越好？

由于摄像头采集的画面是曝光阶段中所有现实物体的完整映射，所以并不是帧率越高，感官就越流畅，这一点跟游

戏里的FPS是不一样的。恰恰相反，帧率过高，会拉低每帧画面的画质，也会减少摄像机的曝光时间，效果可能会更差。

3. 码率越高越好？

较高的码率也需要较高的分辨率来匹配，对于 320 x 240 这样分辨率，1000kbps 的码率就很浪费了，推荐参考 分辨率码率参照表 进行设置。

4. 用 Wi-Fi 的时候就可以设置很高的分辨率和码率

并不是说 Wi-Fi 的网速是恒定不变的，如果离无线路由器较远， 或者路由器信道被占用，可能网速还不如 4G。
针对这种情况， TRTC SDK 提供了测速功能，可以在视频通话前先进行测速，根据打分值来确定网络好坏。

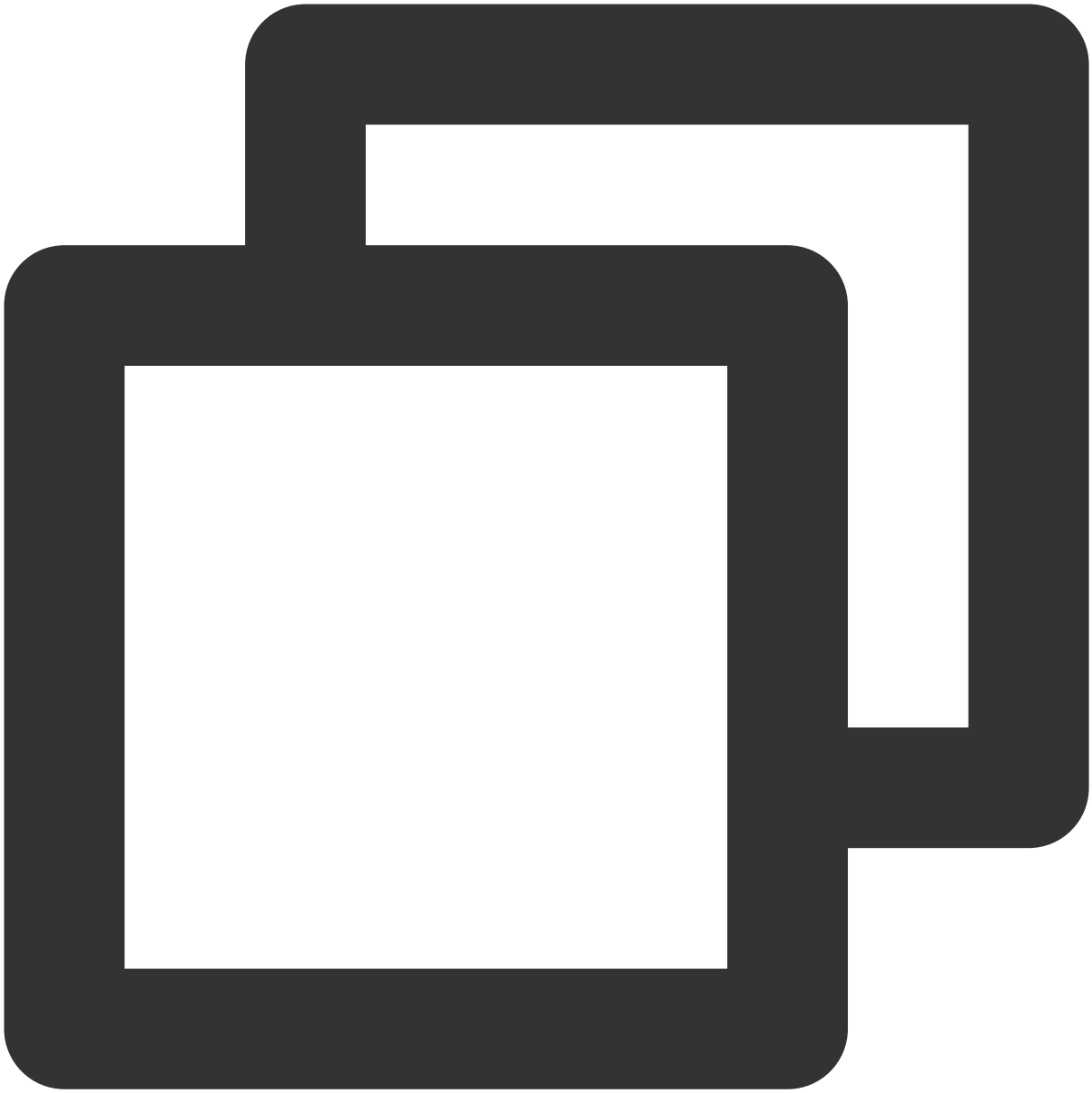
Web

最近更新时间：2023-05-30 17:14:32

本文主要介绍如何在视频通话或互动直播中设置视频属性，开发者可以根据具体业务需求调整视频画面的清晰度和流畅度，获得更好的用户体验。
视频属性包括分辨率、帧率和码率。

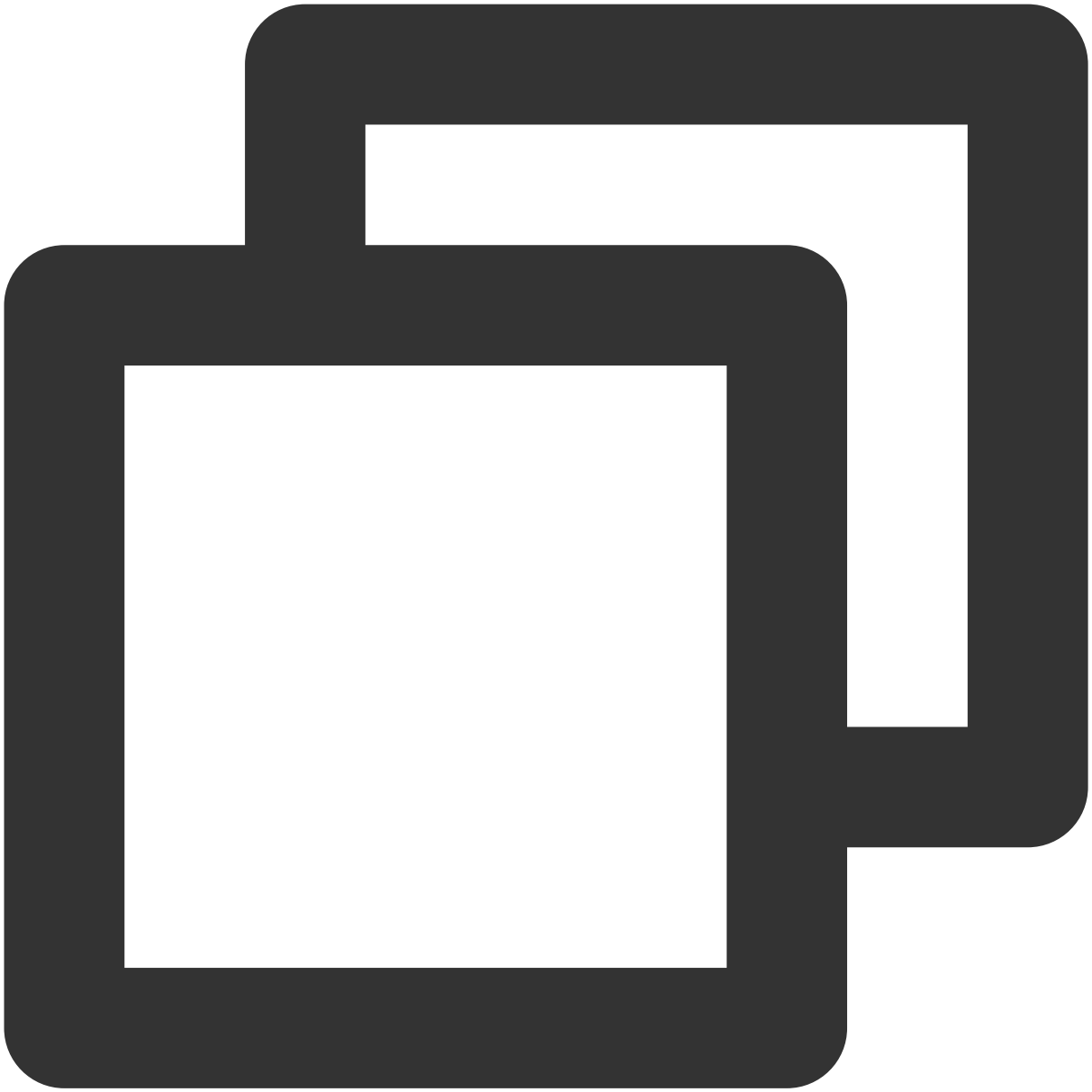
实现方式

通过 trtc 对象的 `TRTC.startLocalVideo()` 或 `TRTC.updateLocalVideo()` 方法设置视频属性：
指定一个预定义的 Profile，每个 Profile 对应着一套推荐的分辨率、帧率和码率。



```
// 启动时指定视频属性
await trtc.startLocalVideo({
option: { profile: '480p' }
});
// 通话过程中动态调整视频属性
await trtc.updateLocalVideo({
option: { profile: '360p' }
});
```

指定自定义分辨率、帧率和码率



```
// 启动时指定视频属性
await trtc.startLocalVideo({
option: { profile: { width: 640, height: 480, frameRate: 15, bitrate: 900 /* kpbs *
});
// 通话过程中动态调整视频属性
await trtc.updateLocalVideo({
option: { profile: { width: 640, height: 360, frameRate: 15, bitrate: 800 /* kpbs *
});
```

版权所有：腾讯云计算（北京）有限责任公司

视频属性 Profile 列表

视频 Profile	分辨率（宽 x 高）	帧率（fps）	码率（kbps）
120p	160 x 120	15	200
180p	320 x 180	15	350
240p	320 x 240	15	400
360p	640 x 360	15	800
480p	640 x 480	15	900
720p	1280 x 720	15	1500
1080p	1920 x 1080	15	2000
1440p	2560 x 1440	30	4860
4K	3840 x 2160	30	9000

由于设备和浏览器的限制，视频分辨率不一定能够完全匹配，在这种情况下，浏览器会自动调整分辨率使其接近 Profile 对应的分辨率

版权所有：腾讯云计算（北京）有限责任公司

Electron

最近更新时间：2022-07-21 17:07:40

本文主要介绍如何在视频通话或互动直播中设置画面质量，开发者可以根据具体业务需求调整视频画面的清晰度和流畅度，获得更好的用户体验。
视频属性包括分辨率、帧率和码率。

内容介绍

在 Electron SDK 中，您可以通过以下方式调整画质：

- enterRoom 中的 TRTCAppScene 参数：用于选择您的应用场景。
- setVideoEncoderParam ：用于设置编码参数。
- setNetworkQosParam ：用于设置网络调控策略。

本文主要介绍如何配置上述参数，使 TRTC SDK 的画质效果符合您的项目需要。
您也可以参考 Electron API Example：video-quality Demo。

TRTCAppScene

- **VideoCall**：对应视频通话场景，即绝大多数时间都是两人或两人以上视频通话的场景，内部编码器和网络协议优化侧重流畅性，降低通话延迟和卡顿率。
- **LIVE**：对应直播场景，即绝大多数时间都是一人直播，偶尔有多人视频互动的场景，内部编码器和网络协议优化侧重性能和兼容性，性能和清晰度表现更佳。

TRTCVideoEncParam

推荐的配置

应用场景	videoResolution	videoFps	videoBitrate
视频会议（主画面 @ Mac Win）	1280x720	15	1200kbps
在线教育（老师 @ Mac Win）	960x540	15	850kbps

各字段详解

- **(TRTCVideoResolution) videoResolution**
  编码分辨率，例如 640 x 360 是指编码出的画面的宽（像素） x 高（像素），我们在 TRTCVideoResolution 枚举定义里只定义了宽 >= 高的横屏（Landscape）分辨率，如果想要使用竖屏分辨率，需要将 resMode 设置为 Portrait。

  > 注意：
  > 由于很多硬件编解码器只支持能被 16 整除的像素宽度，所以 SDK 实际编码出的分辨率并不一定完全按照参数自定，而是会自动进行 16 整除修正。例如 640 x 360 的分辨率，在 SDK 内部有可能会适配为 640 x 368。

- **(TRTCVideoResolutionMode) resMode**
  指横屏或竖屏分辨率，由于 TRTCVideoResolution 中只定义了横屏分辨率，如果您希望使用 360 x 640 这样的竖屏分辨率，就需要指定 resMode 为 TRTCVideoResolutionModePortrait。一般 PC 和 Mac 采用横屏（Landscape）分辨率，手机采用竖屏（Portrait）分辨率。

- **(int) videoFps**
  帧率（FPS），也就是每秒钟要编码多少帧画面。推荐设置为 15 FPS，这样既能保证画面足够流畅，又不会因为每秒帧数太多而拉低单幅画面的清晰度。

  如果您对流畅度要求比较高，可以设置为 20 FPS 或 25 FPS。但请不要设置 25 FPS 以上的数值，因为电影的常规帧率也只有 24 FPS。

- **(int) videoBitrate**
  视频码率（Bitrate），即每秒钟编码器输出多少 Kbit 的编码后的二进制数据。如果您将 videoBitrate 设置为 800kbps，那么每秒钟编码器会产生 800kbit 的视频数据，这些数据如果存储成一个文件，那么文件大小就是 800kbit，也就是100KB，也就是 0.1M。

  视频码率并不是越高越好，它跟分辨率之间要有比较恰当的映射关系，如下表所示。

分辨率码率参照表

分辨率定义	宽高比	建议码率（VideoCall）	建议码率（LIVE）
TRTCVideoResolution_120_120	1:1	80kbps	120kbps
TRTCVideoResolution_160_160	1:1	100kbps	150kbps
TRTCVideoResolution_270_270	1:1	200kbps	300kbps

版权所有：腾讯云计算（北京）有限责任公司

分辨率定义	宽高比	建议码率（VideoCall）	建议码率（LIVE）
TRTCVideoResolution_480_480	1:1	350kbps	525kbps
TRTCVideoResolution_160_120	4:3	100kbps	150kbps
TRTCVideoResolution_240_180	4:3	150kbps	225kbps
TRTCVideoResolution_280_210	4:3	200kbps	300kbps
TRTCVideoResolution_320_240	4:3	250kbps	375kbps
TRTCVideoResolution_400_300	4:3	300kbps	450kbps
TRTCVideoResolution_480_360	4:3	400kbps	600kbps
TRTCVideoResolution_640_480	4:3	600kbps	900kbps
TRTCVideoResolution_960_720	4:3	1000kbps	1500kbps
TRTCVideoResolution_160_90	16:9	150kbps	250kbps
TRTCVideoResolution_256_144	16:9	200kbps	300kbps
TRTCVideoResolution_320_180	16:9	250kbps	400kbps
TRTCVideoResolution_480_270	16:9	350kbps	550kbps
TRTCVideoResolution_640_360	16:9	550kbps	900kbps
TRTCVideoResolution_960_540	16:9	850kbps	1300kbps
TRTCVideoResolution_1280_720	16:9	1200kbps	1800kbps
TRTCVideoResolution_1920_1080	16:9	2000kbps	3000kbps

TRTCNetworkQosParam

QosPreference

在网络带宽比较充裕的情况下，清晰和流畅是可以兼顾的，但当用户的网络并不理想时，究竟是优先保证清晰还是优先保证流畅？您可以通过指定 TRTCNetworkQosParam 中的 preference 参数来做出选择。

- **流畅优先（TRTCVideoQosPreferenceSmooth）**
  在用户遭遇弱网环境时，画面会变得模糊，且会有较多马赛克，但可以保持流畅或轻微卡顿。

- **清晰优先（TRTCVideoQosPreferenceClear）**
  在用户遭遇弱网环境时，画面会尽可能保持清晰，但可能会更容易出现卡顿。

ControlMode

controlMode 参数选择 **TRTCQosControlModeServer** 即可，TRTCQosControlModeClient 是腾讯云研发团队做内部调试用的，请勿关注。

常见的误区

1. 分辨率越高越好？

较高的分辨率也需要较高的码率来支撑，如果分辨率选择 1280 x 720，但码率却指定为 200kbps，画面就会有大量的马赛克。推荐参考 分辨率码率参照表 进行设置。

2. 帧率越高越好？

由于摄像头采集的画面是曝光阶段中所有现实物体的完整映射，所以并不是帧率越高，感官就越流畅，这一点跟游戏里的FPS是不一样的。恰恰相反，帧率过高，会拉低每帧画面的画质，也会减少摄像机的曝光时间，效果可能会更差。

3. 码率越高越好？

较高的码率也需要较高的分辨率来匹配，对于 320 x 240 这样分辨率，1000kbps 的码率就很浪费了，推荐参考 分辨率码率参照表 进行设置。

4. 用 Wi-Fi 的时候就可以设置很高的分辨率和码率

并不是说 Wi-Fi 的网速是恒定不变的，如果离无线路由器较远， 或者路由器信道被占用，可能网速还不如 4G。
针对这种情况， TRTC SDK 提供了测速功能，可以在视频通话前先进行测速，根据打分值来确定网络好坏。

Flutter

最近更新时间：2024-02-02 18:49:46

内容介绍

在 TRTCCloud 中，您可以通过以下方式调整画质：
TRTCCloud.enterRoom 中的 TRTCAppScene 参数：用于选择您的应用场景。
TRTCCloud.setVideoEncoderParam：用于设置编码参数。
TRTCCloud.setNetworkQosParam：用于设置网络调控策略。
本文主要介绍如何配置上述参数，使 TRTC SDK 的画质效果符合您的项目需要。
您也可以参考以下 Demo：
Flutter：SetVideoQualityPage.dart

支持的平台

iOS	Android	Mac OS	Windows	Web	Electron	Flutter
✓	✓	✓	✓	✓	✓	✓

Web端设定画面质量的详细操作，请参见 设定指引。

房间场景

场景类型	场景介绍
TRTC_APP_SCENE_VIDEOCALL	在视频通话场景中，支持 720p 和 1080p 高清图像质量。单个房间可容纳最多300名同时在线用户，最多50人同时发言。
TRTC_APP_SCENE_LIVE	在互动视频直播场景中，麦克风可以顺畅地打开/关闭，无需等待切换，主播延迟低至300毫秒以下。支持数十万并发观众用户的直播，播放延迟降至1000毫秒。 **注意**：在此场景下，您必须使用 TRTCParams 中的 role 字段指定当前用户的角色。
TRTC_APP_SCENE_AUDIOCALL	在音频通话场景中，支持48 kHz双声道音频通话。单个房间可容纳最多300名同时在线用户，最多50人同时发言。

TRTC_APP_SCENE_VOICE_CHATROOM	在互动音频直播场景中，麦克风可以顺畅地打开/关闭，无需等待切换，主播延迟低至300毫秒以下。支持数十万并发观众用户的直播，播放延迟降至1000毫秒。 **注意**：在此场景下，您必须使用 TRTCParams 中的 role 字段指定当前用户的角色。

TRTCVideoEncParam

推荐的配置

应用场景	videoResolution	videoFps	videoBitrate
视频通话（手机）	640x360	15	550kbps
视频会议（主画面 @ Mac Win）	1280x720	15	1200kbps
视频会议（主画面 @ 手机）	640x360	15	900kbps
视频会议（小画面）	320x180	15	250kbps
在线教育（老师 @ Mac Win）	960x540	15	850kbps
在线教育（老师 @ iPad）	640x360	15	550kbps
在线教育（学生）	320x180	15	250kbps

各字段详解

(int) videoResolution

编码分辨率（TRTCCloudDef.TRTC_VIDEO_RESOLUTION_），例如 640 x 360 是指编码出的画面的宽（像素） x 高（像素），我们在 TRTCVideoResolution 枚举定义里只定义了宽 >= 高的横屏（Landscape）分辨率，如果想要使用竖屏分辨率，需要将 resMode 设置为 Portrait。

注意：

由于很多硬件编解码器只支持能被 16 整除的像素宽度，所以 SDK 实际编码出的分辨率并不一定完全按照参数自定，而是会自动进行 16 整除修正。例如 640 x 360 的分辨率，在 SDK 内部有可能会适配为 640 x 368。

(int) videoResolutionMode

指横屏或竖屏分辨率（TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_），由于 TRTCVideoResolution 中只定义了横屏分辨率，如果您希望使用 360 x 640 这样的竖屏分辨率，就需要指定 resMode 为 TRTCVideoResolutionModePortrait。一般 PC 和 Mac 采用横屏（Landscape）分辨率，手机采用竖屏（Portrait）分辨率。

(int) videoFps

帧率（FPS），也就是每秒钟要编码多少帧画面。推荐设置为 15 FPS，这样既能保证画面足够流畅，又不会因为每

秒帧数太多而拉低单幅画面的清晰度。如果您对流畅度要求比较高，可以设置为 20 FPS 或 25 FPS。但请不要设置 25 FPS 以上的数值，因为电影的常规帧率也只有 24 FPS。

(int) videoBitrate

视频码率（Bitrate），即每秒钟编码器输出多少 Kbit 的编码后的二进制数据。如果您将 videoBitrate 设置为 800kbps，那么每秒钟编码器会产生 800kbit 的视频数据，这些数据如果存储成一个文件，那么文件大小就是 800kbit，也就是100KB，也就是 0.1M。

视频码率并不是越高越好，它跟分辨率之间要有比较恰当的映射关系，如下表所示。

分辨率码率参照表

分辨率定义	宽高比	建议码率（VideoCall）	建议码率（LIVE）
TRTCVideoResolution_120_120	1:1	80kbps	120kbps
TRTCVideoResolution_160_160	1:1	100kbps	150kbps
TRTCVideoResolution_270_270	1:1	200kbps	300kbps
TRTCVideoResolution_480_480	1:1	350kbps	525kbps
TRTCVideoResolution_160_120	4:3	100kbps	150kbps
TRTCVideoResolution_240_180	4:3	150kbps	225kbps
TRTCVideoResolution_280_210	4:3	200kbps	300kbps
TRTCVideoResolution_320_240	4:3	250kbps	375kbps
TRTCVideoResolution_400_300	4:3	300kbps	450kbps
TRTCVideoResolution_480_360	4:3	400kbps	600kbps
TRTCVideoResolution_640_480	4:3	600kbps	900kbps
TRTCVideoResolution_960_720	4:3	1000kbps	1500kbps
TRTCVideoResolution_160_90	16:9	150kbps	250kbps
TRTCVideoResolution_256_144	16:9	200kbps	300kbps
TRTCVideoResolution_320_180	16:9	250kbps	400kbps
TRTCVideoResolution_480_270	16:9	350kbps	550kbps
TRTCVideoResolution_640_360	16:9	550kbps	900kbps
TRTCVideoResolution_960_540	16:9	850kbps	1300kbps

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoResolution_1280_720	16:9	1200kbps	1800kbps
TRTCVideoResolution_1920_1080	16:9	2000kbps	3000kbps

TRTCNetworkQosParam

QosPreference

在网络带宽比较充裕的情况下，清晰和流畅是可以兼顾的，但当用户的网络并不理想时，究竟是优先保证清晰还是优先保证流畅？您可以通过指定 TRTCNetworkQosParam 中的 preference 参数来做出选择。

流畅优先（TRTCVideoQosPreferenceSmooth）

在用户遭遇弱网环境时，画面会变得模糊，且会有较多马赛克，但可以保持流畅或轻微卡顿。

清晰优先（TRTCVideoQosPreferenceClear）

在用户遭遇弱网环境时，画面会尽可能保持清晰，但可能会更容易出现卡顿。

ControlMode

controlMode 参数选择 **TRTCQosControlModeServer** 即可，TRTCQosControlModeClient 是腾讯云研发团队做内部调试用的，请勿关注。

常见的误区

1. 分辨率越高越好？

较高的分辨率也需要较高的码率来支撑，如果分辨率选择 1280 x 720，但码率却指定为 200kbps，画面就会有大量的马赛克。推荐参考 分辨率码率参照表 进行设置。

2. 帧率越高越好？

由于摄像头采集的画面是曝光阶段中所有现实物体的完整映射，所以并不是帧率越高，感官就越流畅，这一点跟游戏里的FPS是不一样的。恰恰相反，帧率过高，会拉低每帧画面的画质，也会减少摄像机的曝光时间，效果可能会更差。

3. 码率越高越好？

较高的码率也需要较高的分辨率来匹配，对于 320 x 240 这样分辨率，1000kbps 的码率就很浪费了，推荐参考 分辨率码率参照表 进行设置。

4. 用 Wi-Fi 的时候就可以设置很高的分辨率和码率

并不是说 Wi-Fi 的网速是恒定不变的，如果离无线路由器较远， 或者路由器信道被占用，可能网速还不如 4G。

针对这种情况， TRTC SDK 提供了测速功能，可以在视频通话前先进行测速，根据打分值来确定网络好坏。

视频画面旋转和播放 Android&iOS&Windows&Mac

最近更新时间：2023-10-09 17:12:42

内容介绍

跟手机直播千篇一律的竖屏体验不同，实时音视频（TRTC）需要兼顾横屏和竖屏两种场景，因此就会有很多横竖屏的处理逻辑需要去应对，本文主要介绍：

如何实现竖屏模式，例如：微信的视频通话就是一个典型的竖屏体验模式。

如何实现横屏模式，例如：多人音视频房间 App（类似小鱼易连）往往都是采用横屏模式。

如何自定义控制本地画面和远程画面的旋转方向和填充模式。



平台支持

iOS	Android	Mac OS	Windows	Electron	Web 端
✓	✓	✓	✓	✓	×

竖屏模式

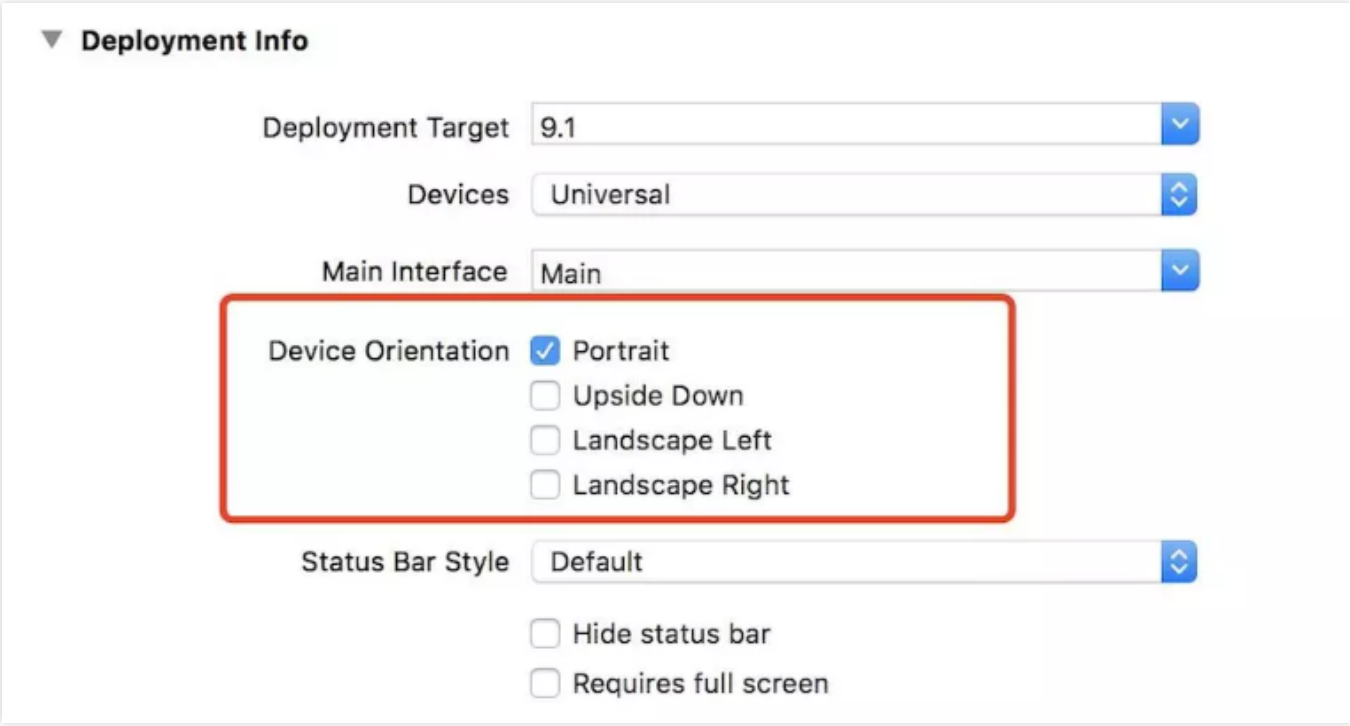
如果要实现类似微信视频通话的体验模式，需要做两项工作：

1. 配置 App 的 UI 界面为竖屏

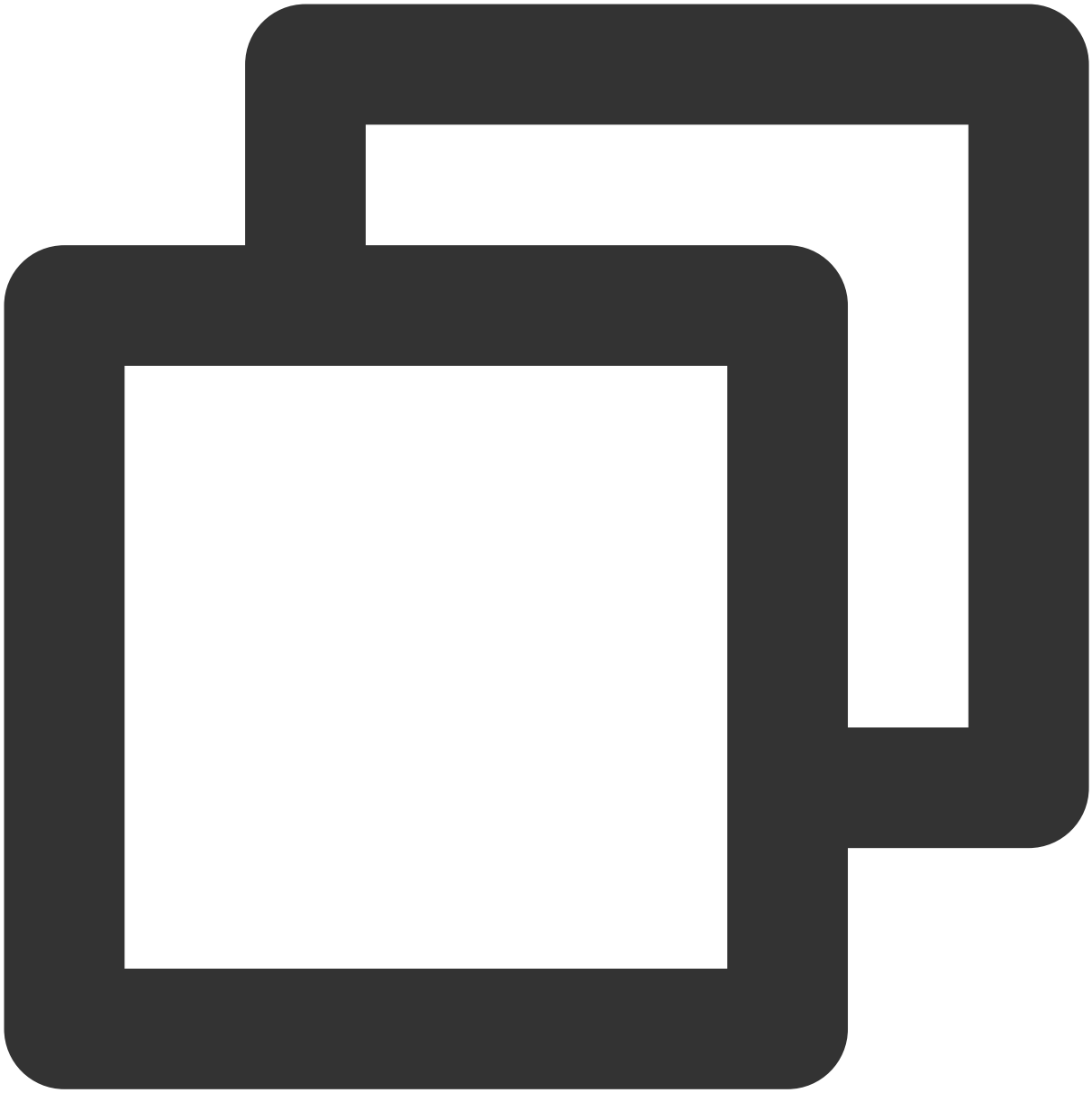
iOS 平台

Android 平台

可以直接在 XCode 的 **General** > **Deployment Info** > **Device Orientation** 中进行设置：



您也可以通过实现 Appdelegate 中的 `supportedInterfaceOrientationsForWindow` 方法来达到相同目标：



```
- (UIInterfaceOrientationMask)application:(UIApplication *)application
    supportedInterfaceOrientationsForWindow:(UIWindow *)window
{

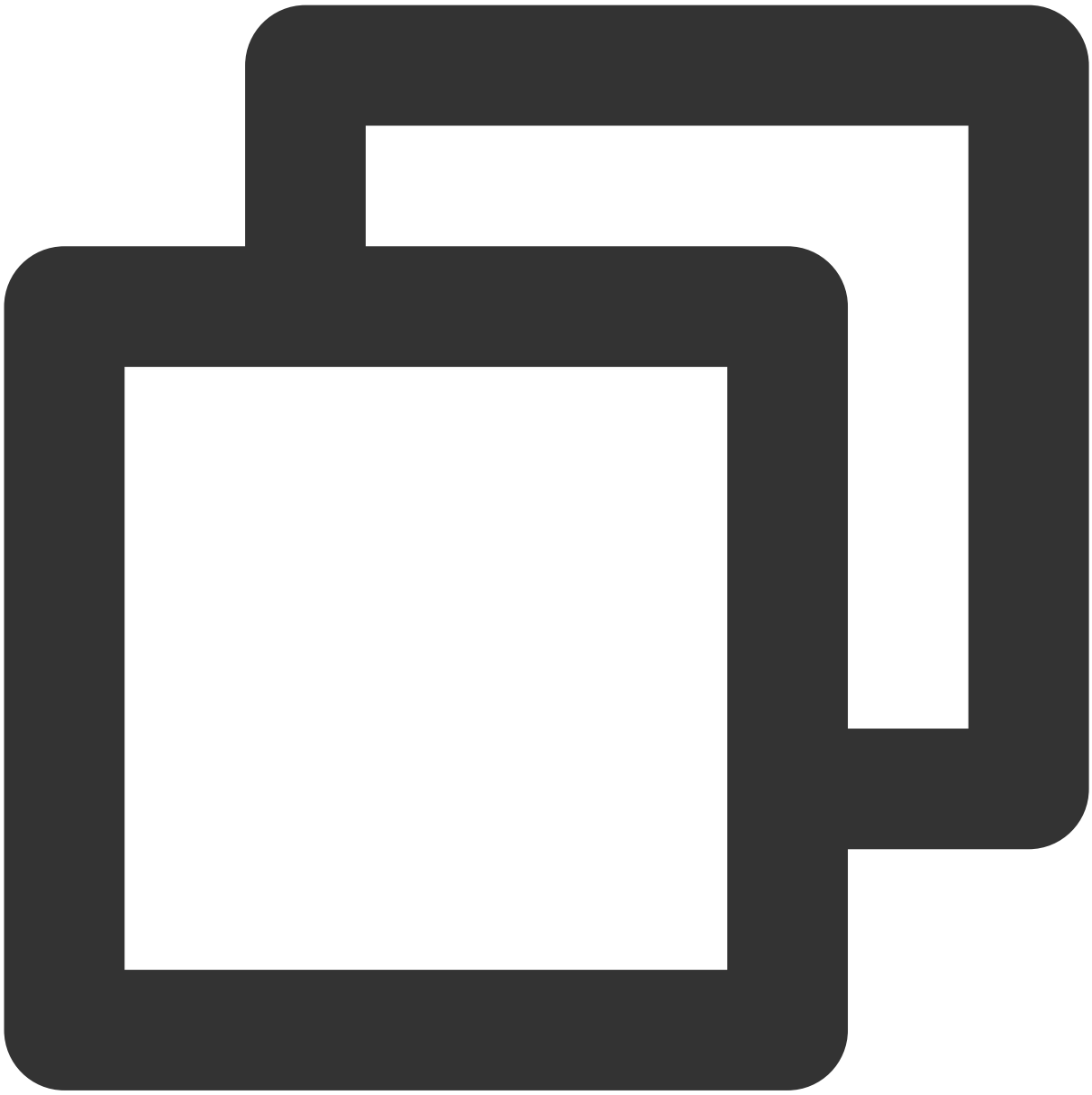
    return  UIInterfaceOrientationMaskPortrait ;

}
```

说明：

CSDN 上有一篇文章 iOS横竖屏旋转及其基本适配方法，详细介绍了 iOS 平台中关于屏幕方向的一些开发经验。

通过指定 activity 的 `screenOrientation` 属性为 portrait，即可指定该界面为竖屏模式：



```
<activity android:name=".trtc.TRTCMainActivity"  android:launchMode="singleTask" an
          android:screenOrientation="portrait" />
```

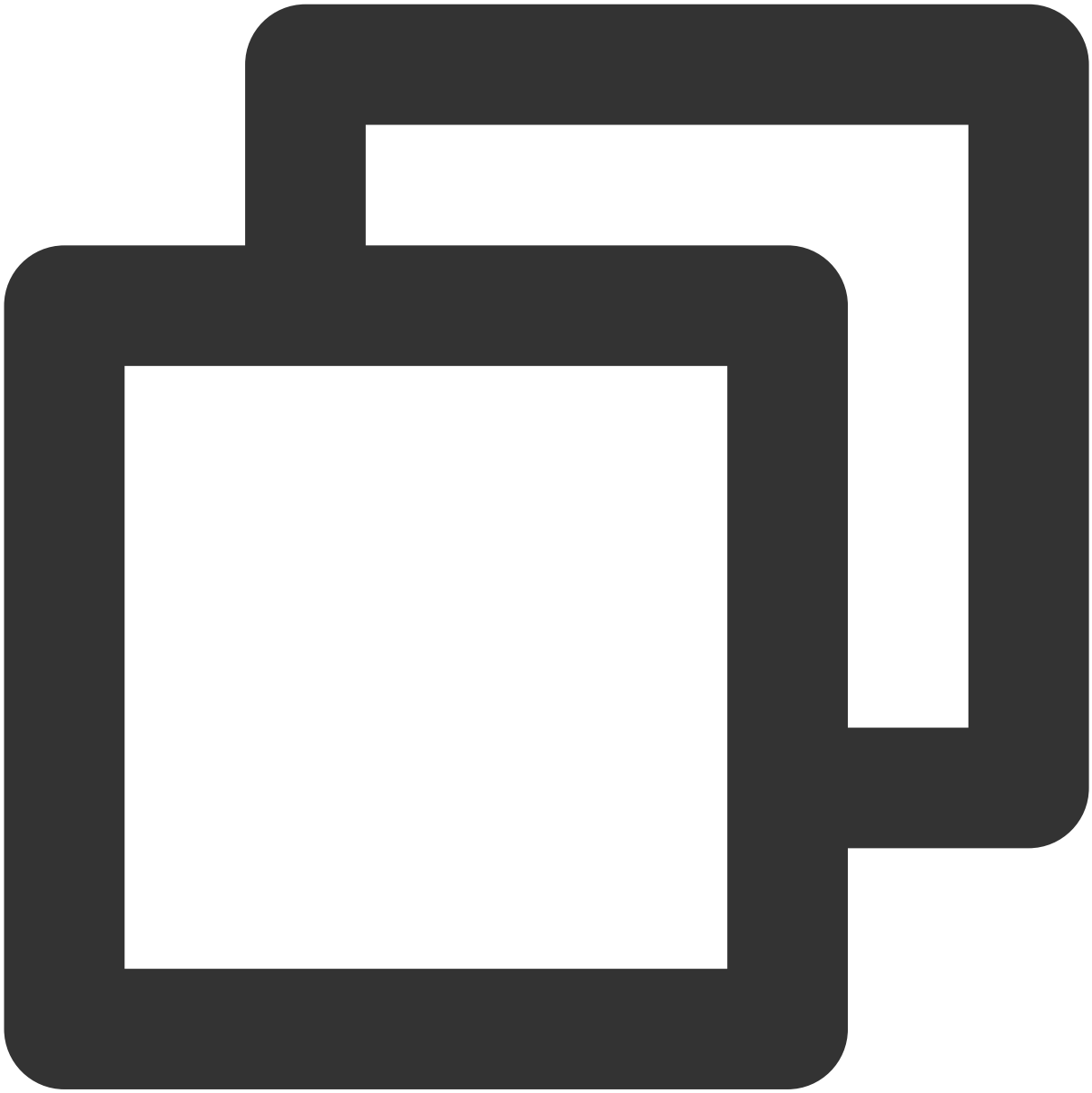
2. 配置 SDK 使用竖屏分辨率

在使用 TRTCCloud 的 setVideoEncoderParam 接口设置视频编码参数时，将 resMode 指定为 `TRTCVideoResolutionModePortrait` 即可。

示例代码如下：

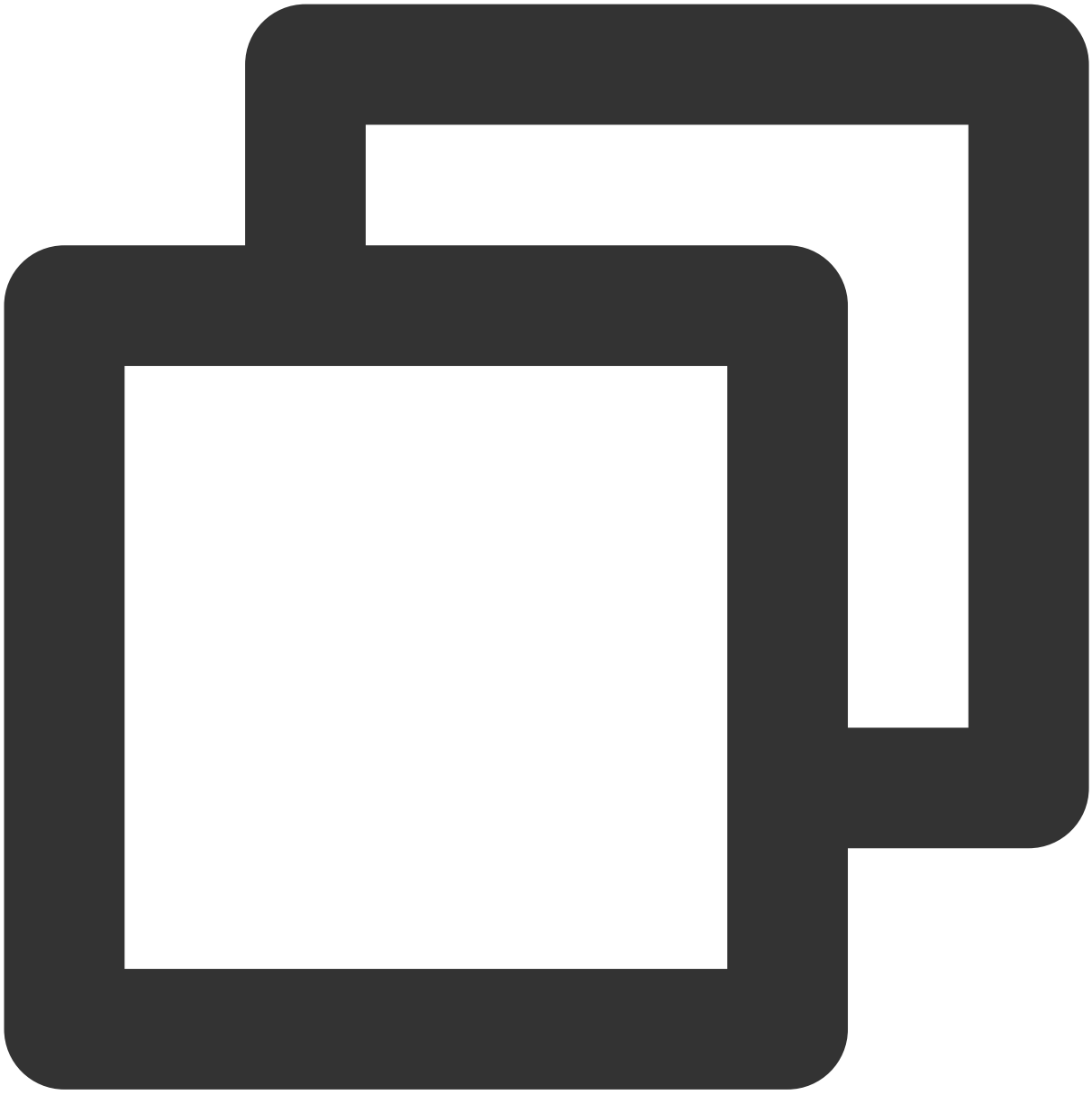
iOS

Android



```
TRTCVideoEncParam* encParam = [TRTCVideoEncParam new];
encParam.videoResolution = TRTCVideoResolution_640_360;
encParam.videoBitrate = 600;
encParam.videoFps = 15;
encParam.resMode = TRTCVideoResolutionModePortrait; //设置分辨率模式为竖屏模式

[trtc setVideoEncoderParam: encParam];
```



```
TRTCCloudDef.TRTCVideoEncParam encParam = new TRTCCloudDef.TRTCVideoEncParam();
encParam.videoResolution = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_640_360;
encParam.videoBitrate = 600;
encParam.videoFps = 15;
encParam.videoResolutionMode = TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT; //
trtc.setVideoEncoderParam(encParam);
```

横屏模式

如果希望 App 是横屏体验，那么您需要做的工作跟竖屏模式类似，只是将第一步和第二步中的参数都进行相应的调整即可。

尤其是 第二步 中，TRTCVideoEncParam 中的 resMode 值：

在 iOS 平台中应该指定为 `TRTCVideoResolutionModeLandscape` 。

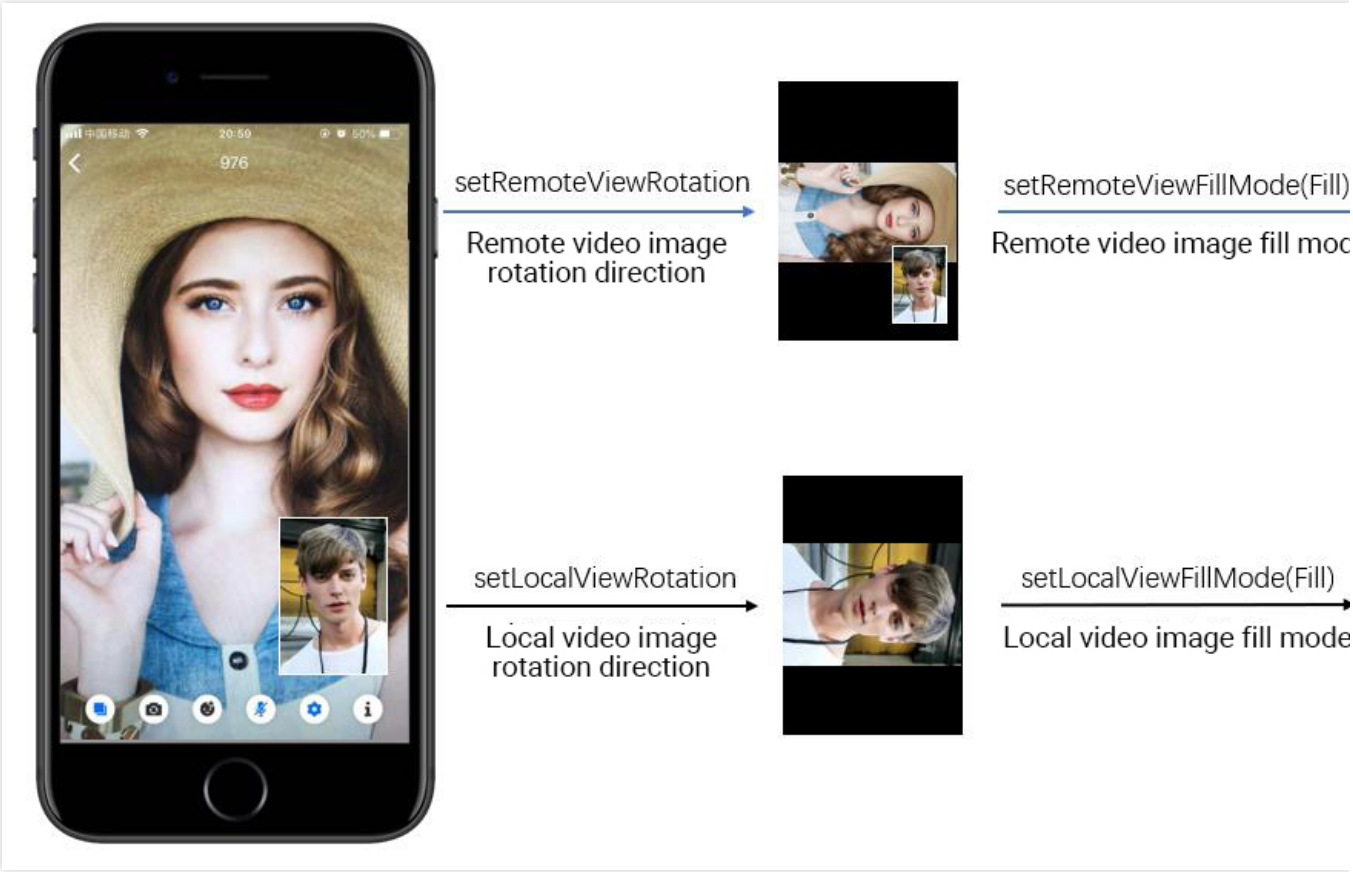
在 Android 平台中应该指定为 `TRTC_VIDEO_RESOLUTION_MODE_LANDSCAPE` 。

自定义控制

TRTC SDK 本身提供了大量的接口函数可以操控本地和远程画面的旋转方向和填充模式：

接口函数	功能作用	备注说明
setLocalViewRotation	本地预览画面的顺时针旋转角度	支持顺时针旋转90度、180度和270度三个方向
setLocalViewFillMode	本地预览画面的填充模式	是裁剪还是留黑边
setRemoteViewRotation	远端视频画面的顺时针旋转角度	支持顺时针旋转90度、180度和270度三个方向
setRemoteViewFillMode	远端视频画面的填充模式	是裁剪还是留黑边
setVideoEncoderRotation	设置编码器输出的画面顺时针旋转角度	支持顺时针旋转90度、180度和270度三个方向

版权所有：腾讯云计算（北京）有限责任公司



GSensorMode

考虑到画面旋转牵扯到录制和 CDN 旁路直播的各种适配问题， TRTC SDK 仅提供了一种简单的重力感应自适应功能，您可以通过 TRTCCloud 的 `setGSensorMode` 接口来开启。

该功能支持90度、180度、270度旋转的自适应，也就是当用户自己的手机旋转时，对方看到的画面朝向还是会保持不变。而且这种自适应是基于对编码器的方向调整而实现的，因此录制出的视频，以及小程序和 H5 端看到的视频画面也能做到保持原方向不变。

注意：

重力感应自适应的另一种实现方案是在每一帧视频信息里都带上当前视频的重力朝向，然后在远程用户那里自适应的调整渲染方向，但这种方案需要引入额外的转码资源才能解决录制出的视频朝向跟期望的视频朝向保持一致的问题，因此并不推荐。

版权所有：腾讯云计算（北京）有限责任公司

Electron

最近更新时间：2022-07-21 17:10:34

实时音视频（TRTC）支持自定义控制本地画面和远程画面的旋转方向和填充模式。

自定义控制本地画面

可以通过调用 setLocalRenderParams 设置本地渲染参数。

```
import TRTCCloud, {
TRTCRenderParams, TRTCVideoRotation, TRTCVideoFillMode,
TRTCVideoMirrorType
} from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation90,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Enable
);
trtcCloud.setLocalRenderParams(param);
const localUserDom = document.querySelector('local-user');
trtcCloud.startLocalPreview(localUserDom);
```

自定义控制远程画面画面

可以通过调用 setRemoteRenderParams 设置远端渲染参数。

```
import TRTCCloud, {
TRTCRenderParams, TRTCVideoRotation, TRTCVideoFillMode,
TRTCVideoMirrorType, TRTCVideoStreamType
} from 'trtc-electron-sdk';
const trtcCloud = new TRTCCloud();
const param = new TRTCRenderParams(
TRTCVideoRotation.TRTCVideoRotation180,
TRTCVideoFillMode.TRTCVideoFillMode_Fill,
TRTCVideoMirrorType.TRTCVideoMirrorType_Disable
);
const remoteUserId = 'remoteUser';
trtcCloud.setRemoteRenderParams(remoteUserId, TRTCVideoStreamType.TRTCVideoStream
TypeBig, param);
```

版权所有：腾讯云计算（北京）有限责任公司

```
const remoteUserDom = document.querySelector('remote-user');
trtcCloud.startRemoteView(remoteUserId, remoteUserDom, TRTCVideoStreamType.TRTCVi
deoStreamTypeBig);
```

版权所有：腾讯云计算（北京）有限责任公司

Web

最近更新时间：2023-09-11 17:49:16

本文主要介绍如何使用 TRTC Web SDK 控制视频渲染时的镜像、填充模式。

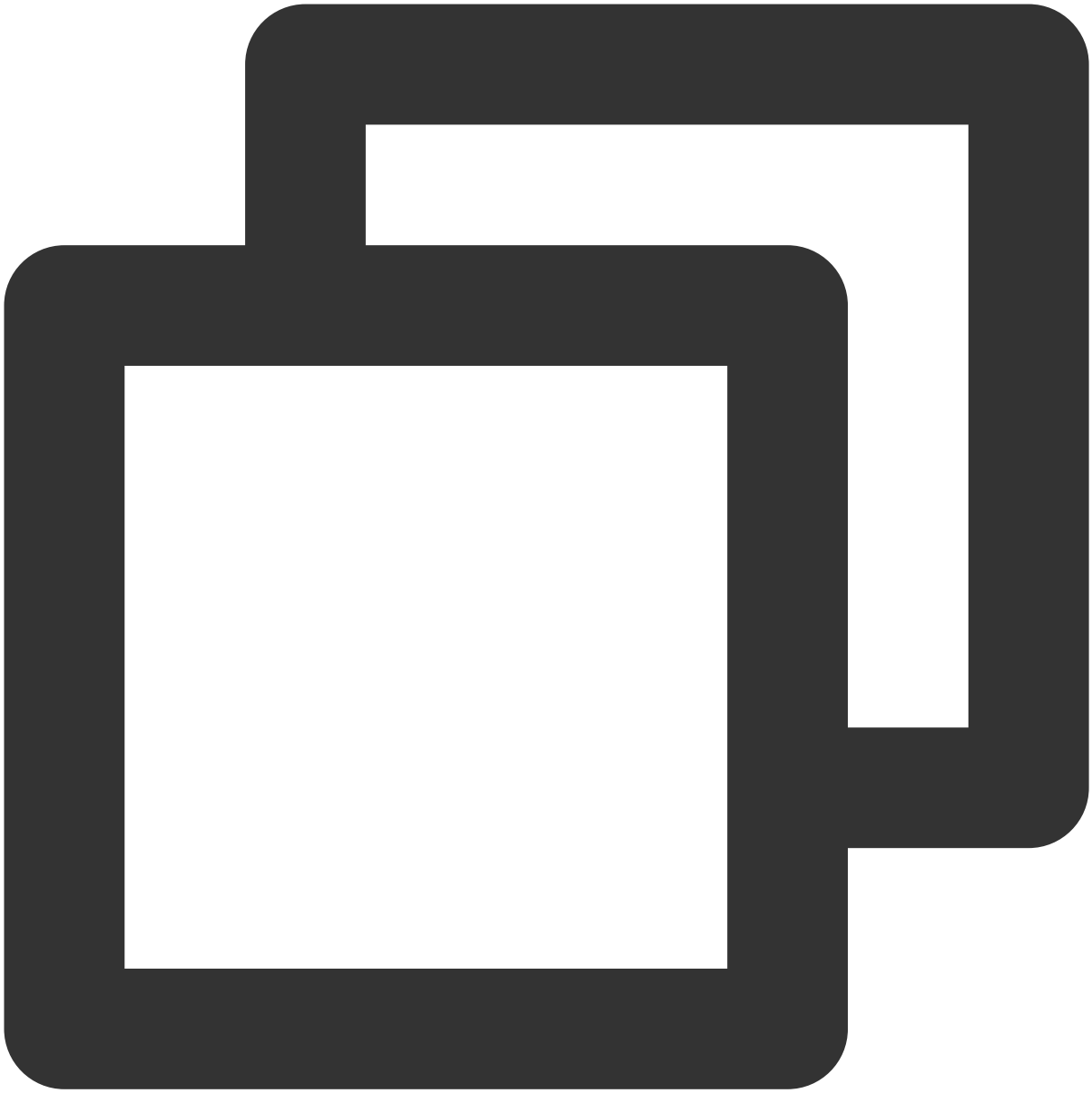
说明：

本教程基于5.x TRTC Web SDK 实现，若您使用4.x版本 SDK，可参考stream.play 的 objectFit 属性。

镜像

通过 trtc.startLocalVideo({ option: { mirror: true } }) 和 trtc.startRemoteVideo({ option: { mirror: true }}) 的方式来控制本地和远端视频的渲染镜像效果。

版权所有：腾讯云计算（北京）有限责任公司



```
// 本地摄像头渲染镜像，默认为 true
await trtc.startLocalVideo({ option: { mirror: true }});

// 动态更新参数
await trtc.updateLocalVideo({ option: { mirror: false }});

trtc.on(TRTC.EVENT.REMOTE_VIDEO_AVAILABLE, async ({ userId, streamType }) => {
  await trtc.startRemoteVideo({
    userId,
    streamType,
    // 您需在 DOM 中提前放置视频容器，建议以 `${userId}_${streamType}` 作为 element id。
```

```
    view: `${userId}_${streamType}`,
    // 镜像播放远端视频，默认为 false
    option: { mirror: true }
    });

  // 动态更新参数
  await trtc.updateRemoteVideo({ userId, streamType, option: { mirror: false }})
});
```

注意：

该镜像效果只是针对渲染时做处理，实际编码或解码出的画面是没有镜像效果的。您可以通过 canvas 自定义采集 的方式，对 canvas 进行翻转，从而实现编码镜像的效果。

填充模式

通过 trtc.startLocalVideo({ option: { fillMode: 'cover' } }) 和 trtc.startRemoteVideo({ option: { fillMode: 'cover' }}) 的方式来控制本地和远端视频的渲染填充模式。

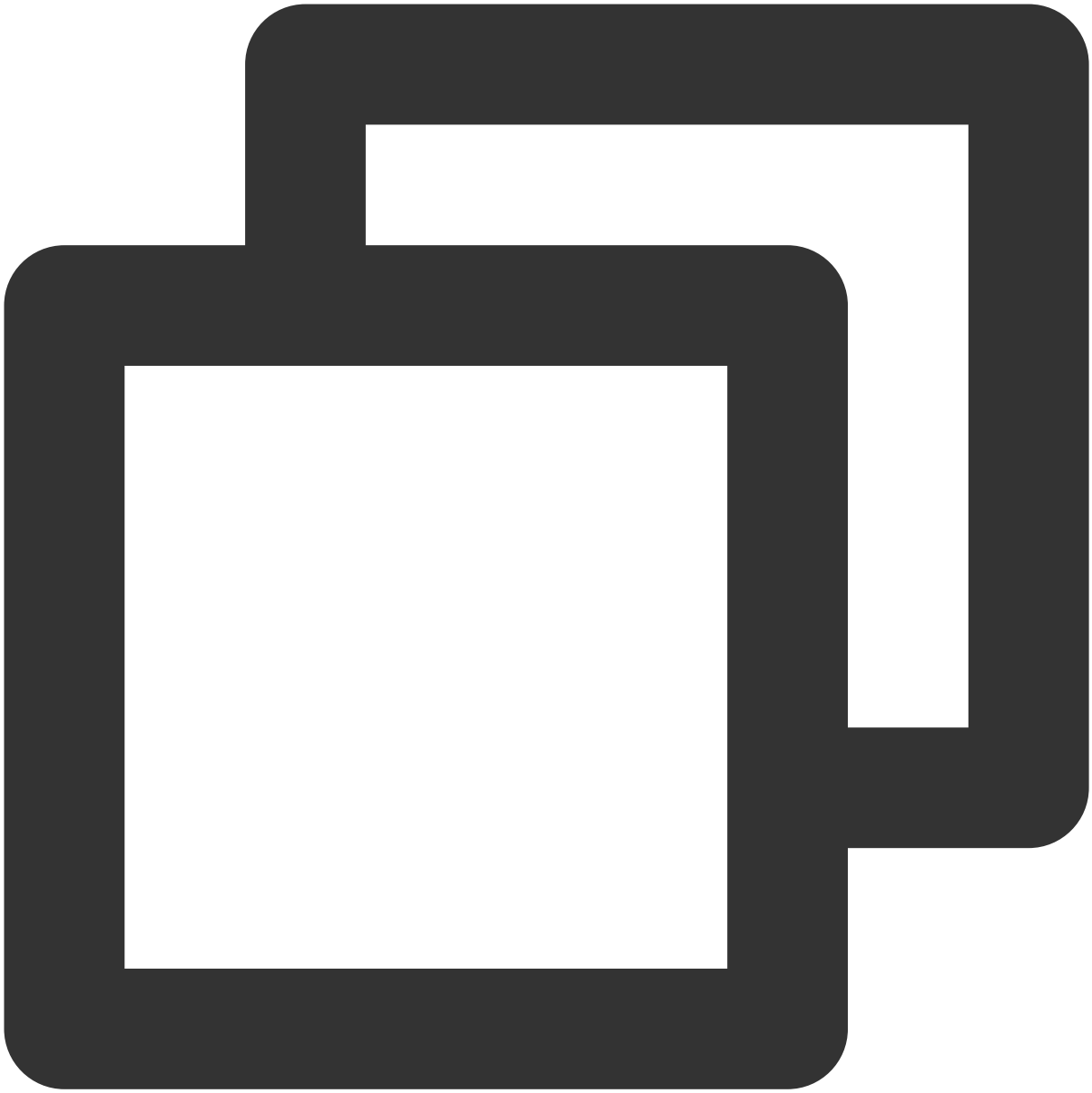
不同参数的含义：

`contain` 保留宽高比，在目标容器中完整显示画面，若宽高比与目标容器不匹配，则会以黑边填充。建议播放屏幕分享使用该参数。

`cover` 默认值，保留宽高比，在目标容器中显示，若宽高比与目标容器不匹配，则画面则会被裁剪，以填满整个目标容器。

`fill` 不保留宽高比，在目标容器中显示，若宽高比与目标容器不匹配，则画面会被拉伸，以填满整个模板容器。

参考：CSS object-fit 属性。



```
// 本地摄像头填充模式，默认为 cover
await trtc.startLocalVideo({ option: { fillMode: 'cover' }});

// 动态更新参数
await trtc.updateLocalVideo({ option: { fillMode: 'contain' }});

trtc.on(TRTC.EVENT.REMOTE_VIDEO_AVAILABLE, async ({ userId, streamType }) => {
  await trtc.startRemoteVideo({
    userId,
    streamType,
    // 您需在 DOM 中提前放置视频容器，建议以 `${userId}_${streamType}` 作为 element id。
```

```
    view: `${userId}_${streamType}`,
    option: { fillMode: 'contain' }
    });

  // 动态更新参数
  await trtc.updateRemoteVideo({ userId, streamType, option: { fillMode: 'cover' }}
});
```

Flutter

最近更新时间：2024-02-02 18:50:17

内容介绍

跟手机直播千篇一律的竖屏体验不同，实时音视频（TRTC）需要兼顾横屏和竖屏两种场景，因此就会有很多横竖屏的处理逻辑需要去应对，本文主要介绍：
如何实现竖屏模式，例如：微信的视频通话就是一个典型的竖屏体验模式。
如何实现横屏模式，例如：多人音视频房间 App（类似小鱼易连）往往都是采用横屏模式。
如何自定义控制本地画面和远程画面的旋转方向和填充模式。



平台支持

iOS	Android	Mac OS	Windows	Electron	Web 端

✓	✓	✓	✓	✓	×

竖屏模式

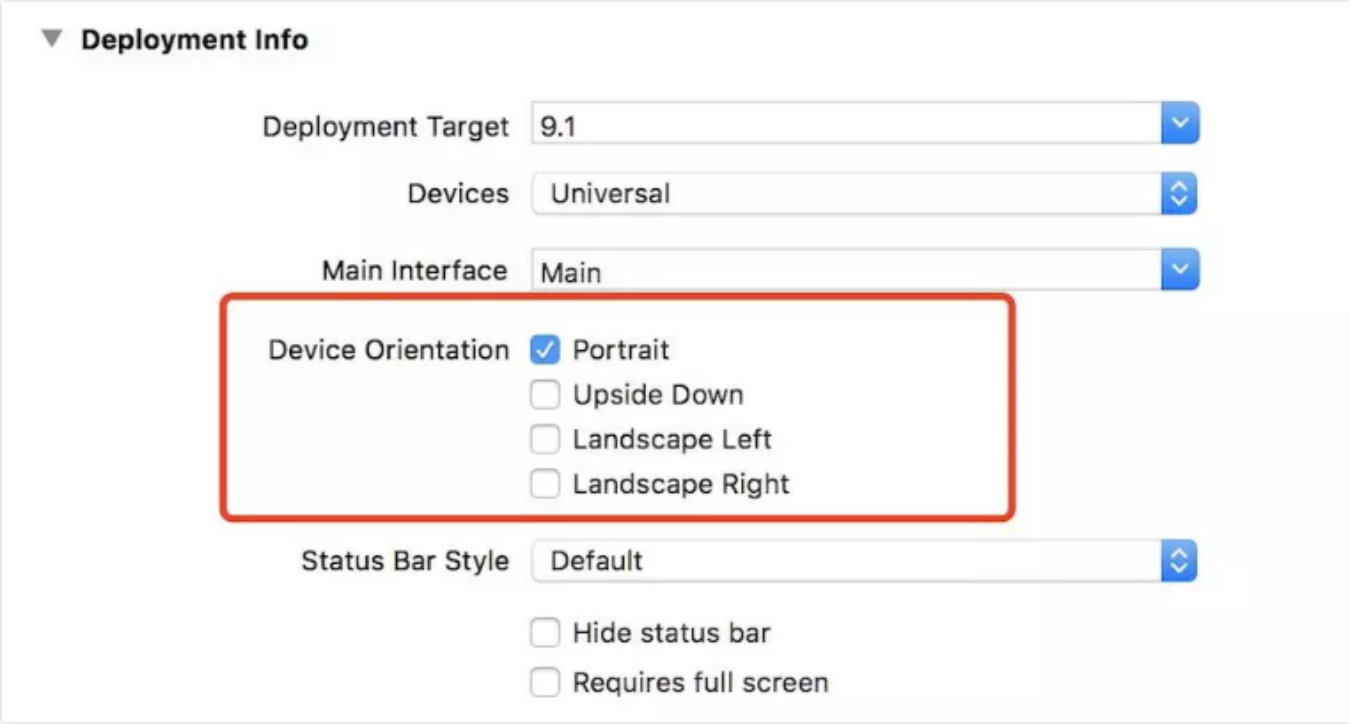
如果要实现类似微信视频通话的体验模式，需要做两项工作：

1. 配置 App 的 UI 界面为竖屏

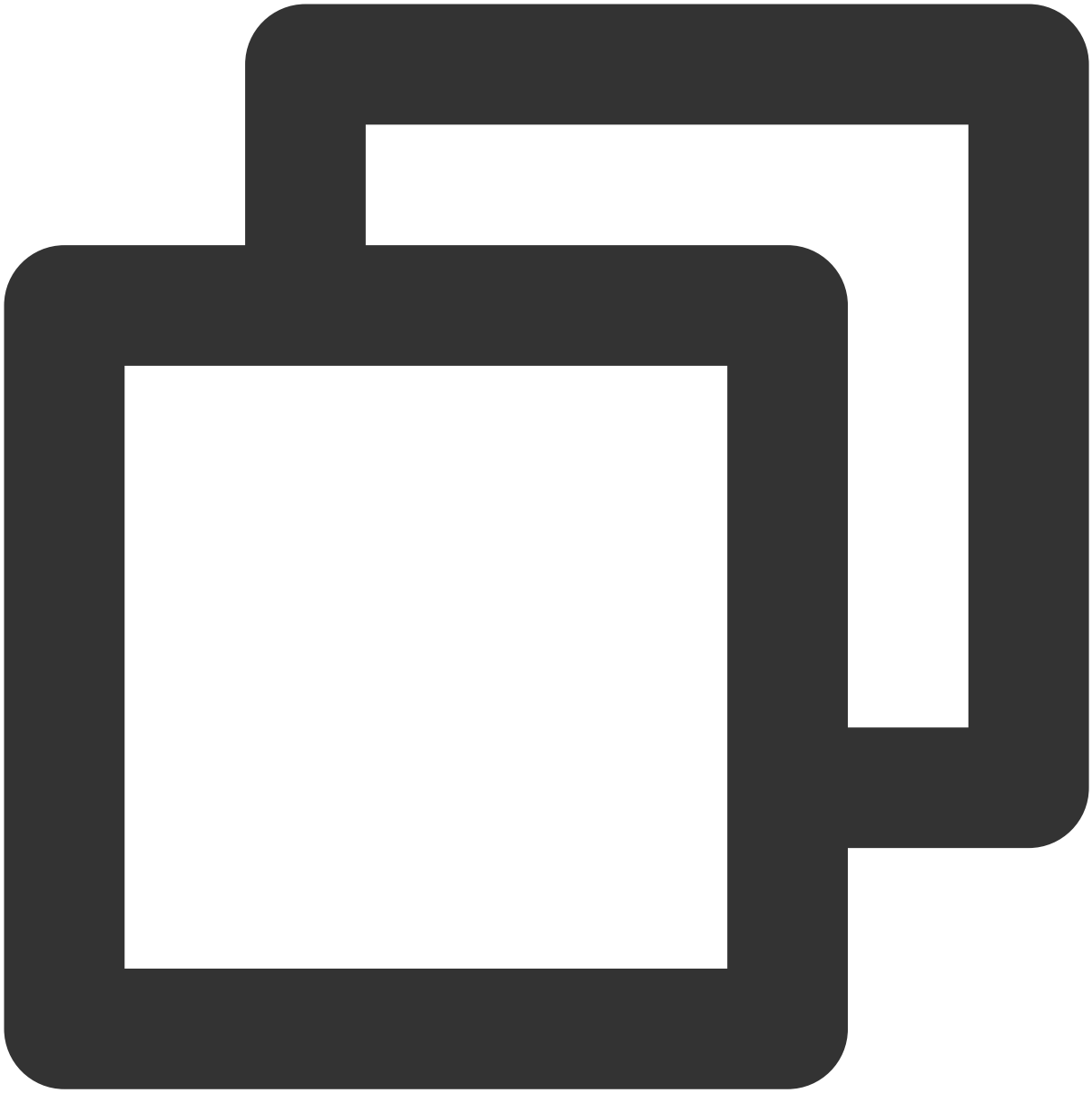
iOS 平台

Android 平台

可以直接在 XCode 的 **General** > **Deployment Info** > **Device Orientation** 中进行设置：



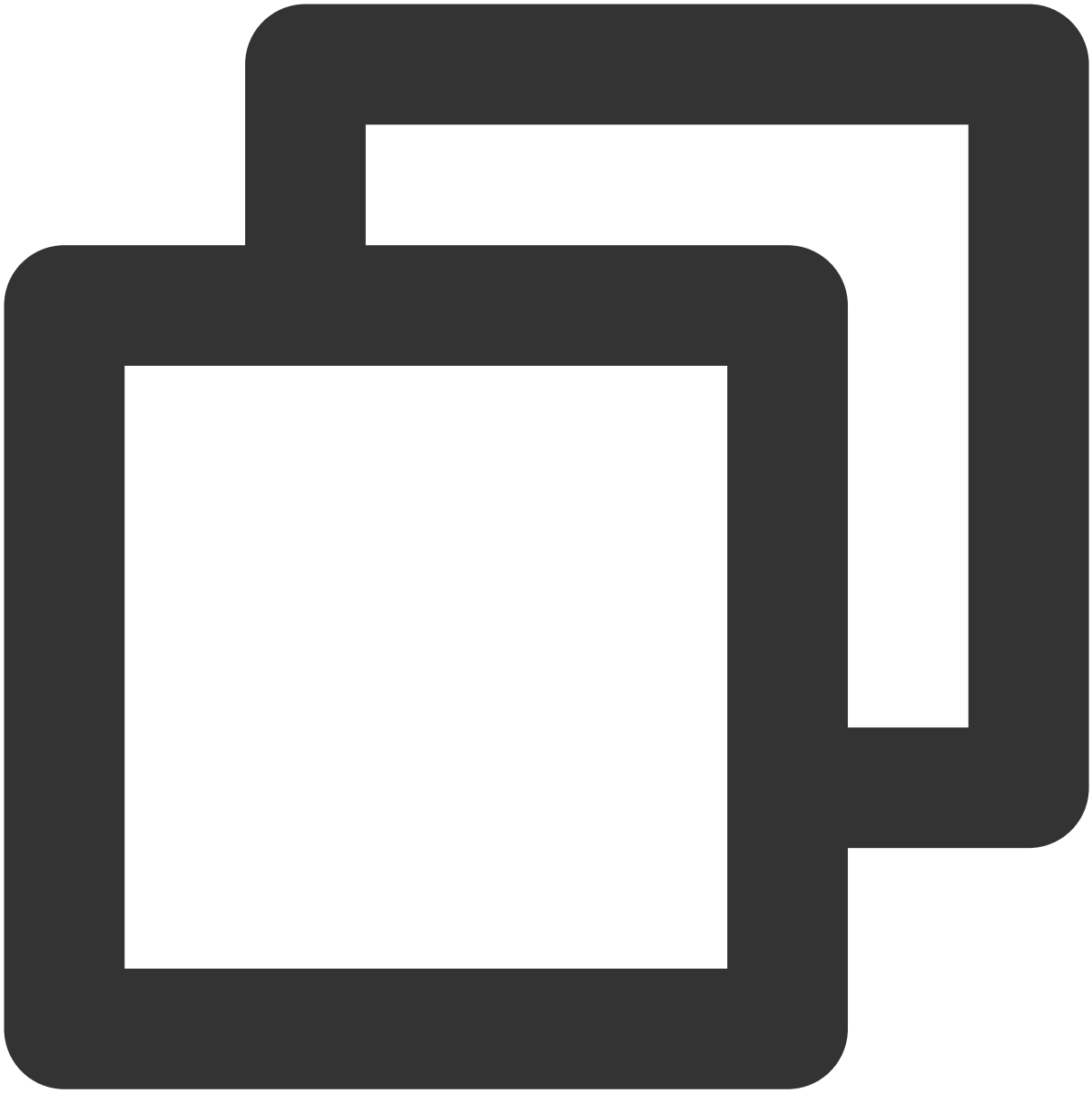
通过指定 activity 的 `screenOrientation` 属性为 portrait，即可指定该界面为竖屏模式：



```
<activity android:name=".trtc.TRTCMainActivity"  android:launchMode="singleTask" an
          android:screenOrientation="portrait" />
```

2. 配置 SDK 使用竖屏分辨率

在使用 TRTCCloud 的 setVideoEncoderParam 设置视频编码参数时，将 videoResolutionMode 指定为 `TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT` 即可。
示例代码如下：



```
trtcCloud.setVideoEncoderParam(TRTCVideoEncParam(
    videoFps: 15,
    videoResolution: TRTCCloudDef.TRTC_VIDEO_RESOLUTION_640_360,
    videoBitrate: 600,
    videoResolutionMode:
        TRTCCloudDef.TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT));
```

横屏模式

如果希望 App 是横屏体验，那么您需要做的工作跟竖屏模式类似，只是将第一步和第二步中的参数都进行相应的调整即可。

尤其是 第二步 中，TRTCVideoEncParam 中的 videoResolutionMode

值： `TRTC_VIDEO_RESOLUTION_MODE_LANDSCAPE` 。

自定义控制

TRTC SDK 本身提供了接口函数可以操控本地和远程画面的旋转方向和填充模式：

接口函数	功能作用	备注说明
setVideoEncoderRotation	设置编码器输出的画面顺时针旋转角度	目前仅支持 0 度和 180 度

GSensorMode

考虑到画面旋转牵扯到录制和 CDN 旁路直播的各种适配问题， TRTC SDK 仅提供了一种简单的重力感应自适应功能，您可以通过 TRTCCloud 的 `setGSensorMode` 接口来开启。

该功能支持90度、180度、270度旋转的自适应，也就是当用户自己的手机旋转时，对方看到的画面朝向还是会保持不变。而且这种自适应是基于对编码器的方向调整而实现的，因此录制出的视频，以及小程序和 H5 端看到的视频画面也能做到保持原方向不变。

注意：

重力感应自适应的另一种实现方案是在每一帧视频信息里都带上当前视频的重力朝向，然后在远程用户那里自适应的调整渲染方向，但这种方案需要引入额外的转码资源才能解决录制出的视频朝向跟期望的视频朝向保持一致的问题，因此并不推荐。

版权所有：腾讯云计算（北京）有限责任公司

07. 常见问题

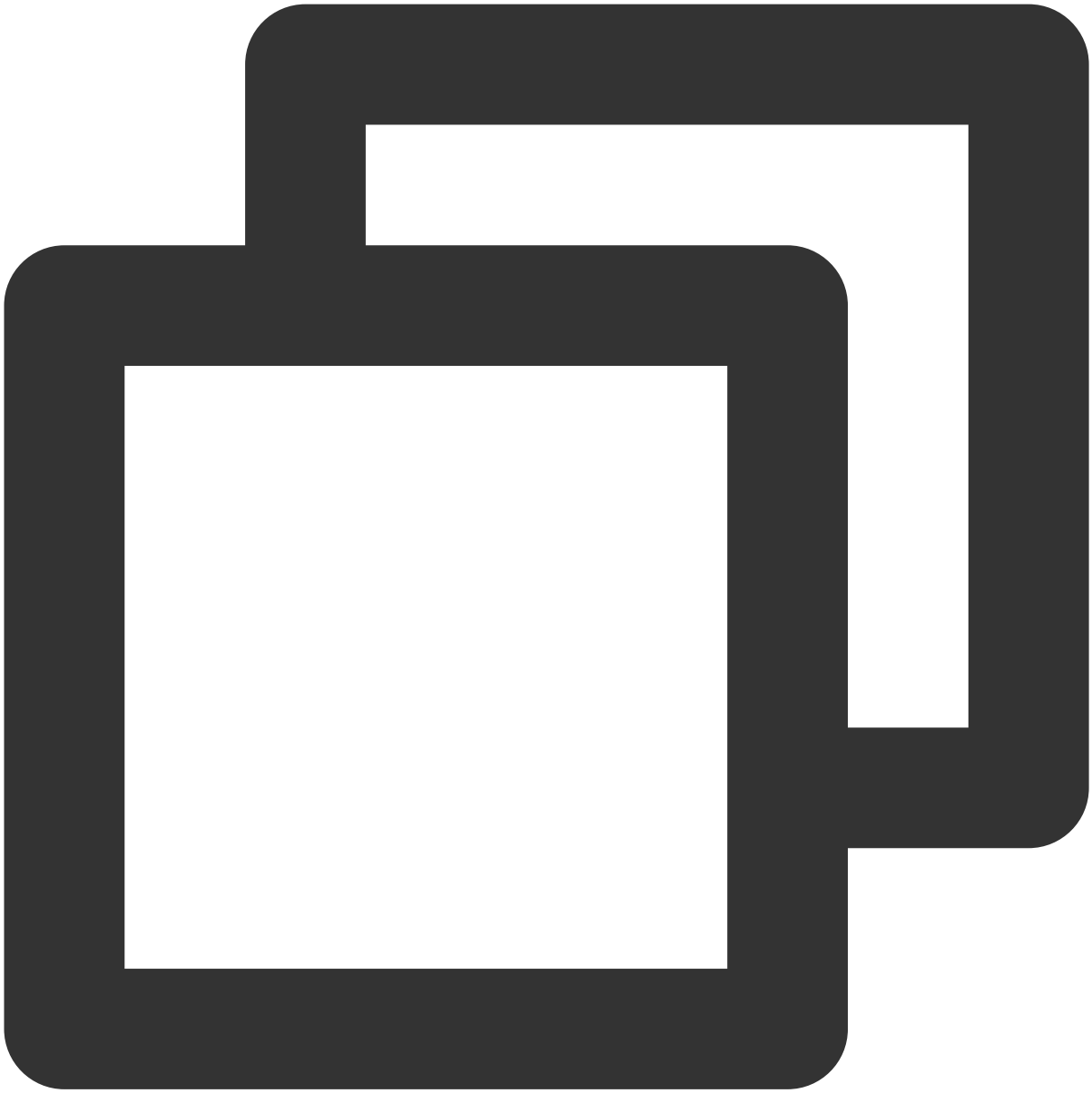
全平台

最近更新时间：2023-05-30 17:15:48

什么是 UserSig？

UserSig 是腾讯云设计的一种安全保护签名，目的是为了阻止恶意攻击者盗用您的云服务使用权。
目前，腾讯云的实时音视频（TRTC）、即时通信（IM）以及移动直播（MLVB）等服务都采用了该套安全保护机制。要使用这些服务，您都需要在相应 SDK 的初始化或登录函数中提供 SDKAppID，UserID 和 UserSig 三个关键信息。
其中 SDKAppID 用于标识您的应用，UserID 用于标识您的用户，而 UserSig 则是基于前两者计算出的安全签名，它由 **HMAC SHA256** 加密算法计算得出。只要攻击者不能伪造 UserSig，就无法盗用您的云服务流量。
UserSig 的计算原理如下图所示，其本质就是对 SDKAppID、UserID、ExpireTime 等关键信息进行了一次哈希加密：

版权所有：腾讯云计算（北京）有限责任公司



```
//UserSig 计算公式，其中 secretkey 为计算 usersig 用的加密密钥

usersig = hmacsha256(secretkey, (userid + sdkappid + currtime + expire +
                                 base64(userid + sdkappid + currtime + expire)))
```

说明：

`currtime` 为当前系统的时间，`expire` 为签名过期的时间。

如需了解 UserSig 具体计算获取方法，请参见 UserSig 详情说明。

实时音视频最多可以同时创建多少个房间？

支持同时并发存在4294967294个房间，累计房间数量无限制。

实时音视频延时大约多少？

全球端到端平均延时小于300ms。

实时音视频接入 PC 端是否支持屏幕分享功能？

支持，您可以参考如下文档：
屏幕分享（Windows）
屏幕分享（Mac）
屏幕分享（Web）
屏幕分享接口详情请参见 Windows（C++）API 。另外，您也可以使用 Electron 接口。

TRTC 支持哪些平台？

支持的平台包括 iOS、Android、Windows(C++)、Windows(C#)、Mac、Web、Electron，更多详情请参见 平台支持。

实时音视频最多可以支持多少个人同时通话？

通话模式下，单个房间最多支持300人同时在线，最多支持50人同时开启摄像头或麦克风。
直播模式下，单个房间支持10万人以观众身份在线观看，最多支持50人以主播身份开启摄像头或麦克风。

TRTC 怎么实现直播场景类应用？

TRTC 专门针对在线直播场景推出了10万人低延时互动直播解决方案，能保证主播与连麦主播的最低延时到200ms，普通观众的延时在1s以内，并且超强的抗弱网能力适应移动端复杂的网络环境。
具体操作指引请参考 跑通直播模式。

TRTC 直播支持什么角色？有什么区别？

直播场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）支持 TRTCRoleAnchor（主播）和 TRTCRoleAudience（观众）两种角色，区别是主播角色可以同时上行、下行音视频数据，观众角色只支持下行播放其他人的数据。您可以通过调用 switchRole() 进行角色切换。

版权所有：腾讯云计算（北京）有限责任公司

TRTC 房间支不支持踢人、禁止发言、静音？

支持。
如果是简单的信令操作，可以使用 TRTC 的自定义信令接口 sendCustomCmdMsg，开发者自己定义相应的控制信令，收到控制信令的通话方执行对应操作即可。例如，踢人就是定义一个踢人的信令，收到此信令的用户就自行退出房间。
如果是需要实现更完善的操作逻辑，建议开发者通过 即时通信 IM 来实现相关逻辑，将 TRTC 的房间与 IM 群组进行映射，在 IM 群组中收发自定义消息来实现相应的操作。

TRTC 音视频流是否支持通过 CDN 拉流观看？

支持，详情请参见 实现 CDN 直播观看。

在 iOS 端是否支持 Swift 集成？

支持，直接按照支持集成三方库的流程集成 SDK 即可，还可以参考 跑通Demo(iOS&Mac)。

Web 端 SDK 的支持哪些浏览器？

目前主要在桌面版 Chrome 浏览器、桌面版 Safari 浏览器以及移动版的 Safari 浏览器上有较为完整的支持，其他平台（例如 Android 平台的浏览器）支持情况均比较差，具体详情请参见 支持的平台。
您可以在浏览器打开 WebRTC 能力测试 测试是否完整的支持 WebRTC 的功能。

Web 端 SDK 日志中报错 NotFoundError、NotAllowedError、NotReadableError、OverConstrainedError 以及 AbortError 分别是什么意思？

错误名	描述	处理建议
NotFoundError	找不到满足请求参数的媒体类型（包括音频、视频、屏幕分享）。 例如：PC 没有摄像头，但是请求浏览器获取视频流，则会报此错误。	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外设。
NotAllowedError	用户拒绝了当前的浏览器实例的访问音频、视频、屏幕分享请求。	提示用户不授权摄像头/麦克风访问将无法进行音视频通话。
NotReadableError	用户已授权使用相应的设备，但由于操作系统上某个硬件、浏览器或者网页层面发生的错误导致设备无法被访问。	根据浏览器的报错信息处理，并提示用户“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”。
OverConstrainedError	cameraId/microphoneId 参数的值无	检查 cameraId/microphoneId 是否是通过获

	效。	取设备信息接口返回的值
AbortError	由于某些未知原因导致设备无法被使用。	-

更多详情请参考 initialize。

怎么确认 Web 端 SDK 是否能正常获取到设备（摄像头/麦克风）列表？

1. 检查浏览器是否能够正常使用设备：
直接在页面打开控制台，输入 `navigator.mediaDevices.enumerateDevices()` 确认能否获取到设备列表。
如果正常获取到设备会返回一个 Promise，里面会有 MediaDeviceInfo 对象数组，数组里的每个对象对应一个可用的媒体设备。
如果枚举失败，Promise 将返回 rejected，说明浏览器都没有识别到设备，需检查浏览器或设备。
2. 如果能获取设备列表，则输入 `navigator.mediaDevices.getUserMedia({ audio: true, video: true })` 确认能否正常返回 MediaStream 对象，不能正常返回说明浏览器没有获取到数据，需检查浏览器的配置。

直播、互动直播、实时音视频以及旁路直播有什么区别和关系？

直播（关键词：一对多，RTMP/HLS/HTTP-FLV，CDN）
直播分为推流端、播放端以及直播云服务，云服务使用 CDN 进行直播流的分发。推流使用的是通用标准的协议 RTMP，经过 CDN 分发后，播放时一般可以选择 RTMP、HTTP-FLV 或 HLS（H5 支持）等方式进行观看。
互动直播（关键词：连麦、PK）
互动直播是一种业务形式，指主播与观众之间进行互动连麦，主播与主播之间进行互动PK的一种直播类型。
实时音视频（关键词：多人互动，UDP 私有协议，低延时）
实时音视频（Tencent Real-Time Communication，TRTC）主要应用场景是音视频互动和低延时直播，使用基于 UDP 的私有协议，其延迟可低至100ms，典型的场景就是 zoom meeting、FaceTime、大班课等。 腾讯云实时音视频（TRTC）覆盖全平台，除了 iOS/Android/Windows 之外，还支持 WebRTC 互通，并且支持通过云端混流的方式将画面旁路直播到 CDN。
旁路直播（关键词：云端混流，RTC 旁路转推，CDN）
旁路直播是一种技术，指的是将低延时连麦房间里的多路推流画面复制出来，在云端将画面混合成一路，并将混流后的画面推流给直播 CDN 进行分发播放。

TRTC 如何查看通话时长和使用量？

可在实时音视频控制台的【用量统计】页面查看。

TRTC 出现卡顿怎么排查？

可以通过对应的 RoomID、UserID 在实时音视频控制台的【监控仪表盘】页面查看通话质量：
通过接受端视角查看发送端和接收端用户情况。
查看发送端和接收端是否丢包率比较高，如果丢包率过高一般是网络状况不稳定导致卡顿。
查看帧率和 CPU 占用率，帧率比较低和 CPU 使用率过高都会导致卡顿现象。

TRTC 出现画质不佳，模糊、马赛克等现象怎么排查？

清晰度主要和码率有关，检查 SDK 码率是否配置的比较低，如果高分辨率低码率容易产生马赛克现象。
TRTC 会通过云端 QOS 流控策略，根据网络状况动态调整码率、分辨率，网络比较差时容易降低码率导致清晰度下降。
检查进房时使用的 VideoCall 模式还是 Live 模式，针对通话场景 VideoCall 模式主打低延时和保流畅，所以在弱网情况下会更容易牺牲画质确保流畅，对画质更加看重的场景建议使用 Live 模式。

如何查询 SDK 最新版本号？

若您使用自动加载的方法，`latest.release` 为匹配最新版并进行自动加载，不需要对版本号进行修改。具体集成方法请参见 一分钟集成 SDK。
当前 SDK 最新版本号可通过发布日志查看，具体请参见：
iOS & Android 端，请参见 发布日志（App）。
Web 端，请参见 发布日志（Web）。
Electron 端，请参见 发布日志（Electron）。

API-参考手册

iOS&Mac

API 概览

最近更新时间：2024-06-06 15:26:14

API OVERVIEW

创建实例和事件回调

函数列表	描述
sharedInstance	创建 TRTCCloud 实例（单例模式）
destroySharedInstance	销毁 TRTCCloud 实例（单例模式）
addDelegate:	添加 TRTC 事件回调
removeDelegate:	移除 TRTC 事件回调
delegateQueue	设置驱动 TRTCCloudDelegate 事件回调的队列

房间相关接口函数

函数列表	描述
enterRoom:appScene:	进入房间
exitRoom	离开房间
switchRole:	切换角色
switchRole:privateMapKey:	切换角色（支持设置权限位）
switchRoom:	切换房间
connectOtherRoom:	请求跨房通话
disconnectOtherRoom	退出跨房通话

版权所有：腾讯云计算（北京）有限责任公司

setDefaultStreamRecvMode:video:	设置订阅模式（需要在进入房前设置才能生效）
createSubCloud	创建子房间实例（用于多房间并发观看）
destroySubCloud:	销毁子房间实例
updateOtherRoomForwardMode:	更改跨房主播在本房间的上行能力

CDN 相关接口函数

函数列表	描述
startPublishing:type:	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream:	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig:	设置云端混流的排版布局和转码参数
startPublishMediaStream:encoderParam:mixingConfig:	开始发布媒体流
updatePublishMediaStream:publishTarget:encoderParam:mixingConfig:	更新发布媒体流
stopPublishMediaStream:	停止发布媒体流

视频相关接口函数

函数列表	描述
startLocalPreview:view:	开启本地摄像头的预览画面（移动端）
startLocalPreview:	开启本地摄像头的预览画面（桌面端）
updateLocalView:	更新本地摄像头的预览画面

版权所有：腾讯云计算（北京）有限责任公司

stopLocalPreview	停止摄像头预览
muteLocalVideo:mute:	暂停/恢复发布本地的视频流
setVideoMuteImage:fps:	设置本地画面被暂停期间的替代图片
startRemoteView:streamType:view:	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView:streamType:forUser:	更新远端用户的视频渲染控件
stopRemoteView:streamType:	停止订阅远端用户的视频流，并释放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源
muteRemoteVideoStream:streamType:mute:	暂停/恢复订阅远端用户的视频流
muteAllRemoteVideoStreams:	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam:	设置视频编码器的编码参数
setNetworkQosParam:	设置网络质量控制的相关参数
setLocalRenderParams:	设置本地画面的渲染参数
setRemoteRenderParams:streamType:params:	设置远端画面的渲染模式
enableEncSmallVideoStream:withQuality:	开启大小画面双路编码模式
setRemoteVideoStreamType:type:	切换指定远端用户的大小画面
snapshotVideo:type:sourceType:	视频画面截图
setPerspectiveCorrectionWithUser:srcPoints:dstPoints:	视频画面透视校正坐标设置
setGravitySensorAdaptiveMode:	设置重力感应的适配模式（11.7 及以上版本）

音频相关接口函数

函数列表	描述
startLocalAudio:	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布

muteLocalAudio:	暂停/恢复发布本地的音频流
muteRemoteAudio:mute:	暂停/恢复播放远端的音频流
muteAllRemoteAudio:	暂停/恢复播放所有远端用户的音频流
setAudioRoute:	设置音频路由
setRemoteAudioVolume:volume:	设定某一个远端用户的声音播放音量
setAudioCaptureVolume:	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume:	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
enableAudioVolumeEvaluation:withParams:	启用音量大小提示
startAudioRecording:	开始录音
stopAudioRecording	停止录音
startLocalRecording:	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams:	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect:	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition:	设置 3D 音效中远端用户坐标信息
set3DSpatialReceivingRange:range:	设置指定用户所发出声音的可被接收范围

设备管理相关接口

函数列表	描述
getDeviceManager	获取设备管理类（TXDeviceManager）

美颜特效和图像水印

函数列表	描述
getBeautyManager	获取美颜管理类（TXBeautyManager）
setWatermark:streamType:rect:	添加水印

背景音乐和声音特效

函数列表	描述
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集（iOS 端暂未支持）
stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）
setSystemAudioLoopbackVolume:	设置系统声音的采集音量

屏幕分享相关接口

函数列表	描述
startScreenCaptureInApp:encParam:	开始应用内的屏幕分享（仅支持 iOS 13.0 及以上系统）
startScreenCaptureByReplaykit:encParam:appGroup:	开始全系统的屏幕分享（仅支持 iOS 11.0 及以上系统）
startScreenCapture:streamType:encParam:	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
getScreenCaptureSourcesWithThumbnailSize:iconSize:	枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）
selectScreenCaptureTarget:rect:capturesCursor:highlight:	选取要分享的屏幕或窗口（该接口仅支持

	Mac OS 系统）
setSubStreamEncoderParam:	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）
setSubStreamMixVolume:	设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）
addExcludedShareWindow:	将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）
removeExcludedShareWindow:	将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
removeAllExcludedShareWindows	将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
addIncludedShareWindow:	将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）
removeIncludedShareWindow:	将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
removeAllIncludedShareWindows	将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

自定义采集和自定义渲染

函数列表	描述
enableCustomVideoCapture:enable:	启用/关闭视频自定义采集模式
sendCustomVideoData:frame:	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture:	启用音频自定义采集模式
sendCustomAudioData:	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame:playout:	启用/关闭自定义音轨
mixExternalAudioFrame:	向 SDK 混入自定义音轨
setMixExternalAudioVolume:playoutVolume:	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳

版权所有：腾讯云计算（北京）有限责任公司

setLocalVideoProcessDelegete:pixelFormat:bufferType:	设置第三方美颜的视频数据回调
setLocalVideoRenderDelegate:pixelFormat:bufferType:	设置本地视频自定义渲染回调
setRemoteVideoRenderDelegate:delegate:pixelFormat:bufferType:	设置远端视频自定义渲染回调
setAudioFrameDelegate:	设置音频数据自定义回调
setCapturedAudioFrameDelegateFormat:	设置本地麦克风采集出的音频帧回调格式
setLocalProcessedAudioFrameDelegateFormat:	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameDelegateFormat:	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering:	开启音频自定义播放
getCustomAudioRenderingFrame:	获取可播放的音频数据

自定义消息发送接口

函数列表	描述
sendCustomCmdMsg:data:reliable:ordered:	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg:repeatCount:	使用 SEI 通道发送自定义消息给房间内所有用户

网络测试接口

函数列表	描述
startSpeedTest:	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速

调试相关接口

函数列表	描述
getSDKVersion	获取 SDK 版本信息

setLogLevel:	设置 Log 输出级别
setConsoleEnabled:	启用/禁用控制台日志打印
setLogCompressEnabled:	启用/禁用日志的本地压缩
setLogDirPath:	设置本地日志的保存路径
setLogDelegate:	设置日志回调
showDebugView:	显示仪表盘
setDebugViewMargin:margin:	设置仪表盘的边距
callExperimentalAPI:	调用实验性接口

加密接口

函数列表	描述
enablePayloadPrivateEncryption:params:	开启或关闭媒体流私有加密

错误和警告事件

函数列表	描述
onError:errMsg:extInfo:	错误事件回调
onWarning:warningMsg:extInfo:	警告事件回调

房间相关事件回调

函数列表	描述
onEnterRoom:	进入房间成功与否的事件回调
onExitRoom:	离开房间的事件回调
onSwitchRole:errMsg:	切换角色的事件回调
onSwitchRoom:errMsg:	切换房间的结果回调

版权所有：腾讯云计算（北京）有限责任公司

onConnectOtherRoom:errCode:errMsg:	请求跨房通话的结果回调
onDisconnectOtherRoom:errMsg:	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode:errMsg:	更改跨房主播上行能力的结果回调

用户相关事件回调

函数列表	描述
onRemoteUserEnterRoom:	有用户加入当前房间
onRemoteUserLeaveRoom:reason:	有用户离开当前房间
onUserVideoAvailable:available:	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable:available:	某远端用户发布/取消了辅路视频画面
onUserAudioAvailable:available:	某远端用户发布/取消了自己的音频
onFirstVideoFrame:streamType:width:height:	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame:	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame:	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated:streamType:streamStatus:reason:extrainfo:	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated:streamStatus:reason:extrainfo:	远端音频状态变化的事件回调
onUserVideoSizeChanged:streamType:newWidth:newHeight:	用户视频大小发生改变回调

网络和技术指标统计回调

函数列表	描述
onNetworkQuality:remoteQuality:	网络质量的实时统计回调
onStatistics:	音视频技术指标的实时统计回调
onSpeedTestResult:	网速测试的结果回调

与云端连接情况的事件回调

函数列表	描述
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复

硬件设备相关事件回调

函数列表	描述
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onAudioRouteChanged:fromRoute:	当前音频路由发生变化（仅适用于移动设备）
onUserVoiceVolume:totalVolume:	音量大小的反馈回调
onDevice:type:stateChanged:	本地设备的通断状态发生变化（仅适用于桌面系统）
onAudioDeviceCaptureVolumeChanged:muted:	当前麦克风的系统采集音量发生变化
onAudioDevicePlayoutVolumeChanged:muted:	当前系统的播放音量发生变化
onSystemAudioLoopbackError:	系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）

自定义消息的接收事件回调

函数列表	描述
onRecvCustomCmdMsgUserId:cmdID:seq:message:	收到自定义消息的事件回调
onMissCustomCmdMsgUserId:cmdID:errCode:missed:	自定义消息丢失的事件回调
onRecvSEIMsg:message:	收到 SEI 消息的回调

CDN 相关事件回调

函数列表	描述
onStartPublishing:errMsg:	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing:errMsg:	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream:errMsg:	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream:errMsg:	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig:errMsg:	设置云端混流的排版布局和转码参数的事件回调
onStartPublishMediaStream:code:message:extraInfo:	开始发布媒体流的事件回调
onUpdatePublishMediaStream:code:message:extraInfo:	更新媒体流的事件回调
onStopPublishMediaStream:code:message:extraInfo:	停止媒体流的事件回调
onCdnStreamStateChanged:status:code:msg:extraInfo:	RTMP/RTMPS 推流状态发生改变回调

屏幕分享相关事件回调

函数列表	描述
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused:	屏幕分享暂停的事件回调
onScreenCaptureResumed:	屏幕分享恢复的事件回调

版权所有：腾讯云计算（北京）有限责任公司

onScreenCaptureStoped:	屏幕分享停止的事件回调

本地录制和本地截图的事件回调

函数列表	描述
onLocalRecordBegin:storagePath:	本地录制任务已经开始的事件回调
onLocalRecording:storagePath:	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment:	本地录制分片的事件回调
onLocalRecordComplete:storagePath:	本地录制任务已经结束的事件回调

废弃的事件回调

函数列表	描述
onUserEnter:	有主播加入当前房间（已废弃）
onUserExit:reason:	有主播离开当前房间（已废弃）
onAudioEffectFinished:code:	音效播放已结束（已废弃）

视频数据自定义回调

函数列表	描述
onRenderVideoFrame:userId:streamType:	自定义视频渲染回调
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame:dstFrame:	用于对接第三方美颜组件的视频处理回调
onGLContextDestory	SDK 内部 OpenGL 环境被销毁的通知

音频数据自定义回调

函数列表	描述
onCapturedAudioFrame:	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame:	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onRemoteUserAudioFrame:userId:	混音前的每一路远程用户的音频数据
onMixedPlayAudioFrame:	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame:	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）
onVoiceEarMonitorAudioFrame:	耳返的音频数据

更多事件回调接口

函数列表	描述
onLog:LogLevel:WhichModule:	本地 LOG 的打印回调

人声相关的特效接口

函数列表	描述
enableVoiceEarMonitor:	开启耳返
setVoiceEarMonitorVolume:	设置耳返音量
setVoiceReverbType:	设置人声的混响效果
setVoiceChangerType:	设置人声的变声特效
setVoiceVolume:	设置语音音量
setVoicePitch:	设置语音音调

背景音乐的相关接口

函数列表	描述

startPlayMusic:onStart:onProgress:onComplete:	开始播放背景音乐
stopPlayMusic:	停止播放背景音乐
pausePlayMusic:	暂停播放背景音乐
resumePlayMusic:	恢复播放背景音乐
setAllMusicVolume:	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume:volume:	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume:volume:	设置某一首背景音乐的本地音量的大小
setMusicPitch:pitch:	调整背景音乐的音调高低
setMusicSpeedRate:speedRate:	调整背景音乐的变速效果
getMusicCurrentPosInMS:	获取背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS:	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInMS:pts:	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate:speedRate:	调整搓碟的变速效果
preloadMusic:onProgress:onError:	预加载背景音乐
getMusicTrackCount:	获取背景音乐的音轨数量
setMusicTrack:track:	指定背景音乐的播放音轨

美颜相关接口

函数列表	描述
setBeautyStyle:	设置美颜（磨皮）算法
setBeautyLevel:	设置美颜级别
setWhitenessLevel:	设置美白级别
enableSharpnessEnhancement:	开启清晰度增强
setRuddyLevel:	设置红润级别
setFilter:	设置色彩滤镜效果

版权所有：腾讯云计算（北京）有限责任公司

setFilterStrength:	设置色彩滤镜的强度
setGreenScreenFile:	设置绿幕背景视频
setEyeScaleLevel:	设置大眼级别
setFaceSlimLevel:	设置瘦脸级别
setFaceVLevel:	设置 V 脸级别
setChinLevel:	设置下巴拉伸或收缩
setFaceShortLevel:	设置短脸级别
setFaceNarrowLevel:	设置窄脸级别
setNoseSlimLevel:	设置瘦鼻级别
setEyeLightenLevel:	设置亮眼级别
setToothWhitenLevel:	设置牙齿美白级别
setWrinkleRemoveLevel:	设置祛皱级别
setPounchRemoveLevel:	设置祛眼袋级别
setSmileLinesRemoveLevel:	设置法令纹去除级别
setForeheadLevel:	设置发际线调整级别
setEyeDistanceLevel:	设置眼距
setEyeAngleLevel:	设置眼角调整级别
setMouthShapeLevel:	设置嘴型调整级别
setNoseWingLevel:	设置鼻翼调整级别
setNosePositionLevel:	设置鼻子位置
setLipsThicknessLevel:	设置嘴唇厚度
setFaceBeautyLevel:	设置脸型
setMotionTmpl:inDir:	选择 AI 动效挂件
setMotionMute:	是否在动效素材播放时静音

音视频设备相关的类型定义

函数列表	描述
onDeviceChanged:type:state:	本地设备的通断状态发生变化（仅适用于桌面系统）

设备操作接口

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera:	切换前置或后置摄像头（仅适用于移动端）
isCameraZoomSupported	查询当前摄像头是否支持缩放（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio:	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus:	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition:	设置摄像头的对焦位置（仅适用于移动端）
isCameraTorchSupported	查询是否支持开启闪光灯（仅适用于移动端）
enableCameraTorch:	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）
setAudioRoute:	设置音频路由（仅适用于移动端）
setExposureCompensation:	设置摄像头的曝光参数，取值范围从-1到1
getDevicesList:	获取设备列表（仅适用于桌面端）
setCurrentDevice:deviceId:	设置当前要使用的设备（仅适用于桌面端）
getCurrentDevice:	获取当前正在使用的设备（仅适用于桌面端）
setCurrentDeviceVolume:deviceType:	设置当前设备的音量（仅适用于桌面端）
getCurrentDeviceVolume:	获取当前设备的音量（仅适用于桌面端）
setCurrentDeviceMute:deviceType:	设置当前设备的静音状态（仅适用于桌面端）

版权所有：腾讯云计算（北京）有限责任公司

getCurrentDeviceMute:	获取当前设备的静音状态（仅适用于桌面端）
enableFollowingDefaultAudioDevice:enable:	设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）
startCameraDeviceTest:	开始摄像头测试（仅适用于桌面端）
stopCameraDeviceTest	结束摄像头测试（仅适用于桌面端）
startMicDeviceTest:	开始麦克风测试（仅适用于桌面端）
startMicDeviceTest:playback:	开始麦克风测试（仅适用于桌面端）
stopMicDeviceTest	结束麦克风测试（仅适用于桌面端）
startSpeakerDeviceTest:	开始扬声器测试（仅适用于桌面端）
stopSpeakerDeviceTest	结束扬声器测试（仅适用于桌面端）
setObserver:	设备热插拔回调（仅适用于 Mac 系统）
setCameraCapturerParam:	设置摄像头采集偏好

弃用接口

函数列表	描述
setSystemVolumeType:	设置系统音量类型（仅适用于移动端）

废弃接口

函数列表	描述
destroySharedIntance	销毁 TRTCCloud 实例（单例模式）
delegate	设置 TRTC 事件回调
setBeautyStyle:beautyLevel:whitenessLevel:ruddinessLevel:	设置美颜、美白以及红润效果级别
setEyeScaleLevel:	设置大眼级别
setFaceScaleLevel:	设置瘦脸级别

setFaceVLevel:	设置 V 脸级别
setChinLevel:	设置下巴拉伸或收缩幅度
setFaceShortLevel:	设置短脸级别
setNoseSlimLevel:	设置瘦鼻级别
selectMotionTmpl:	设置动效贴纸
setMotionMute:	设置动效静音
setFilter:	设置色彩滤镜效果
setFilterConcentration:	设置色彩滤镜浓度
setGreenScreenFile:	设置绿幕背景视频
setReverbType:	设置混响效果
setVoiceChangerType:	设置变声类型
enableAudioEarMonitoring:	开启（或关闭）耳返
enableAudioVolumeEvaluation:	启用音量大小提示
enableAudioVolumeEvaluation:enable_vad:	启用音量大小提示
switchCamera	切换摄像头
isCameraZoomSupported	查询当前摄像头是否支持缩放
setZoom:	设置摄像头缩放倍数（焦距）
isCameraTorchSupported	查询是否支持开关闪光灯
enbaleTorch:	开关/关闭闪光灯
isCameraFocusPositionInPreviewSupported	查询摄像头是否支持设置焦点
setFocusPosition:	设置摄像头焦点坐标位置
isCameraAutoFocusFaceModeSupported	查询是否支持自动识别人脸位置
enableAutoFaceFoucs:	开启/关闭人脸跟踪对焦
setSystemVolumeType:	设置系统音量类型
snapshotVideo:type:	视频截图

版权所有：腾讯云计算（北京）有限责任公司

startScreenCaptureByReplaykit:appGroup:	开始全系统的屏幕分享（iOS）
startLocalAudio	设置音频质量
startRemoteView:view:	开始显示远端视频画面
stopRemoteView:	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode:	设置本地图像的渲染模式
setLocalViewRotation:	设置本地图像的顺时针旋转角度
setLocalViewMirror:	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode:mode:	设置远端图像的渲染模式
setRemoteViewRotation:rotation:	设置远端图像的顺时针旋转角度
startRemoteSubStreamView:view:	开始显示远端用户的辅路画面
stopRemoteSubStreamView:	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode:mode:	设置辅路画面的填充模式
setRemoteSubStreamViewRotation:rotation:	设置辅路画面的顺时针旋转角度
setAudioQuality:	设置音频质量
setPriorRemoteVideoStreamType:	设定优先观看大画面还是小画面
setMicVolumeOnMixing:	设置麦克风音量大小
playBGM:	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration:	获取背景音乐总时长（单位：毫秒）
setBGMPosition:	设置背景音乐的播放进度
setBGMVolume:	设置背景音乐的音量大小
setBGMPlayoutVolume:	设置背景音乐的本地播放音量
setBGMPublishVolume:	设置背景音乐的远端播放音量

版权所有：腾讯云计算（北京）有限责任公司

playAudioEffect:	播放音效
setAudioEffectVolume:volume:	设置音效音量
stopAudioEffect:	停止播放音效
stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume:	设置所有音效音量
pauseAudioEffect:	暂停音效
resumeAudioEffect:	暂停音效
enableCustomVideoCapture:	启用视频自定义采集模式
sendCustomVideoData:	投送自己采集的视频数据
muteLocalVideo:	暂停/恢复发布本地的视频流
muteRemoteVideoStream:mute:	暂停 / 恢复订阅远端用户的视频流
startSpeedTest:userId:userSig:	开始进行网络测速（进入房间前使用）
startScreenCapture:	启动屏幕分享
getCameraDevicesList	获取摄像头设备列表
setCurrentCameraDevice:	选定当前要使用的摄像头
getCurrentCameraDevice	获取当前使用的摄像头
getMicDevicesList	获取麦克风设备列表
getCurrentMicDevice	获取当前的麦克风设备
setCurrentMicDevice:	选定当前使用的麦克风
getCurrentMicDeviceVolume	获取当前麦克风的设备音量
setCurrentMicDeviceVolume:	设置当前麦克风的设备音量
setCurrentMicDeviceMute:	设置系统当前麦克风设备的静音状态
getCurrentMicDeviceMute	获取系统当前麦克风设备是否静音
getSpeakerDevicesList	获取扬声器设备列表
getCurrentSpeakerDevice	获取当前的扬声器设备

版权所有：腾讯云计算（北京）有限责任公司

setCurrentSpeakerDevice:	设置要使用的扬声器
getCurrentSpeakerDeviceVolume	获取当前扬声器的设备音量
setCurrentSpeakerDeviceVolume:	设置当前扬声器的设备音量
getCurrentSpeakerDeviceMute	获取系统当前扬声器设备是否静音
setCurrentSpeakerDeviceMute:	设置系统当前扬声器设备的静音状态
startCameraDeviceTestInView:	开始进行摄像头测试
stopCameraDeviceTest	停止进行摄像头测试
startMicDeviceTest:	开始进行麦克风测试
stopMicDeviceTest	开始进行麦克风测试
startSpeakerDeviceTest:	开始进行扬声器测试
stopSpeakerDeviceTest	停止进行扬声器测试
startScreenCaptureInApp:	开始应用内的屏幕分享（iOS）
setVideoEncoderRotation:	设置视频编码器输出的画面方向
setVideoEncoderMirror:	设置编码器输出的画面镜像模式
setGSensorMode:	设置重力感应的适配模式

版权所有：腾讯云计算（北京）有限责任公司

TRTCCloud

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTCCloud @ TXLiteAVSDK
Function: 腾讯云 TRTC 主功能接口
Version: 11.9

TRTCCloud

TRTCCloud

函数列表	描述
sharedInstance	创建 TRTCCloud 实例（单例模式）
destroySharedInstance	销毁 TRTCCloud 实例（单例模式）
addDelegate:	添加 TRTC 事件回调
removeDelegate:	移除 TRTC 事件回调
delegateQueue	设置驱动 TRTCCloudDelegate 事件回调的队列
enterRoom:appScene:	进入房间
exitRoom	离开房间
switchRole:	切换角色
switchRole:privateMapKey:	切换角色（支持设置权限位）
switchRoom:	切换房间
connectOtherRoom:	请求跨房通话
disconnectOtherRoom	退出跨房通话
setDefaultStreamRecvMode:video:	设置订阅模式（需要在进入房前设置才能生效）

createSubCloud	创建子房间实例（用于多房间并发观看）
destroySubCloud:	销毁子房间实例
updateOtherRoomForwardMode:	更改跨房主播在本房间的上行能力
startPublishing:type:	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream:	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig:	设置云端混流的排版布局和转码参数
startPublishMediaStream:encoderParam:mixingConfig:	开始发布媒体流
updatePublishMediaStream:publishTarget:encoderParam:mixingConfig:	更新发布媒体流
stopPublishMediaStream:	停止发布媒体流
startLocalPreview:view:	开启本地摄像头的预览画面（移动端）
startLocalPreview:	开启本地摄像头的预览画面（桌面端）
updateLocalView:	更新本地摄像头的预览画面
stopLocalPreview	停止摄像头预览
muteLocalVideo:mute:	暂停/恢复发布本地的视频流
setVideoMuteImage:fps:	设置本地画面被暂停期间的替代图片
startRemoteView:streamType:view:	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView:streamType:forUser:	更新远端用户的视频渲染控件
stopRemoteView:streamType:	停止订阅远端用户的视频流，并释

版权所有：腾讯云计算（北京）有限责任公司

	放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源
muteRemoteVideoStream:streamType:mute:	暂停/恢复订阅远端用户的视频流
muteAllRemoteVideoStreams:	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam:	设置视频编码器的编码参数
setNetworkQosParam:	设置网络质量控制的相关参数
setLocalRenderParams:	设置本地画面的渲染参数
setRemoteRenderParams:streamType:params:	设置远端画面的渲染模式
enableEncSmallVideoStream:withQuality:	开启大小画面双路编码模式
setRemoteVideoStreamType:type:	切换指定远端用户的大小画面
snapshotVideo:type:sourceType:	视频画面截图
setPerspectiveCorrectionWithUser:srcPoints:dstPoints:	视频画面透视校正坐标设置
setGravitySensorAdaptiveMode:	设置重力感应的适配模式（11.7 及以上版本）
startLocalAudio:	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布
muteLocalAudio:	暂停/恢复发布本地的音频流
muteRemoteAudio:mute:	暂停/恢复播放远端的音频流
muteAllRemoteAudio:	暂停/恢复播放所有远端用户的音频流
setAudioRoute:	设置音频路由
setRemoteAudioVolume:volume:	设定某一个远端用户的声音播放音量
setAudioCaptureVolume:	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量

版权所有：腾讯云计算（北京）有限责任公司

setAudioPlayoutVolume:	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
enableAudioVolumeEvaluation:withParams:	启用音量大小提示
startAudioRecording:	开始录音
stopAudioRecording	停止录音
startLocalRecording:	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams:	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect:	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition:	设置 3D 音效中远端用户坐标信息
set3DSpatialReceivingRange:range:	设置指定用户所发出声音的可被接收范围
getDeviceManager	获取设备管理类（TXDeviceManager）
getBeautyManager	获取美颜管理类（TXBeautyManager）
setWatermark:streamType:rect:	添加水印
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集（iOS 端暂未支持）
stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）
setSystemAudioLoopbackVolume:	设置系统声音的采集音量
startScreenCaptureInApp:encParam:	开始应用内的屏幕分享（仅支持 iOS 13.0 及以上系统）
startScreenCaptureByReplaykit:encParam:appGroup:	开始全系统的屏幕分享（仅支持 iOS 11.0 及以上系统）

版权所有：腾讯云计算（北京）有限责任公司

startScreenCapture:streamType:encParam:	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
getScreenCaptureSourcesWithThumbnailSize:iconSize:	枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）
selectScreenCaptureTarget:rect:capturesCursor:highlight:	选取要分享的屏幕或窗口（该接口仅支持 Mac OS 系统）
setSubStreamEncoderParam:	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）
setSubStreamMixVolume:	设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）
addExcludedShareWindow:	将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）
removeExcludedShareWindow:	将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
removeAllExcludedShareWindows	将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
addIncludedShareWindow:	将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）
removeIncludedShareWindow:	将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
removeAllIncludedShareWindows	将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
enableCustomVideoCapture:enable:	启用/关闭视频自定义采集模式
sendCustomVideoData:frame:	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture:	启用音频自定义采集模式
sendCustomAudioData:	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame:playout:	启用/关闭自定义音轨

版权所有：腾讯云计算（北京）有限责任公司

mixExternalAudioFrame:	向 SDK 混入自定义音轨
setMixExternalAudioVolume:playoutVolume:	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳
setLocalVideoProcessDelegete:pixelFormat:bufferType:	设置第三方美颜的视频数据回调
setLocalVideoRenderDelegate:pixelFormat:bufferType:	设置本地视频自定义渲染回调
setRemoteVideoRenderDelegate:delegate:pixelFormat:bufferType:	设置远端视频自定义渲染回调
setAudioFrameDelegate:	设置音频数据自定义回调
setCapturedAudioFrameDelegateFormat:	设置本地麦克风采集出的音频帧回调格式
setLocalProcessedAudioFrameDelegateFormat:	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameDelegateFormat:	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering:	开启音频自定义播放
getCustomAudioRenderingFrame:	获取可播放的音频数据
sendCustomCmdMsg:data:reliable:ordered:	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg:repeatCount:	使用 SEI 通道发送自定义消息给房间内所有用户
startSpeedTest:	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速
getSDKVersion	获取 SDK 版本信息
setLogLevel:	设置 Log 输出级别
setConsoleEnabled:	启用/禁用控制台日志打印
setLogCompressEnabled:	启用/禁用日志的本地压缩
setLogDirPath:	设置本地日志的保存路径

版权所有：腾讯云计算（北京）有限责任公司

setLogDelegate:	设置日志回调
showDebugView:	显示仪表盘
setDebugViewMargin:margin:	设置仪表盘的边距
callExperimentalAPI:	调用实验性接口
enablePayloadPrivateEncryption:params:	开启或关闭媒体流私有加密

sharedInstance

sharedInstance

创建 TRTCCloud 实例（单例模式）

参数	描述
context	仅适用于 Android 平台，SDK 内部会将其转化为 Android 平台的 ApplicationContext 用于调用 Android System API。 如果传入的 context 参数为空，SDK 内部会自动获取当前进程的 ApplicationContext。

注意

1. 如果您使用 delete ITRTCCloud* 会导致编译错误，请使用 destroyTRTCCloud 释放对象指针。
2. 在 Windows、Mac 和 iOS 平台上，请调用 getTRTCShareInstance() 接口。
3. 在 Android 平台上，请调用 getTRTCShareInstance(void *context) 接口。

destroySharedInstance

destroySharedInstance

销毁 TRTCCloud 实例（单例模式）

addDelegate:

addDelegate:

- (void)addDelegate:	(id<TRTCCloudDelegate>)delegate

添加 TRTC 事件回调

您可以通过 TRTCCloudDelegate 获得来自 SDK 的各类事件通知（比如：错误码，警告码，音视频状态参数等）。

removeDelegate:

removeDelegate:

- (void)removeDelegate:	(id<TRTCCloudDelegate>)delegate

移除 TRTC 事件回调

delegateQueue

delegateQueue

设置驱动 TRTCCloudDelegate 事件回调的队列

如果您不指定自己的 delegateQueue，SDK 默认会采用 MainQueue 作为驱动 TRTCCloudDelegate 事件回调的队列。
也就是当您不设置 delegateQueue 属性时，TRTCCloudDelegate 中的所有回调函数都是由 MainQueue 来驱动的。
注意
如果您指定了自己的 delegateQueue，请不要在 TRTCCloudDelegate 回调函数中操作 UI，否则会引发线程安全问题。

enterRoom:appScene:

enterRoom:appScene:

- (void)enterRoom:	(TRTCParams *)param
appScene:	(TRTCAppScene)scene

进入房间

TRTC 的所有用户都需要进入房间才能“发布”或“订阅”音视频流，“发布”是指将自己的音频和视频推送到云端，“订阅”是指从云端拉取房间里其他用户的音视频流。
调用该接口时，您需要指定您的应用场景 TRTCAppScene 以获取最佳的音视频传输体验，这些场景可以分成两大类：

实时通话：

包括 TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall 两个可选项，分别是视频通话和语音通话，该模式适合1对1 的音视频通话，或者参会人数在 300 人以内的在线会议。

在线直播：
包括 TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom 两个可选项，分别是视频直播和语音直播，该模式适合十万人以内的在线直播场景，但需要您在接下来介绍的 TRTCParams 参数中指定 **角色(role)** 这个字段，也就是将房间中的用户区分为 **主播** (TRTCRoleAnchor) 和 **观众** (TRTCRoleAudience) 两种不同的角色。

调用该接口后，您会收到来自 TRTCCloudDelegate 中的 onEnterRoom(result) 回调：
如果进房成功，参数 result 会是一个正数（result > 0），表示从函数调用到进入房间所花费的时间，单位是毫秒（ms）。
如果进房失败，参数 result 会是一个负数（result < 0），表示进房失败的**错误码**。

参数	描述
param	进房参数，用于指定用户的身份、角色和安全票据等信息，详情请参见 TRTCParams 。
scene	应用场景，用于指定您的业务场景，同一个房间内的所有用户需要设定相同的 TRTCAppScene。

注意
1. 同一个房间内的所有用户需要设定相同的 scene。不同的 scene 会导致偶现的异常问题。
2. 当您指定参数 scene 为 TRTCAppSceneLIVE 或 TRTCAppSceneVoiceChatRoom 时，您必须通过 TRTCParams 中的 “role” 字段为当前用户设定他/她在房间中的角色。
3. 请您尽量保证 enterRoom 与 exitRoom 前后配对使用，即保证”先退出前一个房间再进入下一个房间”，否则会导致很多异常问题。

exitRoom

exitRoom

离开房间

调用该接口会让用户离开自己所在的音视频房间，并释放摄像头、麦克风、扬声器等设备资源。
等资源释放完毕之后，SDK 会通过 TRTCCloudDelegate 中的 onExitRoom 回调向您通知。

如果您要再次调用 enterRoom 或者切换到其他的供应商的 SDK，建议等待 onExitRoom 回调到来之后再执行之后的操作，以避免摄像头或麦克风被占用的问题。

switchRole:

switchRole:

-(void)switchRole:	(TRTCRoleType)role

切换角色

调用本接口可以实现用户在 主播 和 观众 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了 **只有主播才能发布自己的音视频** 的规则。因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成 **主播**。

您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

参数	描述
role	角色，默认为 **主播**。 TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。 TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成 **主播**，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTCAppSceneLIVE）和语音聊天室（TRTCAppSceneVoiceChatRoom）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，请不要调用这个接口。

switchRole:privateMapKey:

switchRole:privateMapKey:

-(void)switchRole:	(TRTCRoleType)role
privateMapKey:	(NSString*)privateMapKey

切换角色（支持设置权限位）

调用本接口可以实现用户在 主播 和 观众 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了“只有主播才能发布自己的音视频”的规则。
因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成“主播”。
您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

参数	描述
privateMapKey	用于权限控制的权限票据，当您希望某个房间只能让特定的 userId 进入或者上行视频时，需要使用 privateMapKey 进行权限保护。 仅建议有高级别安全需求的客户使用，更多详情请参见 开启高级权限控制。
role	角色，默认为“主播”： TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。 TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成“主播”，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTCAppSceneLIVE）和语音聊天室（TRTCAppSceneVoiceChatRoom）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，请不要调用这个接口。

switchRoom:

switchRoom:

- (void)switchRoom:	(TRTCSwitchRoomConfig *)config

切换房间

使用该接口可以让用户可以快速从一个房间切换到另一个房间。
如果用户的身份是“观众”，该接口的调用效果等同于 exitRoom(当前房间) + enterRoom（新的房间）。
如果用户的身份是“主播”，该接口在切换房间的同时还会保持自己的音视频发布状态，因此在房间切换过程中，摄像头的预览和声音的采集都不会中断。

该接口适用于在线教育场景中，监课老师在多个房间中进行快速切换的场景。在该场景下使用 switchRoom 可以获得比 exitRoom+enterRoom 更好的流畅性和更少的代码量。
接口调用结果会通过 TRTCCloudDelegate 中的 onSwitchRoom(errCode, errMsg) 回调。

参数	描述
config	房间参数，详情请参见 TRTCSwitchRoomConfig。

注意

由于对老版本 SDK 兼容的需求，参数 config 中同时包含 roomId 与 strRoomId 两个参数，这两个参数的填写格外讲究，请注意如下事项：

1. 若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，将优先选用 roomId。
2. 所有房间需要同时使用 strRoomId 或同时使用 roomId，不可混用，否则将会出现很多预期之外的 bug。

connectOtherRoom:

connectOtherRoom:

- (void)connectOtherRoom:	(NSString *)param

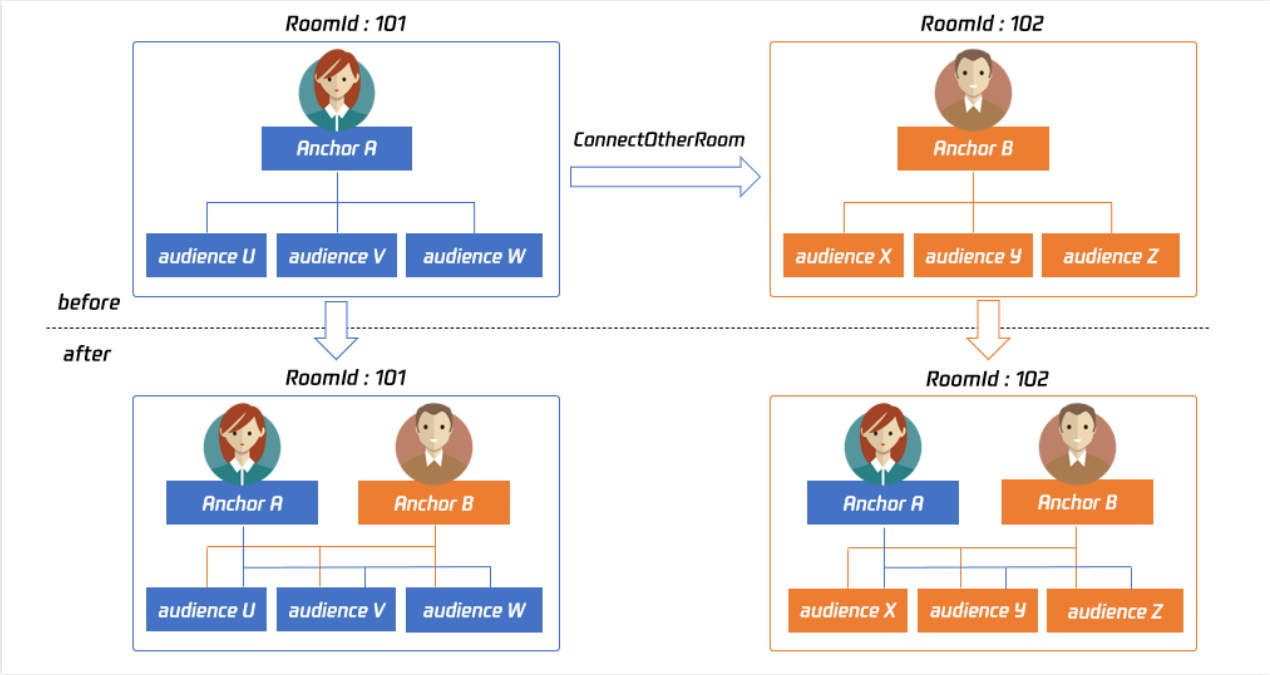
请求跨房通话

默认情况下，只有同一个房间中的用户之间可以进行音视频通话，不同的房间之间的音视频流是相互隔离的。
但您可以通过调用该接口，将另一个房间中某个主播音视频流发布到自己所在的房间中，与此同时，该接口也会将自己的音视频流发布到目标主播的房间中。

也就是说，您可以使用该接口让身处两个不同房间中的主播进行跨房间的音视频流分享，从而让每个房间中的观众都能观看到这两个主播的音视频。该功能可以用来实现主播之间的 PK 功能。

跨房通话的请求结果会通过 TRTCCloudDelegate 中的 onConnectOtherRoom 回调通知给您。

例如：当房间“101”中的主播 A 通过 connectOtherRoom() 跟房间“102”中的主播 B 建立跨房通话后，
房间“101”中的用户都会收到主播 B 的 onRemoteUserEnterRoom(B) 和 onUserVideoAvailable(B,YES) 这两个事件回调，即房间“101”中的用户都可以订阅主播 B 的音视频。
房间“102”中的用户都会收到主播 A 的 onRemoteUserEnterRoom(A) 和 onUserVideoAvailable(A,YES) 这两个事件回调，即房间“102”中的用户都可以订阅主播 A 的音视频。

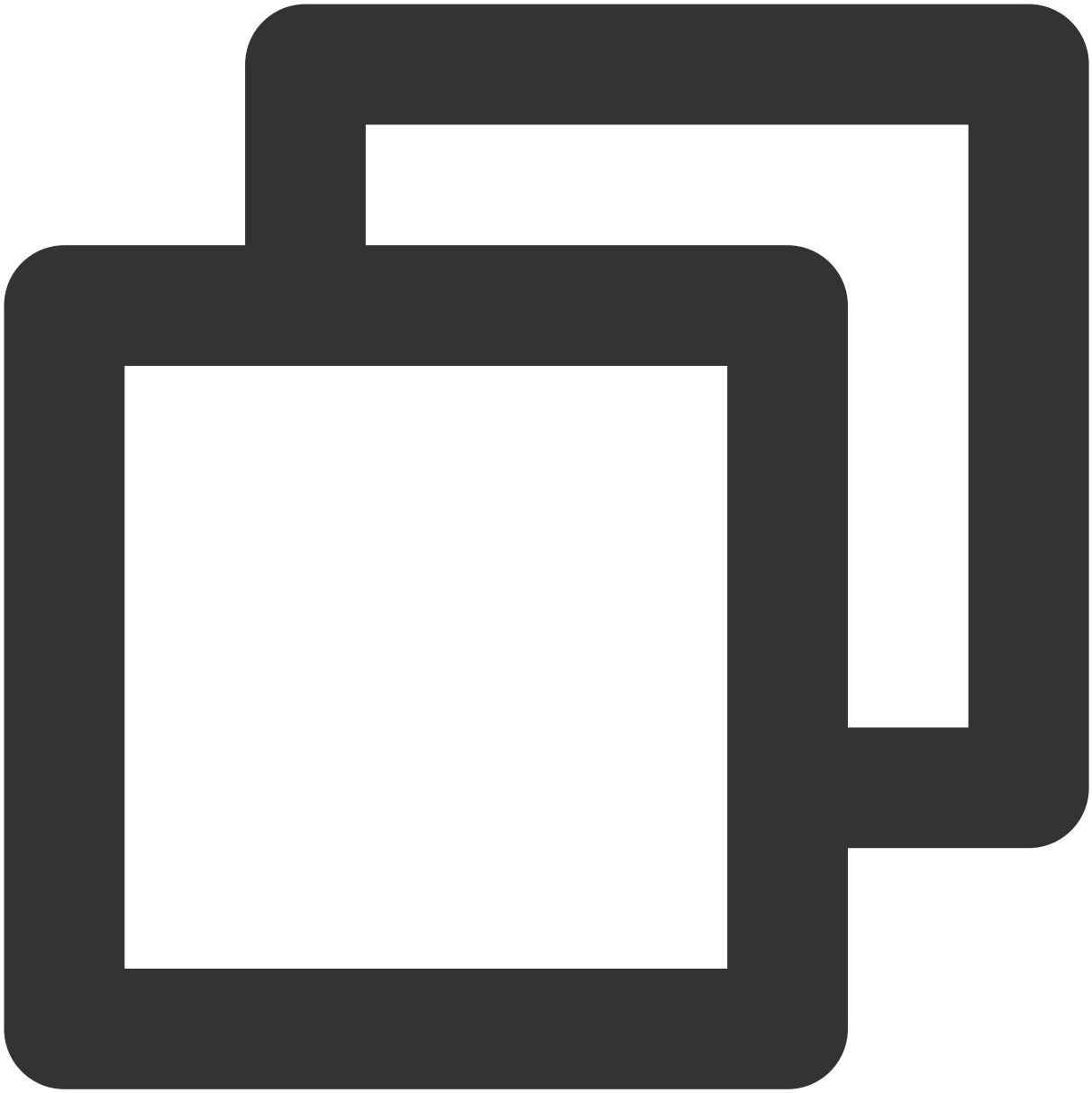


跨房通话的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数：

情况一：数字房间号

如果房间“101”中的主播 A 要跟房间“102”中的主播 B 连麦，那么主播 A 调用该接口时需要传入：{"roomId": 102, "userId": "userB"}

示例代码如下：

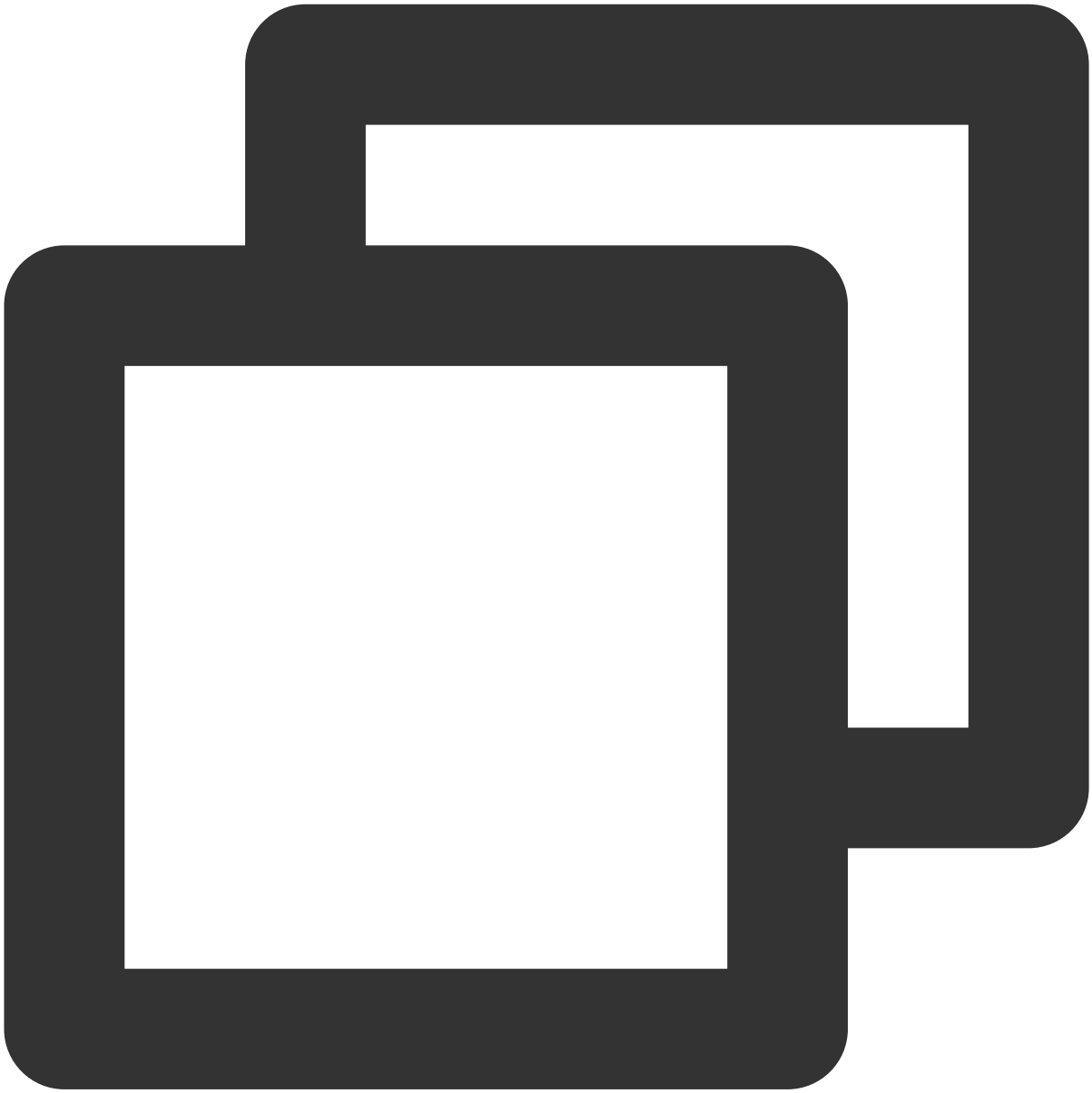


```
NSMutableDictionaryjsonDict = [[NSMutableDictionary alloc] init];
[jsonDict setObject:@(102) forKey:@"roomId"];
[jsonDict setObject:@"userB" forKey:@"userId"];
NSData* jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:NSJSO
NSString* jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8Str
[trtc connectOtherRoom:jsonString];
```

情况二：字符串房间号

如果您使用的是字符串房间号，务必请将 json 中的 “roomId” 替换成 “strRoomId”: {"strRoomId": "102", "userId": "userB"}

示例代码如下：



```
NSMutableDictionaryjsonDict = [[NSMutableDictionary alloc] init];
[jsonDict setObject:@"102" forKey:@"strRoomId"];
[jsonDict setObject:@"userB" forKey:@"userId"];
NSData* jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:NSJSO
NSString* jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8Str
[trtc connectOtherRoom:jsonString];
```

参数	描述
param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID。

disconnectOtherRoom

disconnectOtherRoom

退出跨房通话

退出结果会通过 TRTCCloudDelegate 中的 onDisconnectOtherRoom 回调通知给您。

setDefaultStreamRecvMode:video:

setDefaultStreamRecvMode:video:

- (void)setDefaultStreamRecvMode:	(BOOL)autoRecvAudio
video:	(BOOL)autoRecvVideo

设置订阅模式（需要在进入房前设置才能生效）

您可以通过该接口在“自动订阅”和“手动订阅”两种模式下进行切换：
自动订阅：默认模式，用户在进入房间后会立刻接收到该房间中的音视频流，音频会自动播放，视频会自动开始解码（依然需要您通过 startRemoteView 接口绑定渲染控件）。
手动订阅：在用户进入房间后，需要手动调用 startRemoteView 接口才能启动视频流的订阅和解码，需要手动调用 muteRemoteAudio (NO) 接口才能启动声音的播放。

在绝大多数场景下，用户进入房间后都会订阅房间中所有主播的音视频流，因此 TRTC 默认采用了自动订阅模式，以求得最佳的“秒开体验”。
如果您的应用场景中每个房间同时会有很多路音视频流在发布，而每个用户只想选择性地订阅其中的 1-2 路，则推荐使用“手动订阅”模式以节省流量费用。

参数	描述
autoRecvAudio	YES：自动订阅音频；NO：需手动调用 muteRemoteAudio(NO) 订阅音频。默认值：YES。
autoRecvVideo	YES：自动订阅视频；NO：需手动调用 startRemoteView 订阅视频。默认值：YES。

版权所有：腾讯云计算（北京）有限责任公司

注意

1. 需要在进入房间前调用该接口进行设置才能生效。
2. 在自动订阅模式下，如果用户在进入房间后没有调用 startRemoteView 订阅视频流，SDK 会自动停止订阅视频流，以便达到节省流量的目的。

createSubCloud

createSubCloud

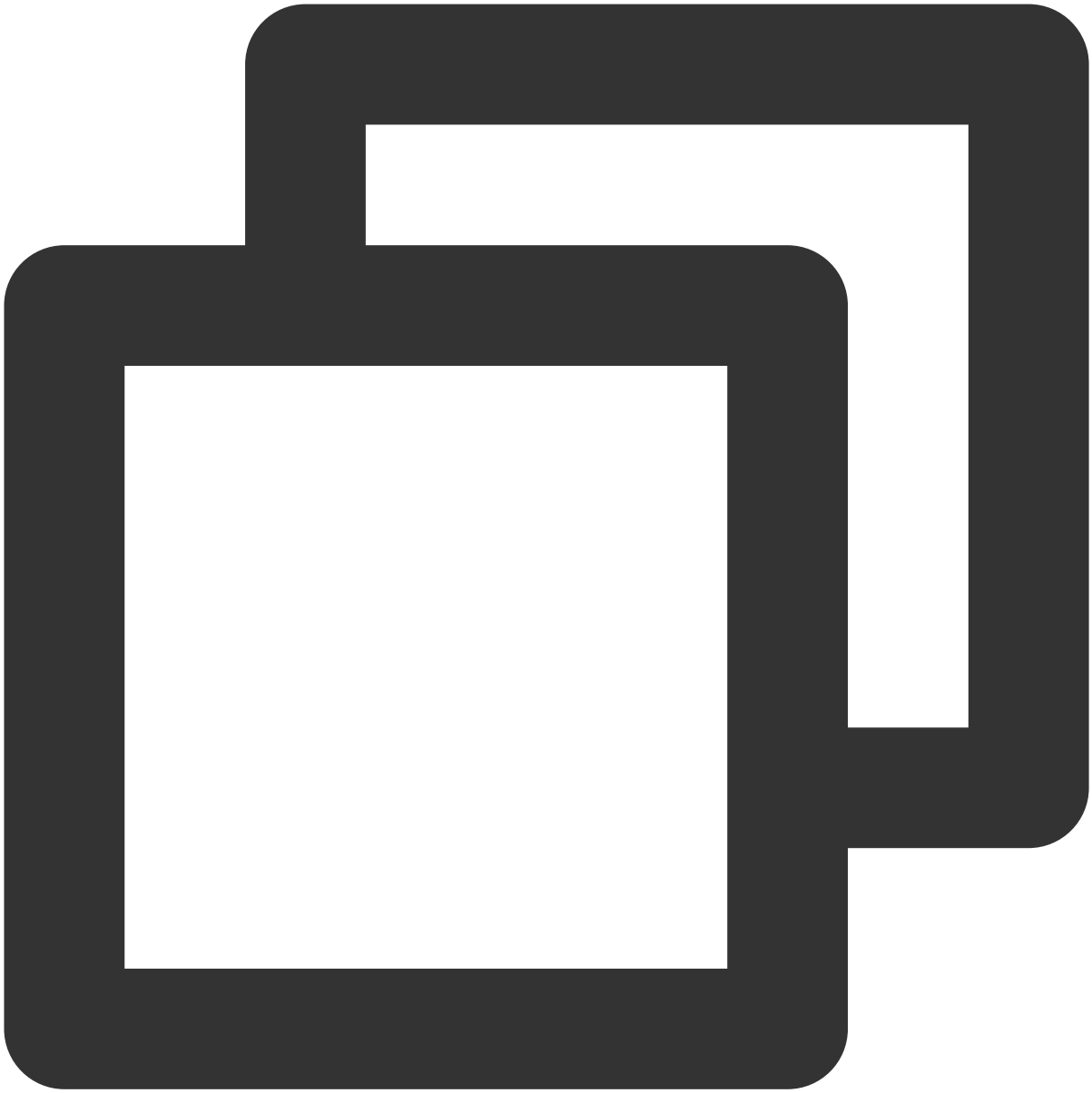
创建子房间实例（用于多房间并发观看）

TRTCCloud 一开始被设计成单例模式，限制了多房间并发观看的能力。
通过调用该接口，您可以创建出多个 TRTCCloud 实例，以便同时进入多个不同的房间观看音视频流。

但需要注意的是，您在多个 TRTCCloud 实例中发布音视频流的能力会受到一定限制。

该功能主要用于在线教育场景中一种被称为“超级小班课”的业务场景中，用于解决“每个 TRTC 的房间中最多只能有 50 人同时发布自己音视频流”的限制。

示例代码如下：



```
//In the small room that needs interaction, enter the room as an anchor and pus
TRTCCloud *mainCloud = [TRTCCloud sharedInstance];
TRTCParams *mainParams = [[TRTCParams alloc] init];
//Fill your params
mainParams.role = TRTCRoleAnchor;
[mainCloud enterRoom:mainParams appScene:TRTCAppSceneLIVE)];
//...
[mainCloud startLocalPreview:YES view:videoView];
[mainCloud startLocalAudio:TRTCAudioQualityDefault];

//In the large room that only needs to watch, enter the room as an audience and
```

```
TRTCCloud *subCloud = [mainCloud createSubCloud];
TRTCParams *subParams = [[TRTCParams alloc] init];
//Fill your params
subParams.role = TRTCRoleAudience;
[subCloud enterRoom:subParams appScene:TRTCAppSceneLIVE)];
//...
[subCloud startRemoteView:userId streamType:TRTCVideoStreamTypeBig view:videoVi
//...
//Exit from new room and release it.
[subCloud exitRoom];
[mainCloud destroySubCloud:subCloud];
```

注意

同一个用户，可以使用同一个 userId 进入多个不同 roomId 的房间。

两台不同的终端设备不可以同时使用同一个 userId 进入同一个 roomId 的房间。

您可以分别为不同实例分别设置 TRTCCloudDelegate 获取各自的事件通知。

同一个用户可以在多个 TRTCCloud 实例中推流，也可以调用子实例中与本地音视频相关的接口。但需要注意：

多实例的音频需要同时为麦克风采集或自定义采集，而且与音频设备相关的接口调用会以最后一次为准；

与摄像头相关的调用会以最后一次为准：startLocalPreview。

返回值说明：

子 TRTCCloud 实例

destroySubCloud:

destroySubCloud:

- (void)destroySubCloud:	(TRTCCloud *)subCloud

销毁子房间实例

参数	描述
subCloud	子房间实例

updateOtherRoomForwardMode:

updateOtherRoomForwardMode:

- (void)updateOtherRoomForwardMode:	(NSString *)param

更改跨房主播在本房间的上行能力

通常情况下，在调用 connectOtherRoom 接口与另一个房间的主播进行跨房通话后，本房间内的所有观众都将收到该主播发布的音视频流。
您可以通过调用该接口，限制跨房主播在本房间内的上行能力，禁止或允许跨房主播发布音频/主路视频/辅路视频，该行为会影响房间内的所有用户。
在禁用跨房主播某种上行能力后，本房间内所有用户将无法收到对应音视频流，且无法再订阅对应的音视频。

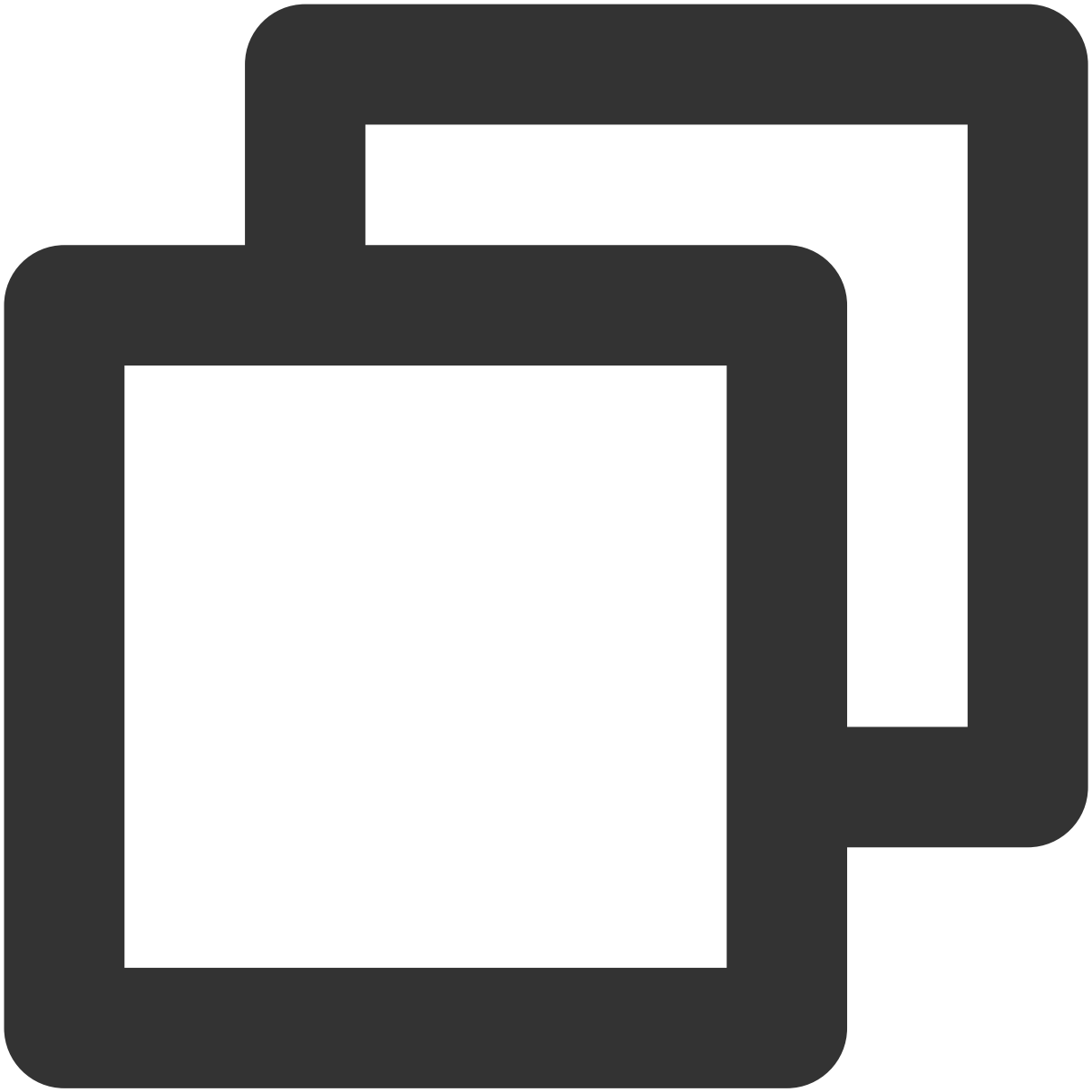
需要注意的是，该接口只能由进行跨房通话的主播调用，且通过该接口设置的限制会因为跨房通话的中断或对应主播退房重置。

该接口的调用结果会通过 TRTCCloudDelegate 中的 onUpdateOtherRoomForwardMode 回调通知给您。

例如：
房间“101”中有主播 A 与观众 B，房间“102”中有主播 C，主播 C 正常发布音视频流。主播 A 通过 connectOtherRoom() 跟主播 C 建立跨房通话。
此时，主播 A 与观众 B 都会收到主播 C 的 onRemoteUserEnterRoom(C) 和 onUserVideoAvailable(C,YES)、onUserAudioAvailable(C,YES) 这三个事件回调，可以订阅主播 C 的音视频。

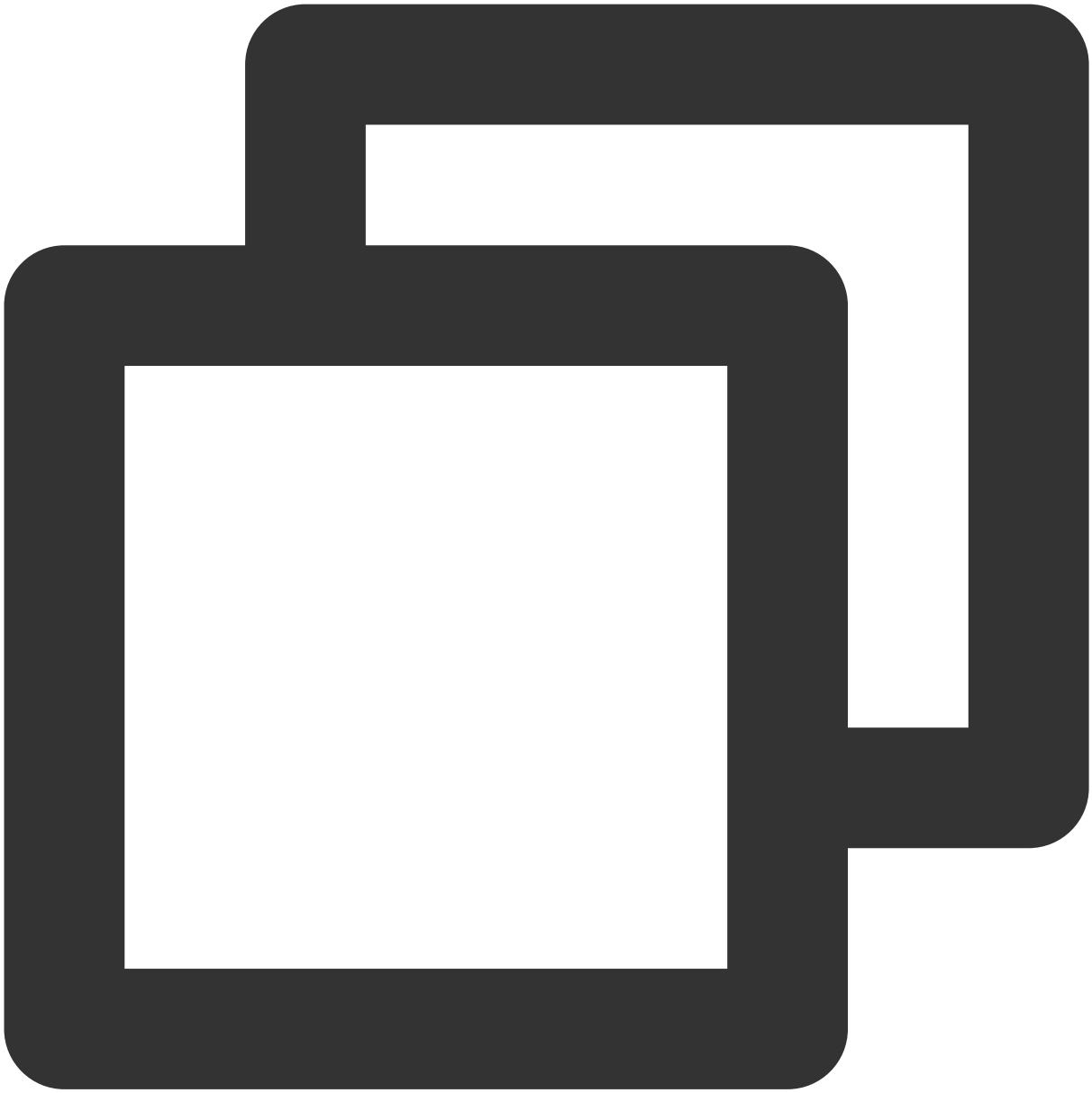
之后，主播 A 调用该接口禁用主播 C 在本房间发布音频的能力。
此后，主播 C 的音频流将无法发布至房间“101”，主播 A 与观众 B 都将收到 onUserAudioAvailable(C,NO) 事件回调，且无法再通过 muteRemoteAudio(C,NO) 订阅主播 C 的音频。
主播 C 的视频流将不受影响。房间“102”中的其他观众也不受影响，可以正常订阅主播 C 的音频。

该接口的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数，示例如下：
情况一：数字房间号



```
{
  "roomId":102,
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

情况二：字符串房间号



```
{
  "strRoomId":"102",
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

参数	描述

param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID，muteAudio、muteVideo、muteSubStream均为可选项，分表代表禁止或允许跨房主播发布音频/主路视频/辅路视频的能力。

startPublishing:type:

startPublishing:type:

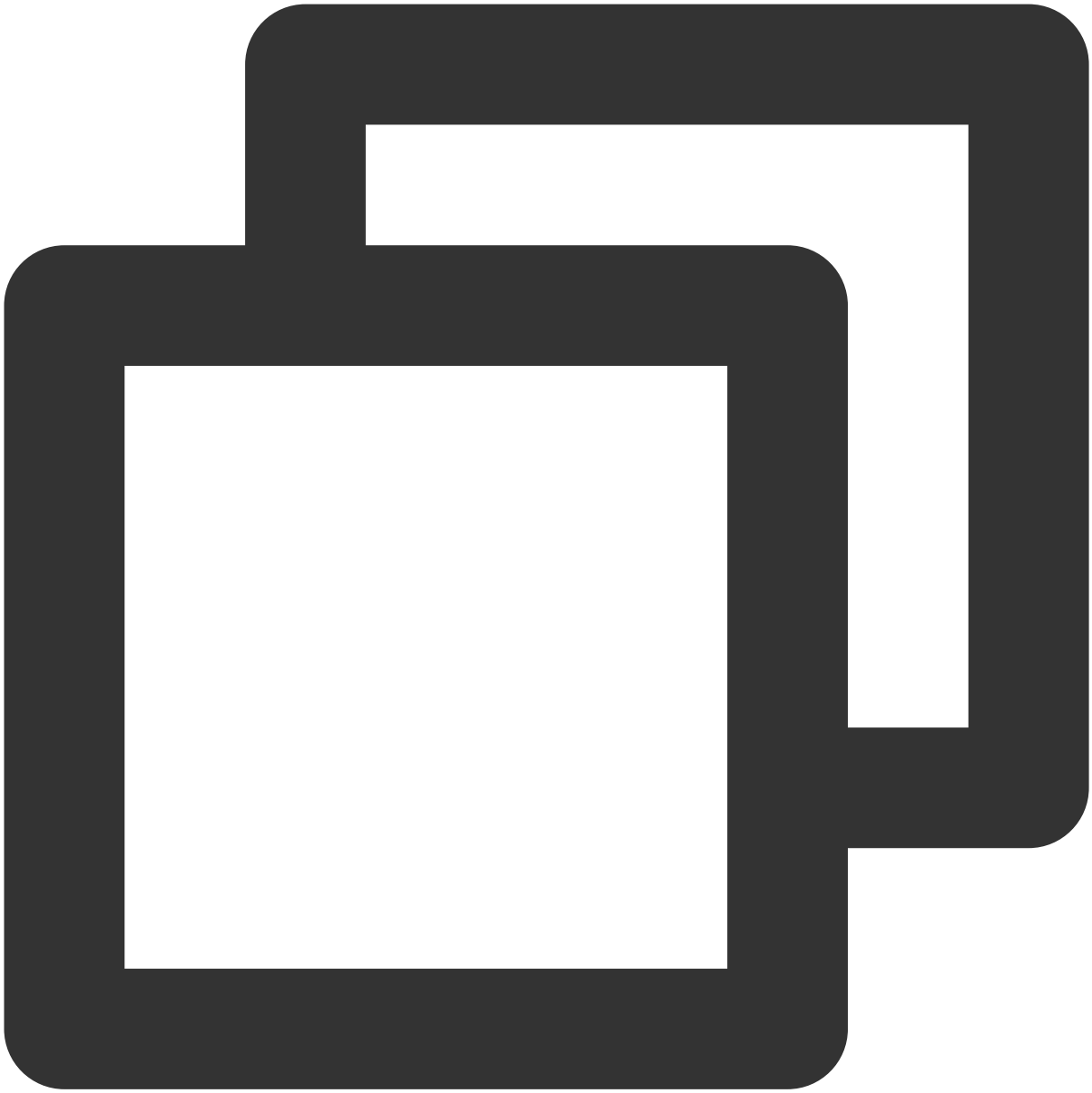
- (void)startPublishing:	(NSString *)streamId
type:	(TRTCVideoStreamType)streamType

开始向腾讯云直播 CDN 上发布音视频流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流旁路到直播 CDN 上。您可以通过参数 streamId 设定直播流的 StreamId，从而可以指定该用户的音视频流对应在直播 CDN 上的播放地址。

例如：您可以通过该接口将当前用户的直播流 ID 指定为 user_stream_001，那么该用户音视频流对应的 CDN 播放地址为：
“http://yourdomain/live/user_stream_001.flv”，其中 yourdomain 为您自己备案的播放域名，

您可以在直播控制台 配置您的播放域名，腾讯云不提供默认的播放域名。



```
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
[trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
[trtcCloud startLocalPreview:frontCamera view:localView];
[trtcCloud startLocalAudio];
[trtcCloud startPublishing: @"user_stream_001" type:TRTCVideoStreamTypeBig];
```

您也可以在设置 enterRoom 的参数 TRTCParams 时指定 streamId, 而且我们更推荐您采用这种方案。

参数	描述

streamId	自定义流 ID。
streamType	仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub。

注意

您需要提前在 实时音视频控制台 中的功能配置页面上开启“启用旁路推流”才能生效。

若您选择“指定流旁路”，则您可以通过该接口将对应音视频流推送到腾讯云 CDN 且指定为填写的流 ID。

若您选择“全局自动旁路”，则您可以通过该接口调整默认的流 ID。

stopPublishing

stopPublishing

停止向腾讯云直播 CDN 上发布音视频流

startPublishCDNStream:

startPublishCDNStream:

- (void)startPublishCDNStream:	(TRTCPublishCDNParam*)param

开始向非腾讯云 CDN 上发布音视频流

该接口跟 startPublishing 功能类似，不同之处在于，startPublishing 仅支持向腾讯云的 CDN 发布，而本接口支持向非腾讯云的直播 CDN 上转推音视频流。

参数	描述
param	CDN 转推参数，详情请参见 TRTCPublishCDNParam

注意

使用 startPublishing 接口向腾讯云的直播 CDN 上发布音视频流不会收取额外费用。

使用 startPublishCDNStream 接口向非腾讯云的直播 CDN 上发布音视频流，需要收取额外的转推带宽费用。

stopPublishCDNStream

stopPublishCDNStream

停止向非腾讯云 CDN 上发布音视频流

版权所有：腾讯云计算（北京）有限责任公司

setMixTranscodingConfig:

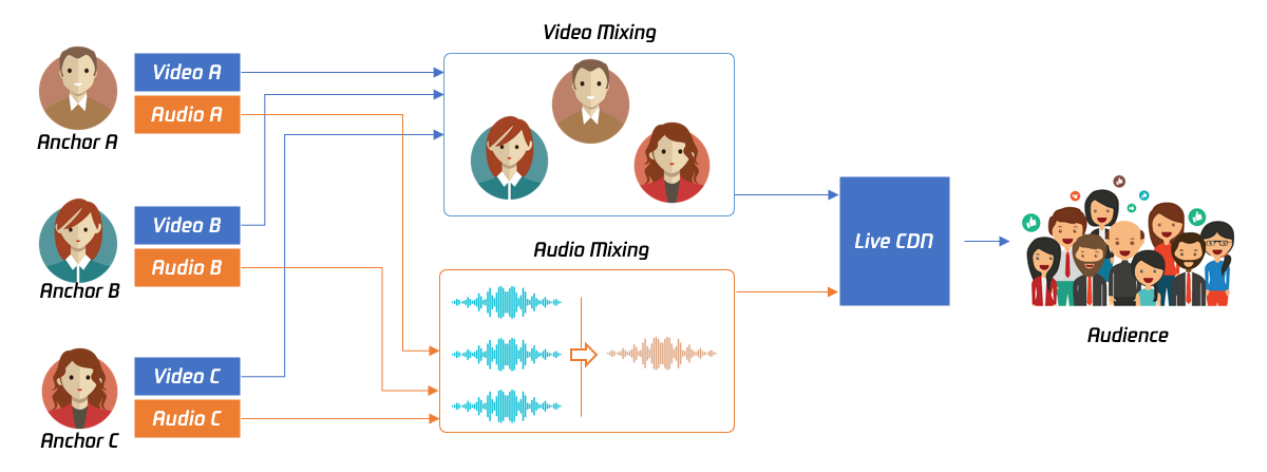
setMixTranscodingConfig:

- (void)setMixTranscodingConfig:	(nullable TRTCTranscodingConfig*)config

设置云端混流的排版布局和转码参数

在一个直播间中可能同时会有多个主播发布自己的音视频流，但对于直播 CDN 上的观众而言，只需要观看一条 HTTP-FLV 或 HLS 格式的视频流即可。
当您调用本接口函数时，SDK 会向腾讯云的 TRTC 混流服务器发送一条指令，混流服务器会将房间里的多路音视频流混合成一路。
您可以通过 TRTCTranscodingConfig 参数来调整每一路画面的排版布局，也可以设置混合后的音视频流的各项编码参数。

参考文档：云端混流转码。



参数	描述
config	如果 config 不为空，则开启云端混流，如果 config 为空则停止云端混流。详情请参见 TRTCTranscodingConfig 。

注意

关于云端混流的注意事项：
混流转码为收费功能，调用接口将产生云端混流转码费用，详见 云端混流转码计费说明 。
调用该接口的用户，如果没设定 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到当前用户所对应的音视频流上，即 A + B => A。

调用该接口的用户，如果设定了 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到您指定的 streamId 上，即 A + B => streamId。

请注意，若您还在房间中且不再需要混流，请务必再次调用本接口并将 config 设置为空以进行取消，不及时取消混流可能会引起不必要的计费损失。

请放心，当您退房时 TRTC 会自动取消混流状态。

startPublishMediaStream:encoderParam:mixingConfig:

startPublishMediaStream:encoderParam:mixingConfig:

- (void)startPublishMediaStream:	(TRTCPublishTarget*)target
encoderParam:	(nullable TRTCStreamEncoderParam*)param
mixingConfig:	(nullable TRTCStreamMixingConfig*)config

开始发布媒体流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流转推/转码到直播 CDN 或者回推到 TRTC 房间中您可以通过 TRTCPublishTarget 配置中的 TRTCPublishMode 指定具体的发布模式

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。

注意

1. SDK 会通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）。
2. 同一个任务（TRTCPublishMode 与 TRTCPublishCdnUrl 均相同）仅支持启动一次。若您后续需要更新或者停止该项任务，需要记录并使用返回的 taskId，通过 updatePublishMediaStream 或者 stopPublishMediaStream 来操作。
3. target 支持同时配置多个 CDN URL（最多同时 10 个）。若您的同一个转推/转码任务需要发布至多路 CDN，则仅需要在 target 中配置多个 CDN URL 即可。同一个转码任务即使有多个转推地址，对应的转码计费仍只收取一份。
4. 使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。一种推荐的方案是 URL 中使用 “sdkappid_roomid_userid_main” 作为区分标识，这种命名方式容易辨认且不会在您的多个应用中发生冲突。

版权所有：腾讯云计算（北京）有限责任公司

updatePublishMediaStream:publishTarget:encoderParam:mixingConfig:

updatePublishMediaStream:publishTarget:encoderParam:mixingConfig:

- (void)updatePublishMediaStream:	(NSString *)taskId
publishTarget:	(TRTCPublishTarget*)target
encoderParam:	(nullable TRTCStreamEncoderParam*)param
mixingConfig:	(nullable TRTCStreamMixingConfig*)config

更新发布媒体流

该接口会向 TRTC 服务器发送指令，更新通过 startPublishMediaStream 启动的媒体流

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 您可以通过本接口来更新发布的 CDN URL（支持增删，最多同时 10 个），但您使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。
2. 您可以通过 taskId 来更新调整转推/转码任务。例如在 pk 业务中，您可以先通过 startPublishMediaStream 发起转推，接着在主播发起 pk 时，通过 taskId 和本接口将转推更新为转码任务。此时，CDN 播放将连续并且不会发生断流（您需要保持媒体流编码输出参数 param 一致）。
3. 同一个任务不支持纯音频、音视频、纯视频之间的切换。

stopPublishMediaStream:

stopPublishMediaStream:

- (void)stopPublishMediaStream:	(NSString *)taskId

停止发布媒体流

该接口会向 TRTC 服务器发送指令，停止通过 startPublishMediaStream 启动的媒体流

参数	描述
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 若您的业务后台并没有保存该 taskId，在您的主播异常退房重进后，如果您需要重新获取 taskId，您可以再次调用 startPublishMediaStream 启动任务。此时 TRTC 后台会返回任务启动失败，同时带给您上一次启动的 taskId
2. 若 taskId 填空字符串，将会停止该用户所有通过 startPublishMediaStream 启动的媒体流，如果您只启动了一个媒体流或者想停止所有通过您启动的媒体流，推荐使用这种方式。

startLocalPreview:view:

startLocalPreview:view:

- (void)startLocalPreview:	(BOOL)frontCamera
view:	(nullable TXView *)view

开启本地摄像头的预览画面（移动端）

在 enterRoom 之前调用此函数，SDK 只会开启摄像头，并一直等到您调用 enterRoom 之后才开始推流。
在 enterRoom 之后调用此函数，SDK 会开启摄像头并自动开始视频推流。当开始渲染首帧摄像头画面时，您会收到 TRTCCloudDelegate 中的 onCameraDidReady 回调通知。

参数	描述
frontCamera	YES：前置摄像头；NO：后置摄像头。
view	承载视频画面的控件。

注意

如果希望开播前预览摄像头画面并通过 BeautyManager 调节美颜参数，您可以：
方案一：在调用 enterRoom 之前调用 startLocalPreview。
方案二：在调用 enterRoom 之后调用 startLocalPreview + muteLocalVideo(YES)。

startLocalPreview:

startLocalPreview:

- (void)startLocalPreview:	(nullable TXView *)view

开启本地摄像头的预览画面（桌面端）

在调用该接口之前，您可以先调用 setCurrentCameraDevice 选择使用 Mac 自带摄像头或外接摄像头。
在 enterRoom 之前调用此函数，SDK 只会开启摄像头，并一直等到您调用 enterRoom 之后才开始推流。
在 enterRoom 之后调用此函数，SDK 会开启摄像头并自动开始视频推流。
当开始渲染首帧摄像头画面时，您会收到 TRTCCloudDelegate 中的 onCameraDidReady 回调通知。

参数	描述
view	承载视频画面的控件。

注意

如果希望开播前预览摄像头画面并通过 BeautyManager 调节美颜参数，您可以：
方案一：在调用 enterRoom 之前调用 startLocalPreview。
方案二：在调用 enterRoom 之后调用 startLocalPreview + muteLocalVideo(YES)。

updateLocalView:

updateLocalView:

- (void)updateLocalView:	(nullable TXView *)view

更新本地摄像头的预览画面

stopLocalPreview

stopLocalPreview

停止摄像头预览

muteLocalVideo:mute:

muteLocalVideo:mute:

- (void)muteLocalVideo:	(TRTCVideoStreamType)streamType
mute:	(BOOL)mute

暂停/恢复发布本地的视频流

该接口可以暂停（或恢复）发布本地的视频画面，暂停之后，同一房间中的其他用户将无法继续看到自己画面。
该接口在指定 TRTCVideoStreamTypeBig 时等效于 start/stopLocalPreview 这两个接口，但具有更好的响应速度。
因为 start/stopLocalPreview 需要打开和关闭摄像头，而打开和关闭摄像头都是硬件设备相关的操作，非常耗时。
相比之下，muteLocalVideo 只需要在软件层面对数据流进行暂停或者放行即可，因此效率更高，也更适合需要频繁打开关闭的场景。

当暂停/恢复发布指定 TRTCVideoStreamTypeBig 后，同一房间中的其他用户将会收到 onUserVideoAvailable 回调通知。
当暂停/恢复发布指定 TRTCVideoStreamTypeSub 后，同一房间中的其他用户将会收到 onUserSubStreamAvailable 回调通知。

参数	描述
mute	YES：暂停；NO：恢复。
streamType	要暂停/恢复的视频流类型（仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub）。

setVideoMuteImage:fps:

setVideoMuteImage:fps:

- (void)setVideoMuteImage:	(nullable TXImage *)image
fps:	(NSInteger)fps

设置本地画面被暂停期间的替代图片

当您调用 muteLocalVideo(YES) 暂停本地画面时，您可以通过调用本接口设置一张替代图片，设置后，房间中的其他用户会看到这张替代图片，而不是黑屏画面。

参数	描述
fps	设置替代图片帧率，最小值为5，最大值为10，默认5。
image	设置替代图片，空值代表在 muteLocalVideo 之后不再发送视频流数据，默认值为空。

版权所有：腾讯云计算（北京）有限责任公司

startRemoteView:streamType:view:

startRemoteView:streamType:view:

- (void)startRemoteView:	(NSString *)userId
streamType:	(TRTCVideoStreamType)streamType
view:	(nullable TXView *)view

订阅远端用户的视频流，并绑定视频渲染控件

调用该接口可以让 SDK 拉取指定 userid 的视频流，并渲染到参数 view 指定的渲染控件上。您可以通过 setRemoteRenderParams 设置画面的显示模式。
如果您已经知道房间中有视频流的用户的 userid，可以直接调用 startRemoteView 订阅该用户的画面。
如果您不知道房间中有哪些用户在发布视频，您可以在 enterRoom 之后等待来自 onUserVideoAvailable 的通知。

调用本接口只是启动视频流的拉取，此时画面还需要加载和缓冲，当缓冲完毕后您会收到来自 onFirstVideoFrame 的通知。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall（需要远端用户通过 enableEncSmallVideoStream 开启双路编码后才有效果）。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。
view	用于承载视频画面的渲染控件。

注意

注意几点规则需要您关注：
1. SDK 支持同时观看某 userid 的大画面和辅路画面，或者同时观看某 userid 的小画面和辅路画面，但不支持同时观看大画面和小画面。
2. 只有当指定的 userid 通过 enableEncSmallVideoStream 开启双路编码后，才能观看该用户的小画面。
3. 当指定的 userid 的小画面不存在时，SDK 默认切换到该用户的大画面。

updateRemoteView:streamType:forUser:

updateRemoteView:streamType:forUser:

- (void)updateRemoteView:	(nullable TXView *)view
streamType:	(TRTCVideoStreamType)streamType
forUser:	(NSString *)userId

更新远端用户的视频渲染控件

该接口可用于更新远端视频画面的渲染控件，常被用于切换显示区域的交互场景中。

参数	描述
streamType	要设置预览窗口的流类型（仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub）。
userId	指定远端用户的 ID。
view	承载视频画面的控件。

stopRemoteView:streamType:

stopRemoteView:streamType:

- (void)stopRemoteView:	(NSString *)userId
streamType:	(TRTCVideoStreamType)streamType

停止订阅远端用户的视频流，并释放渲染控件

调用此接口会让 SDK 停止接收该用户的视频流，并释放该路视频流的解码和渲染资源。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。

stopAllRemoteView

stopAllRemoteView

停止订阅所有远端用户的视频流，并释放全部渲染资源

调用此接口会让 SDK 停止接收所有来自远端的视频流，并释放全部的解码和渲染资源。

注意

如果当前有正在显示的辅路画面（屏幕分享）也会一并被停止。

muteRemoteVideoStream:streamType:mute:

muteRemoteVideoStream:streamType:mute:

- (void)muteRemoteVideoStream:	(NSString*)userId
streamType:	(TRTCVideoStreamType)streamType
mute:	(BOOL)mute

暂停/恢复订阅远端用户的视频流

该接口仅暂停/恢复接收指定用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。
streamType	要暂停/恢复的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。

调用该接口暂停接收指定用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(NO) 或 muteAllRemoteVideoStreams(NO) 才能恢复。

muteAllRemoteVideoStreams:

muteAllRemoteVideoStreams:

- (void)muteAllRemoteVideoStreams:	(BOOL)mute

暂停/恢复订阅所有远端用户的视频流

该接口仅暂停/恢复接收所有用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。

调用该接口暂停接收所有用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(NO) 或 muteAllRemoteVideoStreams(NO) 才能恢复。

setVideoEncoderParam:

setVideoEncoderParam:

- (void)setVideoEncoderParam:	(TRTCVideoEncParam*)param

设置视频编码器的编码参数

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。

参数	描述
param	用于设置视频编码器的相关参数，详情请参见 TRTCVideoEncParam。

注意

从v11.5版本开始，编码输出分辨率会按照宽8高2字节对齐，并且是向下调整，eg:输入分辨率540x960，实际编码输出分辨率536x960。

setNetworkQosParam:

setNetworkQosParam:

- (void)setNetworkQosParam:	(TRTCNetworkQosParam*)param

设置网络质量控制的相关参数

该设置决定在差网络环境下的质量调控策略，如“画质优先”或“流畅优先”等策略。

参数	描述
param	用于设置网络质量控制的相关参数，详情请参见 TRTCNetworkQosParam。

setLocalRenderParams:

setLocalRenderParams:

- (void)setLocalRenderParams:	(TRTCRenderParams *)params

设置本地画面的渲染参数

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述
params	画面渲染参数，详情请参见 TRTCRenderParams。

setRemoteRenderParams:streamType:params:

setRemoteRenderParams:streamType:params:

- (void)setRemoteRenderParams:	(NSString *)userId
streamType:	(TRTCVideoStreamType)streamType
params:	(TRTCRenderParams *)params

设置远端画面的渲染模式

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述
params	画面渲染参数，详情请参见 TRTCRenderParams。
streamType	可以设置为主路画面（TRTCVideoStreamTypeBig）或辅路画面（TRTCVideoStreamTypeSub）。
userId	指定远端用户的 ID。

enableEncSmallVideoStream:withQuality:

enableEncSmallVideoStream:withQuality:

- (int)enableEncSmallVideoStream:	(BOOL)enable
withQuality:	(TRTCVideoEncParam*)smallVideoEncParam

版权所有：腾讯云计算（北京）有限责任公司

开启大小画面双路编码模式

开启双路编码模式后，当前用户的编码器会同时输出【高清大画面】和【低清小画面】两路视频流（但只有一路音频流）。
如此以来，房间中的其他用户就可以根据自身的网络情况或屏幕大小选择订阅【高清大画面】或是【低清小画面】。

参数	描述
enable	是否开启小画面编码，默认值：NO。
smallVideoEncParam	小流的视频参数。

注意

双路编码开启后，会消耗更多的 CPU 和 网络带宽，所以 Mac、Windows 或者高性能 Pad 可以考虑开启，不建议手机端开启。

返回值说明：

0：成功；-1：当前大画面已被设置为较低画质，开启双路编码已无必要。

setRemoteVideoStreamType:type:

setRemoteVideoStreamType:type:

- (void)setRemoteVideoStreamType:	(NSString*)userId
type:	(TRTCVideoStreamType)streamType

切换指定远端用户的大小画面

当房间中某个主播开启了双路编码之后，房间中其他用户通过 startRemoteView 订阅到的画面默认会是【高清大画面】。
您可以通过此接口选定希望订阅的画面是大画面还是小画面，该接口在 startRemoteView 之前和之后调用均可生效。

参数	描述
streamType	视频流类型，即选择看大画面还是小画面，默认为大画面。
userId	指定远端用户的 ID。

注意

此功能需要目标用户已经通过 enableEncSmallVideoStream 提前开启了双路编码模式，否则此调用无实际效果。

版权所有：腾讯云计算（北京）有限责任公司

snapshotVideo:type:sourceType:

snapshotVideo:type:sourceType:

- (void)snapshotVideo:	(nullable NSString *)userId
type:	(TRTCVideoStreamType)streamType
sourceType:	(TRTCSnapshotSourceType)sourceType

视频画面截图

您可以通过本接口截取本地的视频画面，远端用户的主路画面以及远端用户的辅路（屏幕分享）画面。

参数	描述
sourceType	画面来源，可选择截取视频流画面（TRTCSnapshotSourceTypeStream）、视频渲染画面（TRTCSnapshotSourceTypeView）或 采集画面（TRTCSnapshotSourceTypeCapture），采集画面截图更清晰。
streamType	视频流类型，可选择截取主路画面（TRTCVideoStreamTypeBig，常用于摄像头）或辅路画面（TRTCVideoStreamTypeSub，常用于屏幕分享）。
userId	用户 ID，如指定空置表示截取本地的视频画面。

注意

Windows 平台目前仅支持截取 TRTCSnapshotSourceTypeStream 来源的视频画面。

setPerspectiveCorrectionWithUser:srcPoints:dstPoints:

setPerspectiveCorrectionWithUser:srcPoints:dstPoints:

- (void)setPerspectiveCorrectionWithUser:	(nullable NSString *)userId
srcPoints:	(nullable NSArray *)srcPoints
dstPoints:	(nullable NSArray *)dstPoints

视频画面透视校正坐标设置

您可以通过本接口指定渲染画面透视校正的坐标区域。

参数	描述
dstPoints	期望校正到的目标坐标区域4个顶点坐标，需按照左上、左下、右上、右下的顺序传入，所

	有的坐标需要根据画面宽高做 [0,1] 区间的归一化；传 null 则停止透视校正。
srcPoints	原始画面坐标区域4个顶点坐标，需按照左上、左下、右上、右下的顺序传入，所有的坐标需要根据画面宽高做 [0,1] 区间的归一化; 传 null 则停止透视校正。
userId	用户 ID，如指定空值表示校正本地的视频画面。

setGravitySensorAdaptiveMode:

setGravitySensorAdaptiveMode:

- (void)setGravitySensorAdaptiveMode:	(TRTCGravitySensorAdaptiveMode) mode

设置重力感应的适配模式（11.7 及以上版本）

开启重力感应后，如果采集端的设备发生旋转，采集端和观众端的画面都会进行相应地渲染以确保视野中的画面始终朝上。

只在sdk内部摄像头采集场景生效，并且只在移动端生效。

1. 该接口仅对采集端起作用，如果只是观看房间中的画面，开启此接口是无效的
2. 当采集端设备发生 90 度或 270 度旋转时，采集端或者观众看到的画面可能会被裁剪以保持比例的协调

参数	描述
mode	重力感应模式，详情请参见 TRTCGravitySensorAdaptiveMode_Disable、TRTCGravitySensorAdaptiveMode_FillByCenterCrop 和 TRTCGravitySensorAdaptiveMode_FitWithBlackBorder 默认值：TRTCGravitySensorAdaptiveMode_Disable。

startLocalAudio:

startLocalAudio:

- (void)startLocalAudio:	(TRTCAudioQuality)quality

开启本地音频的采集和发布

SDK 默认不开启麦克风，当用户需要发布本地音频时，需要调用该接口开启麦克风采集，并将音频编码并发布到当前的房间中。

开启本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, YES) 的通知。

参数	描述
quality	声音音质

	TRTCAudioQualitySpeech，流畅：采样率：16k；单声道；音频裸码率：16kbps；适合语音通话为主的场景，比如在线会议，语音通话。 TRTCAudioQualityDefault，默认：采样率：48k；单声道；音频裸码率：50kbps；SDK 默认的音频质量，如无特殊需求推荐选择之。 TRTCAudioQualityMusic，高音质：采样率：48k；双声道 + 全频带；音频裸码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

注意

该函数会检查麦克风的使用权限，如果当前 App 没有麦克风权限，SDK 会自动向用户申请麦克风使用权限。

stopLocalAudio

stopLocalAudio

停止本地音频的采集和发布

停止本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, NO) 的通知。

muteLocalAudio:

muteLocalAudio:

- (void)muteLocalAudio:	(BOOL)mute

暂停/恢复发布本地的音频流

当您暂停发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, NO) 的通知。
当您恢复发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, YES) 的通知。

与 stopLocalAudio 的不同之处在于，muteLocalAudio(YES) 并不会释放麦克风权限，而是继续发送码率极低的静音包。
这对于需要云端录制的场景非常适用，因为 MP4 等格式的视频文件，对于音频数据的连续性要求很高，使用 stopLocalAudio 会导致录制出的 MP4 文件不易播放。
因此在对录制文件的质量要求较高的场景中，建议选择 muteLocalAudio 而不建议使用 stopLocalAudio。

参数	描述
mute	YES：静音；NO：恢复。

版权所有：腾讯云计算（北京）有限责任公司

muteRemoteAudio:mute:

muteRemoteAudio:mute:

- (void)muteRemoteAudio:	(NSString *)userId
mute:	(BOOL)mute

暂停/恢复播放远端的音频流

当您静音某用户的远端音频时，SDK 会停止播放指定用户的声音，同时也会停止拉取该用户的音频数据。

参数	描述
mute	YES：静音；NO：取消静音。
userId	用于指定远端用户的 ID。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 NO。

muteAllRemoteAudio:

muteAllRemoteAudio:

- (void)muteAllRemoteAudio:	(BOOL)mute

暂停/恢复播放所有远端用户的音频流

当您静音所有用户的远端音频时，SDK 会停止播放所有来自远端的音频流，同时也会停止拉取所有用户的音频数据。

参数	描述
mute	YES：静音；NO：取消静音。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 NO。

setAudioRoute:

setAudioRoute:

- (void)setAudioRoute:	(TRTCAudioRoute)route

设置音频路由

设置“音频路由”，即设置声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。

手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

参数	描述
route	音频路由，即声音由哪里输出（扬声器、听筒），默认值：TRTCAudioModeSpeakerphone。

setRemoteAudioVolume:volume:

setRemoteAudioVolume:volume:

- (void)setRemoteAudioVolume:	(NSString *)userId
volume:	(int)volume

设定某一个远端用户的声音播放音量

您可以通过 setRemoteAudioVolume(userId, 0) 将某一个远端用户的声音静音。

参数	描述
userId	用于指定远端用户的 ID。
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setAudioCaptureVolume:

setAudioCaptureVolume:

- (void)setAudioCaptureVolume:	(NSInteger)volume

设定本地音频的采集音量

参数	描述
volume	音量大小，取值范围为0 - 100；默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioCaptureVolume

getAudioCaptureVolume

获取本地音频的采集音量

setAudioPlayoutVolume:

setAudioPlayoutVolume:

- (void)setAudioPlayoutVolume:	(NSInteger)volume

设定远端音频的播放音量

该接口会控制 SDK 最终交给系统播放的声音音量，调节效果会影响到本地音频录制文件的音量大小，但不会影响到耳返的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioPlayoutVolume

getAudioPlayoutVolume

获取远端音频的播放音量

enableAudioVolumeEvaluation:withParams:

enableAudioVolumeEvaluation:withParams:

- (void)enableAudioVolumeEvaluation:	(BOOL)enable
withParams:	(TRTCAudioVolumeEvaluateParams *)params

启用音量大小提示

开启此功能后，SDK 会在 TRTCCloudDelegate 中的 onUserVoiceVolume 回调中反馈本地或远端用户的音频音量评估信息，包括音量大小、人声检测、音频频谱等。

参数	描述
enable	是否启用音量提示，默认为关闭状态。
params	音量评估等相关参数，请参见 TRTCAudioVolumeEvaluateParams。

startAudioRecording:

startAudioRecording:

- (int)startAudioRecording:	(TRTCAudioRecordingParams*) param

开始录音

当您调用该接口后， SDK 会将本地和远端的所有音频（包括本地音频，远端音频，背景音乐和音效等）混合并录制到一个本地文件中。
该接口在进入房间前后调用均可生效，如果录制任务在退出房间前尚未通过 stopAudioRecording 停止，则退出房间后录制任务会自动被停止。
本次录制的启动、完成状态会通过本地录制相关回调进行通知。参见 TRTCCloud 相关回调。

参数	描述
param	录音参数，请参见 TRTCAudioRecordingParams。

注意

自 v11.5 版本，音频录制的状态结果由返回值统一调整为异步回调进行通知。参见 TRTCCloud 相关回调。

返回值说明：

0：成功；-1：录音已开始；-2：文件或目录创建失败；-3：后缀指定的音频格式不支持。

stopAudioRecording

stopAudioRecording

停止录音

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

startLocalRecording:

startLocalRecording:

- (void)startLocalRecording:	(TRTCLocalRecordingParams *)params

开启本地媒体录制

开启后把直播过程中的音视频内容录制到本地的一个文件中。

参数	描述
params	录制参数，请参见 TRTCLocalRecordingParams。

stopLocalRecording

stopLocalRecording

停止本地媒体录制

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

setRemoteAudioParallelParams:

setRemoteAudioParallelParams:

- (void)setRemoteAudioParallelParams:	(TRTCAudioParallelParams*)params

设置远端音频流智能并发播放策略

设置远端音频流智能并发播放策略，适用于上麦人数比较多的场景。

参数	描述
params	音频并发参数，请参见 TRTCAudioParallelParams。

enable3DSpatialAudioEffect:

enable3DSpatialAudioEffect:

- (void)enable3DSpatialAudioEffect:	(BOOL)enabled

启用 3D 音效

启用 3D 音效。注意需使用流畅音质 TRTCAudioQualitySpeech 或默认音质 TRTCAudioQualityDefault。

参数	描述
enabled	是否启用 3D 音效，默认为关闭状态。

updateSelf3DSpatialPosition

updateSelf3DSpatialPosition

设置 3D 音效中自身坐标及朝向信息

更新自身在世界坐标系中的位置和朝向， SDK 会根据该方法参数计算自身和远端用户之间的相对位置，进而渲染出空间音效。注意各参数应分别传入长度为 3 的数组。

参数	描述
axisForward	自身坐标系前轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisRight	自身坐标系右轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisUp	自身坐标系上轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
position	自身在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。

注意

请适当限制调用频率，推荐两次坐标设置至少间隔 100ms。

updateRemote3DSpatialPosition:

updateRemote3DSpatialPosition:

- (void)updateRemote3DSpatialPosition:	(NSString *)userId

设置 3D 音效中远端用户坐标信息

更新远端用户在世界坐标系中的位置，SDK 会根据自身和远端用户之间的相对位置，进而渲染出空间音效。注意参数为长度等于 3 的数组。

参数	描述
position	该远端用户在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。
userId	指定远端用户的 ID。

注意

请适当限制调用频率，推荐同一远端用户两次坐标设置至少间隔 100ms。

set3DSpatialReceivingRange:range:

set3DSpatialReceivingRange:range:

- (void)set3DSpatialReceivingRange:	(NSString *)userId
range:	(NSInteger)range

设置指定用户所发出声音的可被接收范围

设置该范围大小之后，该指定用户的声音将在该范围内可被听见，超出该范围将被衰减为 0。

参数	描述
range	声音最大可被接收范围。
userId	指定远端用户的 ID。

getDeviceManager

getDeviceManager

获取设备管理类（TXDeviceManager）

getBeautyManager

getBeautyManager

获取美颜管理类（TXBeautyManager）

通过美颜管理，您可以使用以下功能：
设置“磨皮”、“美白”、“红润”等美颜特效。
设置“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等修脸特效。
设置“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”等修脸特效。
设置“眼影”、“腮红”等美妆特效。
设置动态贴纸和人脸挂件等动画特效。

setWatermark:streamType:rect:

setWatermark:streamType:rect:

- (void)setWatermark:	(nullable TXImage*)image
streamType:	(TRTCVideoStreamType)streamType
rect:	(CGRect)rect

添加水印

水印的位置是通过 rect 参数来指定的，rect 是一个四元组参数，其格式为 (x，y，width，height)
x：水印的坐标，取值范围为0 - 1的浮点数。
y：水印的坐标，取值范围为0 - 1的浮点数。
width：水印的宽度，取值范围为0 - 1的浮点数。
height：是不用设置的，SDK 内部会根据水印图片的宽高比自动计算一个合适的高度。

参数设置举例：
如果当前视频的编码分辨率是 540 × 960，且 rect 参数被您设置为（0.1，0.1，0.2，0.0），
那么水印的左上坐标点就是（540 × 0.1，960 × 0.1）即（54，96），水印的宽度是 540 × 0.2 = 108px，水印的高度会根据水印图片的宽高比由 SDK 自动算出。

参数	描述
image	水印图片，**必须使用透明底色的 png 格式**。
rect	水印相对于编码分辨率的归一化坐标，x，y，width，height 取值范围0 - 1。
streamType	指定给哪一路画面设置水印，详情请参见TRTCVideoStreamType。

注意

如果您要给主画面（一般为摄像头）和辅路画面（一般用作屏幕分享）同时设置水印，需要调用该接口两次，并设定不同的 streamType。

getAudioEffectManager

getAudioEffectManager

获取音效管理类（TXAudioEffectManager）

TXAudioEffectManager 是音效管理接口，您可以通过该接口实现如下功能：
背景音乐：支持在线音乐和本地音乐，支持变速、变调等特效、支持原生和伴奏并播放和循环播放。
耳机耳返：麦克风捕捉的声音实时通过耳机播放，常用于音乐直播。
混响效果：KTV、小房间、大会堂、低沉、洪亮...。
变声特效：萝莉、大叔、重金属...。
短音效：鼓掌声、欢笑声等简短的音效文件（对于小于10秒的文件，请将 isShortFile 参数设置为 YES）。

startSystemAudioLoopback

startSystemAudioLoopback

开启系统声音采集（iOS 端暂未支持）

该接口会从电脑的声卡中采集音频数据，并将其混入到 SDK 当前的音频数据流中，从而使房间中的其他用户也能听到主播的电脑所播放出的声音。
在线教育场景中，老师可以使用此功能让 SDK 采集教学影片中的声音，并广播给同房间中的学生。
音乐直播场景中，主播可以使用此功能让 SDK 采集音乐播放器中的音乐，从而为自己的直播间增加背景音乐。
注意
1. 此功能需要为用户的 Mac 系统安装虚拟音频设备插件，安装完成后，SDK 会从已经安装的虚拟音频设备中采集声音。
2. SDK 会自动从网络上下载合适的插件进行安装，但是下载速度可能比较慢，如果您希望加速这个过程，可以将虚拟音频插件文件打包到您 App Bundle 的 Resources 目录下。

stopSystemAudioLoopback

stopSystemAudioLoopback

停止系统声音采集（iOS 端暂未支持）

setSystemAudioLoopbackVolume:

setSystemAudioLoopbackVolume:

- (void)setSystemAudioLoopbackVolume:	(uint32_t)volume

设置系统声音的采集音量

参数	描述
volume	设置的音量大小，范围是：[0 ~ 150]，默认值为100。

startScreenCaptureInApp:encParam:

startScreenCaptureInApp:encParam:

- (void)startScreenCaptureInApp:	(TRTCVideoStreamType)streamType
encParam:	(TRTCVideoEncParam *)encParams

开始应用内的屏幕分享（仅支持 iOS 13.0 及以上系统）

该接口会抓取当前应用内的实时屏幕内容并将其分享给同房间中的其他用户，适用于 13.0 以上的 iOS 系统。
如果您希望抓取整个 iOS 系统的屏幕内容（而不受当前应用的限制），推荐使用 startScreenCaptureByReplaykit。

iPhone 推荐的屏幕分享视频编码参数（TRTCVideoEncParam）：
分辨率(videoResolution): 1280 x 720。
帧率(videoFps): 10 FPS。
码率(videoBitrate): 1600 kbps。
分辨率自适应(enableAdjustRes): NO。

参数	描述
encParams	设置屏幕分享时的视频编码参数，推荐采用上述推荐配置。 如果您指定 encParams 为 nil，SDK 会使用您在调用 startScreenCapture 接口之前所设置的视频编码参数。
streamType	屏幕分享使用的线路，可以设置为主路（TRTCVideoStreamTypeBig）或者辅路（TRTCVideoStreamTypeSub）。

startScreenCaptureByReplaykit:encParam:appGroup:

startScreenCaptureByReplaykit:encParam:appGroup:

- (void)startScreenCaptureByReplaykit:	(TRTCVideoStreamType)streamType
encParam:	(TRTCVideoEncParam *)encParams
appGroup:	(NSString *)appGroup

开始全系统的屏幕分享（仅支持 iOS 11.0 及以上系统）

该接口支持抓取整个 iOS 系统的屏幕，可以实现类似腾讯会议的全系统级的屏幕分享。
但对接步骤要比 startScreenCaptureInApp 略繁琐一些，需要为您的应用实现一个 Replaykit 扩展模块。

参考文档：实时屏幕分享(iOS)

iPhone 推荐的屏幕分享视频编码参数（TRTCVideoEncParam）：
分辨率(videoResolution)：1280 x 720。
帧率(videoFps)：10 FPS。
码率(videoBitrate)：1600 kbps。
分辨率自适应(enableAdjustRes)：NO。

参数	描述
appGroup	用于指定您的应用与录屏进程共享的 Application Group Identifier，您可以指定该参数为 nil，但推荐您按照文档指示进行设置，从而获得更好的可靠性。
encParams	设置屏幕分享时的视频编码参数，推荐采用上述推荐配置。如果您指定 encParams 为 nil，SDK 会使用您在调用 startScreenCapture 接口之前所设置的视频编码参数。
streamType	屏幕分享使用的线路，可以设置为主路（TRTCVideoStreamTypeBig）或者辅路（TRTCVideoStreamTypeSub）。

startScreenCapture:streamType:encParam:

startScreenCapture:streamType:encParam:

- (void)startScreenCapture:	(nullable NSView *)view
streamType:	(TRTCVideoStreamType)streamType
encParam:	(nullable TRTCVideoEncParam *)encParam

启动屏幕分享

该接口可以抓取整个屏幕的内容，或抓取您指定的某个应用的窗口内容，并将其分享给同房间中的其他用户。

参数	描述
encParam	屏幕分享的画面编码参数，SDK 会优先使用您通过此接口设置的编码参数： 如果您设置 encParam 为空值，且您已通过 setSubStreamEncoderParam 设置过辅路视频编码参数，SDK 将使用您设置过的辅路编码参数进行屏幕分享。 如果您设置 encParam 为空值，且您未通过 setSubStreamEncoderParam 设置过辅路视频编码参数，SDK 将自动选择一个最佳的编码参数进行屏幕分享。
streamType	屏幕分享使用的线路，可以设置为主路（TRTCVideoStreamTypeBig）或者辅路（TRTCVideoStreamTypeSub），推荐使用辅路。
view	渲染控件所在的父控件，可以设置为空值，表示不显示屏幕分享的预览效果。

注意

1. 同一个用户同时最多只能发布一路主路（TRTCVideoStreamTypeBig）画面和一路辅路（TRTCVideoStreamTypeSub）画面。
2. 默认情况下，屏幕分享使用辅路画面。如果使用主路做屏幕分享，您需要提前停止摄像头采集（stopLocalPreview）以避免相互冲突。
3. 同一个房间中同时只能有一个用户使用辅路做屏幕分享，也就是说，同一个房间中同时只允许一个用户开启辅路。
4. 当房间中已经有其他用户在使用辅路分享屏幕时，此时调用该接口会收到来自 TRTCCloudDelegate 的 onError(ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO) 回调。

stopScreenCapture

stopScreenCapture

停止屏幕分享

pauseScreenCapture

pauseScreenCapture

暂停屏幕分享

注意

从v11.5版本开始，暂停屏幕采集会使用最后一帧按照1fps帧率输出。

resumeScreenCapture

resumeScreenCapture

恢复屏幕分享

getScreenCaptureSourcesWithThumbnailSize:iconSize:

getScreenCaptureSourcesWithThumbnailSize:iconSize:

-(NSArray<TRTCScreenCaptureSourceInfo*>*)getScreenCaptureSourcesWithThumbnailSize:	(CGSize)thumbn
iconSize:	(CGSize)iconSiz

枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）

当您在对接桌面端系统的屏幕分享功能时，一般都需要展示一个选择分享目标的界面，这样用户能够使用这个界面选择是分享整个屏幕还是某个窗口。
通过本接口，您就可以查询到当前系统中可用于分享的窗口的 ID、名称以及缩略图。我们在 Demo 中提供了一份默认的界面实现供您参考。

参数	描述
iconSize	指定要获取的窗口图标大小。
thumbnailSize	指定要获取的窗口缩略图大小，缩略图可用于绘制在窗口选择界面上。

注意

返回的列表中包含屏幕和应用窗口，屏幕是列表中的第一个元素。如果用户有多个显示器，那么每个显示器都是一个分享目标。

返回值说明：

窗口列表包括屏幕。

selectScreenCaptureTarget:rect:capturesCursor:highlight:

selectScreenCaptureTarget:rect:capturesCursor:highlight:

- (void)selectScreenCaptureTarget:	(TRTCScreenCaptureSourceInfo *)screenSource

rect:	(CGRect)rect
capturesCursor:	(BOOL)capturesCursor
highlight:	(BOOL)highlight

选取要分享的屏幕或窗口（该接口仅支持 Mac OS 系统）

当您通过 getScreenCaptureSources 获取到可以分享的屏幕和窗口之后，您可以调用该接口选定期望分享的目标屏幕或目标窗口。
在屏幕分享的过程中，您也可以随时调用该接口以切换分享目标。

参数	描述
capturesCursor	是否捕获鼠标光标。
highlight	是否高亮正在分享的窗口。
rect	指定捕获的区域（设定该参数为 CGRectZero：当分享目标是某个窗口时则分享整个窗口，当分享目标是桌面时则分享整个桌面）。
screenSource	指定分享源。

setSubStreamEncoderParam:

setSubStreamEncoderParam:

- (void)setSubStreamEncoderParam:	(TRTCVideoEncParam *)param

设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）

该接口可以设定远端用户所看到的屏幕分享（即辅路）的画面质量，同时也能决定云端录制出的视频文件中屏幕分享的画面质量。
请注意如下两个接口的差异：
setVideoEncoderParam 用于设置主路画面（TRTCVideoStreamTypeBig，一般用于摄像头）的视频编码参数。
setSubStreamEncoderParam 用于设置辅路画面（TRTCVideoStreamTypeSub，一般用于屏幕分享）的视频编码参数。

参数	描述
param	辅流编码参数，详情请参见 TRTCVideoEncParam。

setSubStreamMixVolume:

setSubStreamMixVolume:

- (void)setSubStreamMixVolume:	(NSInteger)volume

设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）

这个数值越高，屏幕分享音量的占比就越高，麦克风音量占比就越小，所以不推荐设置得太大，否则麦克风的声音就被压制了。

参数	描述
volume	设置的混音音量大小，范围0 - 100。

addExcludedShareWindow:

addExcludedShareWindow:

- (void)addExcludedShareWindow:	(NSInteger)windowID

将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）

加入排除列表中的窗口不会被分享出去，常见的用法是将某个应用的窗口加入到排除列表中以避免隐私问题。
支持启动屏幕分享前设置过滤窗口，也支持屏幕分享过程中动态添加过滤窗口。

参数	描述
windowID	不希望分享出去的窗口

注意

1. 该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeScreen 时生效，即只有在分享整个屏幕内容时，排除指定窗口的功能才生效。
2. 使用该接口添加到排除列表中的窗口会在退出房间后被 SDK 自动清除。
3. Mac 平台下请传入窗口 ID（即 CGWindowID），您可以通过 TRTCScreenCaptureSourceInfo 中的 sourceId 成员获得。

removeExcludedShareWindow:

removeExcludedShareWindow:

- (void)removeExcludedShareWindow:	(NSInteger)windowID

将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）

版权所有：腾讯云计算（北京）有限责任公司

参数	描述
windowID	要排除的窗口 id。

removeAllExcludedShareWindows

removeAllExcludedShareWindows

将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）

addIncludedShareWindow:

addIncludedShareWindow:

- (void)addIncludedShareWindow:	(NSInteger)windowID

将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。
即只有在分享窗口内容时，额外包含指定窗口的功能才生效。
您在 startScreenCapture 之前和之后调用均可。

参数	描述
windowID	希望被分享出去的窗口（Windows 平台下为窗口句柄： HWND）

注意
通过该方法添加到包含列表中的窗口，会在退出房间后被 SDK 自动清除。

removeIncludedShareWindow:

removeIncludedShareWindow:

- (void)removeIncludedShareWindow:	(NSInteger)windowID

将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。
即只有在分享窗口内容时，额外包含指定窗口的功能才生效。

参数	描述
windowID	希望被分享出去的窗口（Mac 平台：窗口 ID；Windows 平台：HWND）

removeAllIncludedShareWindows

removeAllIncludedShareWindows

将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。即只有在分享窗口内容时，额外包含指定窗口的功能才生效。

enableCustomVideoCapture:enable:

enableCustomVideoCapture:enable:

- (void)enableCustomVideoCapture:	(TRTCVideoStreamType)streamType
enable:	(BOOL)enable

启用/关闭视频自定义采集模式

开启该模式后，SDK 不在运行原有的视频采集流程，即不再继续从摄像头采集数据和美颜，而是只保留视频编码和发送能力。
您需要通过 sendCustomVideoData 不断地向 SDK 塞入自己采集的视频画面。

参数	描述
enable	是否启用，默认值：NO。
streamType	用于指定视频流类型，TRTCVideoStreamTypeBig：高清大画面；TRTCVideoStreamTypeSub：辅路画面。

sendCustomVideoData:frame:

sendCustomVideoData:frame:

- (void)sendCustomVideoData:	(TRTCVideoStreamType)streamType
frame:	(TRTCVideoFrame *)frame

向 SDK 投送自己采集的视频帧

使用此接口可以向 SDK 投送自己采集的视频帧，SDK 会将视频帧进行编码并通过自身的网络模块传输出去。

参数 TRTCVideoFrame 推荐下列填写方式（其他字段不需要填写）：
pixelFormat：推荐选择 TRTCVideoPixelFormat_NV12。
bufferType：推荐选择 TRTCVideoBufferType_PixelBuffer。
pixelBuffer：iOS/Mac OS 平台上常用的视频数据格式。
data：视频裸数据格式，bufferType 为 NSData 时使用。
timestamp：时间戳，单位为毫秒（ms），请使用视频帧在采集时被记录下来的时间戳（可以在采集到一帧视频帧之后，通过调用 generateCustomPTS 获取时间戳）。
width：视频图像长度，bufferType 为 NSData 时填写。
height：视频图像宽度，bufferType 为 NSData 时填写。

参考文档：自定义采集和渲染。

参数	描述
frame	视频数据，支持 PixelBuffer NV12，BGRA 以及 I420 格式数据。
streamType	用于指定视频流类型，TRTCVideoStreamTypeBig：高清大画面；TRTCVideoStreamTypeSub：辅路画面。

注意

1. 推荐您在采集到的一帧视频帧后，即调用 generateCustomPTS 接口获取该帧的 timestamp 数值，这样可以获得最佳的音画同步效果。
2. SDK 最终编码出的视频帧率并不是由您调用本接口的频率决定的，而是由您在 setVideoEncoderParam 中所设置的 FPS 决定的。
3. 请尽量保持本接口的调用间隔是均匀的，否则会导致编码器输出帧率不稳或者音画不同步等问题。

enableCustomAudioCapture:

enableCustomAudioCapture:

- (void)enableCustomAudioCapture:	(BOOL)enable

启用音频自定义采集模式

开启该模式后，SDK 不在运行原有的音频采集流程，即不再继续从麦克风采集音频数据，而是只保留音频编码和发送能力。
您需要通过 sendCustomAudioData 不断地向 SDK 塞入自己采集的音频数据。

参数	描述
enable	是否启用，默认值：NO。

注意

由于回声抵消（AEC）需要严格的控制声音采集和播放的时间，所以开启自定义音频采集后，AEC 能力可能会失效。

sendCustomAudioData:

sendCustomAudioData:

- (void)sendCustomAudioData:	(TRTCAudioFrame *)frame

向 SDK 投送自己采集的音频数据

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：
audioFormat：音频数据格式，仅支持 TRTCAudioFrameFormatPCM。
data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。
sampleRate：采样率，支持：16000、24000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在采集到一帧音频帧之后，通过调用 generateCustomPTS 获取时间戳）。

参考文档：自定义采集和渲染。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

enableMixExternalAudioFrame:playout:

enableMixExternalAudioFrame:playout:

- (void)enableMixExternalAudioFrame:	(BOOL)enablePublish

playout:	(BOOL)enablePlayout

启用/关闭自定义音轨

开启后，您可以通过本接口向 SDK 混入一条自定义的音轨。通过两个布尔型参数，您可以控制该音轨是否要在远端和本地播放。

参数	描述
enablePlayout	控制混入的音轨是否要在本地播放，默认值：NO。
enablePublish	控制混入的音轨是否要在远端播放，默认值：NO。

注意

如果您指定参数 enablePublish 和 enablePlayout 均为 NO，代表完全关闭您的自定义音轨。

mixExternalAudioFrame:

mixExternalAudioFrame:

- (int)mixExternalAudioFrame:	(TRTCAudioFrame *)frame

向 SDK 混入自定义音轨

调用该接口之前，您需要先通过 enableMixExternalAudioFrame 开启自定义音轨，之后就可以通过本接口将自己的音轨以 PCM 格式混入到 SDK 中。

理想情况下，我们期望您的代码能够以非常均匀的速度向 SDK 提供音轨数据。但我们也非常清楚，完美的调用间隔是一个巨大的挑战。

所以 SDK 内部会开启一个音轨数据的缓冲区，该缓冲区的作用类似一个“蓄水池”，它能够暂存您传入的音轨数据，平抑由于接口调用间隔的抖动问题。

本接口的返回值代表这个音轨缓冲区的大小，单位是毫秒（ms），比如：如果该接口返回 50，则代表当前的音轨缓冲区有 50ms 的音轨数据。因此只要您在 50ms 内再次调用本接口，SDK 就能保证您混入的音轨数据是连续的。

当您调用该接口后，如果发现返回值 > 100ms，则可以等待一帧音频帧的播放时间之后再次调用；如果返回值 < 100ms，则代表缓冲区比较小，您可以再次混入一些音轨数据以确保音轨缓冲区的大小维持在“安全水位”以上。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：

data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。

sampleRate：采样率，支持：16000、24000、32000、44100、48000。

channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。

timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在获得一帧音频帧之后，通过调用 generateCustomPTS 获得时间戳）。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

返回值说明：

>= 0 缓冲的长度，单位：ms。< 0 错误（-1 未启用 mixExternalAudioFrame）

setMixExternalAudioVolume:playoutVolume:

setMixExternalAudioVolume:playoutVolume:

- (void)setMixExternalAudioVolume:	(NSInteger)publishVolume
playoutVolume:	(NSInteger)playoutVolume

设置推流时混入外部音频的推流音量和播放音量

参数	描述
playoutVolume	设置的播放音量大小，范围0 - 100，-1表示不改变。
publishVolume	设置的推流音量大小，范围0 - 100，-1表示不改变。

generateCustomPTS

generateCustomPTS

生成自定义采集时的时间戳

本接口仅适用于自定义采集模式，用于解决音视频帧的采集时间（capture time）和投送时间（send time）不一致所导致的音画不同步问题。

当您通过 sendCustomVideoData 或 sendCustomAudioData 等接口进行自定义视频或音频采集时，请按照如下操作使用该接口：

1. 首先，在采集到一帧视频或音频帧时，通过调用本接口获得当时的 PTS 时间戳。

2. 之后可以将该视频或音频帧送入您使用的前处理模块（如第三方美颜组件，或第三方音效组件）。
3. 在真正调用 sendCustomVideoData 或 sendCustomAudioData 进行投送时，请将该帧在采集时记录的 PTS 时间戳赋值给 TRTCVideoFrame 或 TRTCAudioFrame 中的 timestamp 字段。

返回值说明：

时间戳（单位：ms）

setLocalVideoProcessDelegete:pixelFormat:bufferType:

setLocalVideoProcessDelegete:pixelFormat:bufferType:

- (int)setLocalVideoProcessDelegete:	(nullable id<TRTCVideoFrameDelegate>)delegate
pixelFormat:	(TRTCVideoPixelFormat)pixelFormat
bufferType:	(TRTCVideoBufferType)bufferType

设置第三方美颜的视频数据回调

设置该回调之后，SDK 会把采集到的视频帧通过您设置的 delegate 回调出来，用于第三方美颜组件进行二次处理，之后 SDK 会将处理后的视频帧进行编码和发送。

参数	描述
bufferType	指定回调的数据格式，目前仅支持 TRTCVideoBufferType_Texture
delegate	自定义预处理回调，详见 TRTCVideoFrameDelegate
pixelFormat	指定回调的像素格式，目前仅支持 TRTCVideoPixelFormat_Texture_2D

返回值说明：

0：成功；<0：错误

setLocalVideoRenderDelegate:pixelFormat:bufferType:

setLocalVideoRenderDelegate:pixelFormat:bufferType:

- (int)setLocalVideoRenderDelegate:	(nullable id<TRTCVideoRenderDelegate>)delegate
pixelFormat:	(TRTCVideoPixelFormat)pixelFormat
bufferType:	(TRTCVideoBufferType)bufferType

设置本地视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
pixelFormat 指定回调的数据格式，例如 NV12、i420 以及 32BGRA。
bufferType 指定 buffer 的类型，直接使用 PixelBuffer 效率最高；使用 NSData 相当于让 SDK 在内部做了一次内存转换，因此会有额外的性能损耗。
参考文档：自定义采集和渲染。

参数	描述
bufferType	PixelBuffer：可以直接使用 imageWithCVImageBuffer 转成 UIImage；NSData：经过内存整理的视频数据。
delegate	自定义渲染回调。
pixelFormat	指定回调的像素格式。

返回值说明：

0：成功；< 0：错误。

setRemoteVideoRenderDelegate:delegate:pixelFormat:bufferType:

setRemoteVideoRenderDelegate:delegate:pixelFormat:bufferType:

- (int)setRemoteVideoRenderDelegate:	(NSString*)userId
delegate:	(nullable id<TRTCVideoRenderDelegate>)delegate
pixelFormat:	(TRTCVideoPixelFormat)pixelFormat
bufferType:	(TRTCVideoBufferType)bufferType

设置远端视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
pixelFormat 指定回调的数据格式，例如 NV12、i420 以及 32BGRA。
bufferType 指定 buffer 的类型，直接使用 PixelBuffer 效率最高；使用 NSData 相当于让 SDK 在内部做了一次内存转换，因此会有额外的性能损耗。

参考文档：自定义采集和渲染。

参数	描述
bufferType	PixelBuffer：可以直接使用 imageWithCVImageBuffer 转成 UIImage；NSData：经过内

	存整理的视频数据。
delegate	自定义渲染回调
pixelFormat	指定回调的像素格式
userId	指定远端用户的 ID

注意

调用此函数之前，需要先调用 startRemoteView(nil) 来获取远端用户的视频流（view 设置为 nil 即可），否则不会有数据回调出来。

返回值说明：

0：成功；<0：错误

setAudioFrameDelegate:

setAudioFrameDelegate:

- (void)setAudioFrameDelegate:	(nullable id<TRTCAudioFrameDelegate>)delegate

设置音频数据自定义回调

设置该回调之后，SDK 内部会把音频数据（PCM 格式）回调出来，包括：

onCapturedAudioFrame：本地麦克风采集到的音频数据回调

onLocalProcessedAudioFrame：本地采集并经过音频模块前处理后的音频数据回调

onRemoteUserAudioFrame：混音前的每一路远程用户的音频数据

onMixedPlayAudioFrame：将各路音频混合之后并最终要由系统播放出的音频数据回调

注意

设置回调为空即代表停止自定义音频回调，反之，设置回调不为空则代表启动自定义音频回调。

setCapturedAudioFrameDelegateFormat:

setCapturedAudioFrameDelegateFormat:

- (int)setCapturedAudioFrameDelegateFormat:	(TRTCAudioFrameDelegateFormat *)format

设置本地麦克风采集出的音频帧回调格式

本接口用于设置 onCapturedAudioFrame 回调出来的 AudioFrame 的格式：

sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调 20ms 帧长的数据，则采样点数应该填：960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意
最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setLocalProcessedAudioFrameDelegateFormat:

setLocalProcessedAudioFrameDelegateFormat:

- (int)setLocalProcessedAudioFrameDelegateFormat:	(TRTCAudioFrameDelegateFormat *)format

设置经过前处理后的本地音频帧回调格式

本接口用于设置 onLocalProcessedAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setMixedPlayAudioFrameDelegateFormat:

setMixedPlayAudioFrameDelegateFormat:

- (int)setMixedPlayAudioFrameDelegateFormat:	(TRTCAudioFrameDelegateFormat *)format

设置最终要由系统播放出的音频帧回调格式

本接口用于设置 onMixedPlayAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

enableCustomAudioRendering:

enableCustomAudioRendering:

- (void)enableCustomAudioRendering:	(BOOL)enable

开启音频自定义播放

如果您需要外接一些特定的音频设备，或者希望自己掌控音频的播放逻辑，您可以通过该接口启用音频自定义播放。
启用音频自定义播放后，SDK 将不再调用系统的音频接口播放数据，您需要通过 getCustomAudioRenderingFrame 获取 SDK 要播放的音频帧并自行播放。

参数	描述
enable	是否启用音频自定义播放，默认为关闭状态。

注意
需要您在进入房间前设置才能生效，暂不支持进入房间后再设置。

getCustomAudioRenderingFrame:

getCustomAudioRenderingFrame:

- (void)getCustomAudioRenderingFrame:	(TRTCAudioFrame *)audioFrame

获取可播放的音频数据

调用该接口之前，您需要先通过 enableCustomAudioRendering 开启音频自定义播放。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：
sampleRate：采样率，必填，支持 16000、24000、32000、44100、48000。
channel：声道数，必填，单声道请填1，双声道请填2，双声道时数据是交叉的。
data：用于获取音频数据的 buffer。需要您根据一帧音频帧的帧长度分配好 data 的内存大小。获取的 PCM 数据支持 10ms 或 20ms 两种帧长，推荐使用 20ms 的帧长。

计算公式为：采样率 x 播放时长 x 声道数量 x 2（每个样点的字节数），例如：48000采样率、单声道、且播放时长为 20ms 的一帧音频帧的 buffer 大小为 48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节。

参数	描述
audioFrame	音频数据帧。

注意
1. 参数 audioFrame 中的 sampleRate、channel 需提前设置好，同时分配好所需读取帧长的 data 空间。
2. SDK 内部会根据 sampleRate 和 channel 自动填充 data 数据。
3. 建议由系统的音频播放线程直接驱动该函数的调用，在播放完一帧音频之后，即调用该接口获取下一帧可播放的音频数据。

版权所有：腾讯云计算（北京）有限责任公司

sendCustomCmdMsg:data:reliable:ordered:

sendCustomCmdMsg:data:reliable:ordered:

- (BOOL)sendCustomCmdMsg:	(NSInteger)cmdID
data:	(NSData *)data
reliable:	(BOOL)reliable
ordered:	(BOOL)ordered

使用 UDP 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 UDP 通道，向当前房间里的其他用户广播自定义数据，以达到传输信令的目的。
房间中的其他用户可以通过 TRTCCloudDelegate 中的 onRecvCustomCmdMsg 回调接收消息。

参数	描述
cmdID	消息 ID，取值范围为1 - 10。
data	待发送的消息，单个消息的最大长度被限制为 1KB。
ordered	是否要求有序，即是否要求接收端的数据包顺序和发送端的数据包顺序一致（这会带来一定的接收延时）。
reliable	是否可靠发送，可靠发送可以获得更高的发送成功率，但可靠发送比不可靠发送会带来更大的接收延迟。

注意

1. 发送消息到房间内所有用户（暂时不支持 Web/小程序端），每秒最多能发送30条消息。
2. 每个包最大为 1KB，超过则很有可能会被中间路由器或者服务器丢弃。
3. 每个客户端每秒最多能发送总计 8KB 数据。
4. 请将 reliable 和 ordered 同时设置为 YES 或同时设置为 NO，暂不支持交叉设置。
5. 强烈建议您将不同类型的消息设定为不同的 cmdID，这样可以在要求有序的情况下减小消息时延。
6. 目前仅支持主播身份。

返回值说明：

YES：消息已经发出；NO：消息发送失败。

sendSEIMsg:repeatCount:

sendSEIMsg:repeatCount:

- (BOOL)sendSEIMsg:	(NSData *)data
repeatCount:	(int)repeatCount

使用 SEI 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 SEI 通道，向当前房间里的其他用户广播自定义数据，已达到传输信令的目的。

视频帧的头部有一个叫做 SEI 的头部数据块，该接口的原理就是利用这个被称为 SEI 的头部数据块，将您要发送的自定义信令嵌入其中，使其同视频帧一并发送出去。
因此，与 sendCustomCmdMsg 相比，SEI 通道传输的信令具有更好的兼容性：信令可以伴随着视频帧一直传输到直播 CDN 上。
不过，由于视频帧头部的数据块不能太大，建议您使用该接口时，尽量将信令控制在几个字节的大小。

最常见的用法是把自定义的时间戳（timestamp）用本接口嵌入视频帧中，实现消息和画面的完美对齐（比如：教育场景下的课件和视频信号的对齐）。
房间中的其他用户可以通过 TRTCCloudDelegate 中的 onRecvSEIMsg 回调接收消息。

参数	描述
data	待发送的数据，最大支持 1KB（1000字节）的数据大小
repeatCount	发送数据次数

注意

本接口有以下限制：
1. 数据在接口调用完后不会被即时发送出去，而是从下一帧视频帧开始与视频帧一起发送。
2. 发送消息到房间内所有用户，每秒最多能发送 30 条消息（与 sendCustomCmdMsg 共享限制）。
3. 每个包最大为 1KB，若发送大量数据，会导致视频码率增大，可能导致视频画质下降甚至卡顿（与 sendCustomCmdMsg 共享限制）。
4. 每个客户端每秒最多能发送总计8KB数据（与 sendCustomCmdMsg 共享限制）。
5. 若指定多次发送（repeatCount > 1），则数据会被带在后续的连续 repeatCount 个视频帧中发送出去，同样会导致视频码率增大。
6. 如果 repeatCount > 1，多次发送，接收消息 onRecvSEIMsg 回调也可能会收到多次相同的消息，需要去重。

返回值说明：

YES：消息已通过限制，等待后续视频帧发送；NO：消息被限制发送

startSpeedTest:

startSpeedTest:

- (int)startSpeedTest:	(TRTCSpeedTestParams *)params

开始进行网速测试（进入房间前使用）

参数	描述
params	测速选项

注意

1. 测速过程将产生少量的基础服务费用，详见 计费概述 > 基础服务 文档说明。
2. 请在进入房间前进行网速测试，在房间中网速测试会影响正常的音视频传输效果，而且由于干扰过多，网速测试结果也不准确。
3. 同一时间只允许一项网速测试任务运行。

返回值说明：

接口调用结果，< 0：失败

stopSpeedTest

stopSpeedTest

停止网络测速

getSDKVersion

getSDKVersion

获取 SDK 版本信息

setLogLevel:

setLogLevel:

+ (void)setLogLevel:	(TRTCLogLevel)level

设置 Log 输出级别

参数	描述
level	参见 TRTCLogLevel，默认值：TRTCLogLevelNone

setConsoleEnabled:

setConsoleEnabled:

+ (void)setConsoleEnabled:	(BOOL)enabled

启用/禁用控制台日志打印

参数	描述
enabled	指定是否启用，默认：禁止状态。

setLogCompressEnabled:

setLogCompressEnabled:

+ (void)setLogCompressEnabled:	(BOOL)enabled

启用/禁用日志的本地压缩

开启压缩后，Log 存储体积明显减小，但需要腾讯云提供的 Python 脚本解压后才能阅读。
禁用压缩后，Log 采用明文存储，可以直接用记事本打开阅读，但占用空间较大。

参数	描述
enabled	指定是否启用，默认为启动状态

setLogDirPath:

setLogDirPath:

+ (void)setLogDirPath:	(NSString *)path

设置本地日志的保存路径

通过该接口您可以更改 SDK 本地日志的默认存储路径，SDK 默认的本地日志的存储位置：
Windows 平台：在 C:/Users/[系统用户名]/AppData/Roaming/liteav/log，即 %appdata%/liteav/log 下。

iOS 或 Mac 平台：在 sandbox Documents/log 下。
Android 平台：在 /app私有目录/files/log/liteav/ 下。

参数	描述
path	存储日志的路径

注意

请务必在所有其他接口之前调用，并且保证您指定的目录是存在的，并且您的应用程序拥有对该目录的读写权限。

setLogDelegate:

setLogDelegate:

+ (void)setLogDelegate:	(nullable id<TRTCLogDelegate>)logDelegate

设置日志回调

showDebugView:

showDebugView:

- (void)showDebugView:	(NSInteger)showType

显示仪表盘

“仪表盘”是位于视频渲染控件之上的一个半透明的调试信息浮层，用于展示音视频信息和事件信息，便于对接和调试。

参数	描述
showType	0：不显示；1：显示精简版（仅显示音视频信息）；2：显示完整版（包含音视频信息和事件信息）。

setDebugViewMargin:margin:

setDebugViewMargin:margin:

- (void)setDebugViewMargin:	(NSString *)userId
margin:	(TXEdgeInsets)margin

设置仪表盘的边距

用于调整仪表盘在视频渲染控件中的位置，必须在 showDebugView 之前调用才能生效。

参数	描述
margin	仪表盘内边距，注意这里是基于 parentView 的百分比，margin 的取值范围是0 - 1。
userId	用户 ID。

callExperimentalAPI:

callExperimentalAPI:

- (NSString*)callExperimentalAPI:	(NSString*)jsonStr

调用实验性接口

enablePayloadPrivateEncryption:params:

enablePayloadPrivateEncryption:params:

- (int)enablePayloadPrivateEncryption:	(BOOL)enabled
params:	(TRTCPayloadPrivateEncryptionConfig *)config

开启或关闭媒体流私有加密

在安全要求较高的场景下，TRTC 建议您在加入房间前，调用 enablePayloadPrivateEncryption 方法开启媒体流私有加密。
用户退出房间后，SDK 会自动关闭私有加密。如需重新开启私有加密，您需要在用户再次加入房间前调用该方法。

参数	描述
config	配置媒体流私有加密的算法和密钥，参见 TRTCPayloadPrivateEncryptionConfig。
enabled	是否开启媒体流私有加密。

注意

TRTC 已经内置对媒体流进行加密后再传输，启用媒体流私有加密后将使用您传入的密匙与初始向量进行再次加密。

返回值说明：

接口调用结果，0: 方法调用成功， -1: 传入参数无效， -2: 功能已过期。若需解锁：请前往中国大陆站点开通 TRTC 旗舰版套餐，并填写 **申请**，审核通过后方可使用。

TRTCCloudDelegate

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTCCloudDelegate @ TXLiteAVSDK

Function: 腾讯云实时音视频的事件回调接口

TRTCCloudDelegate

TRTCCloudDelegate

函数列表	描述
onError:errMsg:extInfo:	错误事件回调
onWarning:warningMsg:extInfo:	警告事件回调
onEnterRoom:	进入房间成功与否的事件回调
onExitRoom:	离开房间的事件回调
onSwitchRole:errMsg:	切换角色的事件回调
onSwitchRoom:errMsg:	切换房间的结果回调
onConnectOtherRoom:errCode:errMsg:	请求跨房通话的结果回调
onDisconnectOtherRoom:errMsg:	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode:errMsg:	更改跨房主播上行能力的结果回调
onRemoteUserEnterRoom:	有用户加入当前房间
onRemoteUserLeaveRoom:reason:	有用户离开当前房间
onUserVideoAvailable:available:	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable:available:	某远端用户发布/取消了辅路视频画面

onUserAudioAvailable:available:	某远端用户发布/取消了自己的音频
onFirstVideoFrame:streamType:width:height:	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame:	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame:	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated:streamType:streamStatus:reason:extrainfo:	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated:streamStatus:reason:extrainfo:	远端音频状态变化的事件回调
onUserVideoSizeChanged:streamType:newWidth:newHeight:	用户视频大小发生改变回调
onNetworkQuality:remoteQuality:	网络质量的实时统计回调
onStatistics:	音视频技术指标的实时统计回调
onSpeedTestResult:	网速测试的结果回调
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onAudioRouteChanged:fromRoute:	当前音频路由发生变化（仅适用于移动设备）
onUserVoiceVolume:totalVolume:	音量大小的反馈回调
onDevice:type:stateChanged:	本地设备的通断状态发生变化（仅适用于桌面系统）
onAudioDeviceCaptureVolumeChanged:muted:	当前麦克风的系统采集音量发生变化
onAudioDevicePlayoutVolumeChanged:muted:	当前系统的播放音量发生变化

版权所有：腾讯云计算（北京）有限责任公司

onSystemAudioLoopbackError:	系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）
onRecvCustomCmdMsgUserId:cmdID:seq:message:	收到自定义消息的事件回调
onMissCustomCmdMsgUserId:cmdID:errCode:missed:	自定义消息丢失的事件回调
onRecvSEIMsg:message:	收到 SEI 消息的回调
onStartPublishing:errMsg:	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing:errMsg:	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream:errMsg:	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream:errMsg:	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig:errMsg:	设置云端混流的排版布局和转码参数的事件回调
onStartPublishMediaStream:code:message:extraInfo:	开始发布媒体流的事件回调
onUpdatePublishMediaStream:code:message:extraInfo:	更新媒体流的事件回调
onStopPublishMediaStream:code:message:extraInfo:	停止媒体流的事件回调
onCdnStreamStateChanged:status:code:msg:extraInfo:	RTMP/RTMPS 推流状态发生改变回调
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused:	屏幕分享暂停的事件回调
onScreenCaptureResumed:	屏幕分享恢复的事件回调
onScreenCaptureStoped:	屏幕分享停止的事件回调
onLocalRecordBegin:storagePath:	本地录制任务已经开始的事件回调
onLocalRecording:storagePath:	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment:	本地录制分片的事件回调

版权所有：腾讯云计算（北京）有限责任公司

onLocalRecordComplete:storagePath:	本地录制任务已经结束的事件回调
onUserEnter:	有主播加入当前房间（已废弃）
onUserExit:reason:	有主播离开当前房间（已废弃）
onAudioEffectFinished:code:	音效播放已结束（已废弃）

TRTCVideoRenderDelegate

函数列表	描述
onRenderVideoFrame:userId:streamType:	自定义视频渲染回调

TRTCVideoFrameDelegate

函数列表	描述
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame:dstFrame:	用于对接第三方美颜组件的视频处理回调
onGLContextDestory	SDK 内部 OpenGL 环境被销毁的通知

TRTCAudioFrameDelegate

函数列表	描述
onCapturedAudioFrame:	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame:	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onRemoteUserAudioFrame:userId:	混音前的每一路远程用户的音频数据
onMixedPlayAudioFrame:	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame:	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）
onVoiceEarMonitorAudioFrame:	耳返的音频数据

版权所有：腾讯云计算（北京）有限责任公司

TRTCLogDelegate

函数列表	描述
onLog:LogLevel:WhichModule:	本地 LOG 的打印回调

onError:errMsg:extInfo:

onError:errMsg:extInfo:

- (void)onError:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg
extInfo:	(nullable NSDictionary*)extInfo

错误事件回调

错误事件，表示 SDK 抛出的不可恢复的错误，比如进入房间失败或设备开启失败等。
参考文档：**错误码表**

参数	描述
errCode	错误码
errMsg	错误信息
extInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题

onWarning:warningMsg:extInfo:

onWarning:warningMsg:extInfo:

- (void)onWarning:	(TXLiteAVWarning)warningCode
warningMsg:	(nullable NSString *)warningMsg
extInfo:	(nullable NSDictionary*)extInfo

警告事件回调

警告事件，表示 SDK 抛出的提示性问题，比如视频出现卡顿或 CPU 使用率太高等。
参考文档：**错误码表**

参数	描述
extInfo	扩展信息字段，个别警告码可能会带额外的信息帮助定位问题
warningCode	警告码
warningMsg	警告信息

onEnterRoom:

onEnterRoom:

- (void)onEnterRoom:	(NSInteger)result

进入房间成功与否的事件回调

调用 TRTCCloud 中的 enterRoom() 接口执行进房操作后，会收到来自 TRTCCloudDelegate 的 onEnterRoom(result) 回调：
如果加入成功，回调 result 会是一个正数（result > 0），代表进入房间所消耗的时间，单位是毫秒（ms）。
如果加入失败，回调 result 会是一个负数（result < 0），代表失败原因的错误码。
进房失败的错误码含义请参见**错误码表**。

参数	描述
result	result > 0 时为进房耗时（ms），result < 0 时为进房错误码。

注意

1. 在 Ver6.6 之前的版本，只有进房成功会抛出 onEnterRoom(result) 回调，进房失败由 onError() 回调抛出。
2. 在 Ver6.6 之后的版本：无论进房成功或失败，均会抛出 onEnterRoom(result) 回调，同时进房失败也会有 onError() 回调抛出。

onExitRoom:

onExitRoom:

- (void)onExitRoom:	(NSInteger)reason

离开房间的事件回调

调用 TRTCCloud 中的 exitRoom 接口会执行退出房间的相关逻辑，例如释放音视频设备资源和编解码器资源等。待 SDK 占用的所有资源释放完毕后，SDK 会抛出 onExitRoom() 回调通知到您。

如果您要再次调用 enterRoom 或者切换到其他的音视频 SDK，请等待 onExitRoom 回调到来后再执行相关操作。否则可能会遇到例如摄像头、麦克风设备被强占等各种异常问题。

参数	描述
reason	离开房间原因，0：主动调用 exitRoom 退出房间；1：被服务器踢出当前房间；2：当前房间整个被解散。

onSwitchRole:errMsg:

onSwitchRole:errMsg:

- (void)onSwitchRole:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg

切换角色的事件回调

调用 TRTCCloud 中的 switchRole() 接口会切换主播和观众的角色，该操作会伴随一个线路切换的过程，待 SDK 切换完成后，会抛出 onSwitchRole() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onSwitchRoom:errMsg:

onSwitchRoom:errMsg:

- (void)onSwitchRoom:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg

切换房间的结果回调

调用 TRTCCloud 中的 switchRoom() 接口可以让用户快速地从一个房间切换到另一个房间，待 SDK 切换完成后，会抛出 onSwitchRoom() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onConnectOtherRoom:errCode:errMsg:

onConnectOtherRoom:errCode:errMsg:

- (void)onConnectOtherRoom:	(NSString*)userId
errCode:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg

请求跨房通话的结果回调

调用 TRTCCloud 中的 connectOtherRoom() 接口会将两个不同房间中的主播拉通视频通话，也就是所谓的“主播 PK”功能。

调用者会收到 onConnectOtherRoom() 回调来获知跨房通话是否成功，

如果成功，两个房间中的所有用户都会收到来自另一个房间中的 PK 主播的 onUserVideoAvailable() 回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。
userId	要跨房通话的另一个房间中的主播的用户 ID。

onDisconnectOtherRoom:errMsg:

onDisconnectOtherRoom:errMsg:

- (void)onDisconnectOtherRoom:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg

结束跨房通话的结果回调

onUpdateOtherRoomForwardMode:errMsg:

onUpdateOtherRoomForwardMode:errMsg:

- (void)onUpdateOtherRoomForwardMode:	(TXLiteAVError)errCode
errMsg:	(nullable NSString *)errMsg

更改跨房主播上行能力的结果回调

onRemoteUserEnterRoom:

onRemoteUserEnterRoom:

- (void)onRemoteUserEnterRoom:	(NSString *)userId

有用户加入当前房间

出于性能方面的考虑，在 TRTC 两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：
直播类场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）：该场景下的用户区分主播和观众两种角色，只有主播进入房间时才会触发该通知，观众进入房间时不会触发该通知。
通话类场景（TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall）：该场景下的用户没有角色的区分（可认为都是主播），任何用户进入房间都会触发该通知。

参数	描述
userId	远端用户的用户标识。

注意

1. 事件回调 onRemoteUserEnterRoom 和 onRemoteUserLeaveRoom 只适用于维护当前房间里的“用户列表”，有此事件回调不代表一定有视频画面。
2. 如果需要显示远程画面，请监听代表某个用户是否有视频画面的 onUserVideoAvailable() 事件回调。

onRemoteUserLeaveRoom:reason:

onRemoteUserLeaveRoom:reason:

- (void)onRemoteUserLeaveRoom:	(NSString *)userId
reason:	(NSInteger)reason

有用户离开当前房间

与 onRemoteUserEnterRoom 相对应，在两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：
直播类场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）：只有主播离开房间时才会触发该通知，观众离开房间不会触发该通知。
通话类场景（TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall）：该场景下用户没有角色的区别，任何用户的离开都会触发该通知。

参数	描述
reason	离开原因，0表示用户主动退出房间，1表示用户超时退出，2表示被踢出房间，3表示主播因切换到观众退出房间。
userId	远端用户的用户标识。

onUserVideoAvailable:available:

onUserVideoAvailable:available:

- (void)onUserVideoAvailable:	(NSString *)userId
available:	(BOOL)available

某远端用户发布/取消了主路视频画面

主路画面 一般被用于承载摄像头画面。当您收到 onUserVideoAvailable(userId, YES) 通知时，表示该路画面已经有可播放的视频帧到达。
此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

当您收到 onUserVideoAvailable(userId, NO) 通知时，表示该路远程画面已经被关闭，关闭的原因可能是该用户调用了 muteLocalVideo 或 stopLocalPreview。

参数	描述
available	该用户是否发布（或取消发布）了主路视频画面，YES：发布；NO：取消发布。
userId	远端用户的用户标识。

onUserSubStreamAvailable:available:

onUserSubStreamAvailable:available:

- (void)onUserSubStreamAvailable:	(NSString *)userId
available:	(BOOL)available

某远端用户发布/取消了辅路视频画面

“辅路画面”一般被用于承载屏幕分享的画面。当您收到 onUserSubStreamAvailable(userId, YES) 通知时，表示该路画面已经有可播放的视频帧到达。

此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

参数	描述
available	该用户是否发布（或取消发布）了辅路视频画面，YES: 发布；NO：取消发布。
userId	远端用户的用户标识。

注意

显示辅路画面使用的函数是 startRemoteView 而非 startRemoteSubStreamView，startRemoteSubStreamView 已经被废弃。

onUserAudioAvailable:available:

onUserAudioAvailable:available:

- (void)onUserAudioAvailable:	(NSString *)userId
available:	(BOOL)available

某远端用户发布/取消了自己的音频

当您收到 onUserAudioAvailable(userId, YES) 通知时，表示该用户发布了自己的声音，此时 SDK 的表现为：

在自动订阅模式下，您无需做任何操作，SDK 会自动播放该用户的声音。

在手动订阅模式下，您可以通过 muteRemoteAudio(userid, NO) 来播放该用户的声音。

参数	描述
available	该用户是否发布（或取消发布）了自己的音频，YES: 发布；NO：取消发布。
userId	远端用户的用户标识。

注意

SDK 默认使用自动订阅模式，您可以通过 setDefaultStreamRecvMode 设置为手动订阅，但需要在您进入房间之前调用才生效。

onFirstVideoFrame:streamType:width:height:

onFirstVideoFrame:streamType:width:height:

- (void)onFirstVideoFrame:	(NSString*)userId
streamType:	(TRTCVideoStreamType)streamType
width:	(int)width
height:	(int)height

SDK 开始渲染自己本地或远端用户的首帧画面

SDK 会在渲染自己本地或远端用户的首帧画面时抛出该事件，您可以通过回调事件中的 userId 参数来判断事件来自于“本地”还是来自于“远端”。

如果 userId 为空值，代表 SDK 已经开始渲染自己本地的视频画面，不过前提是您已经调用了 startLocalPreview 或 startScreenCapture。

如果 userId 不为空，代表 SDK 已经开始渲染远端用户的视频画面，不过前提是您已经调用了 startRemoteView 订阅了该用户的视频画面。

参数	描述
height	画面的高度。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	本地或远端的用户标识，如果 userId 为空值代表自己本地的首帧画面已到来，userId 不为空则代表远端用户的首帧画面已到来。
width	画面的宽度。

注意

1. 只有当您调用了 startLocalPreview 或 startScreenCapture 之后，才会触发自己本地的首帧画面事件回调。
2. 只有当您调用了 startRemoteView 或 startRemoteSubStreamView 之后，才会触发远端用户的首帧画面事件回调。

onFirstAudioFrame:

onFirstAudioFrame:

- (void)onFirstAudioFrame:	(NSString*)userId

SDK 开始播放远端用户的首帧音频

SDK 会在播放远端用户的首帧音频时抛出该事件，本地音频的首帧事件暂不抛出。

参数	描述
userId	远端用户的用户标识。

onSendFirstLocalVideoFrame:

onSendFirstLocalVideoFrame:

- (void)onSendFirstLocalVideoFrame:	(TRTCVideoStreamType)streamType

自己本地的首个视频帧已被发布出去

当您成功进入房间并通过 startLocalPreview 或 startScreenCapture 开启本地视频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行视频编码，并通过自身的网络模块向云端发布自己本地的视频数据。当 SDK 成功地向云端送出自己的第一帧视频数据帧以后，就会抛出 onSendFirstLocalVideoFrame 事件回调。

参数	描述
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。

onSendFirstLocalAudioFrame

onSendFirstLocalAudioFrame

自己本地的首个音频帧已被发布出去

当您成功进入房间并通过 startLocalAudio 开启本地音频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行音频编码，并通过自身的网络模块向云端发布自己本地的音频数据。当 SDK 成功地向云端送出自己的第一帧音频数据帧以后，就会抛出 onSendFirstLocalAudioFrame 事件回调。

onRemoteVideoStatusUpdated:streamType:streamStatus:reason:extrainfo:

onRemoteVideoStatusUpdated:streamType:streamStatus:reason:extrainfo:

- (void)onRemoteVideoStatusUpdated:	(NSString *)userId
streamType:	(TRTCVideoStreamType)streamType
streamStatus:	(TRTCAVStatusType)status
reason:	(TRTCAVStatusChangeReason)reason
extrainfo:	(nullable NSDictionary *)extrainfo

远端视频状态变化的事件回调

您可以通过此事件回调获取远端每一路画面的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。
reason	视频状态改变的原因。
status	视频状态：包括 Playing、Loading 和 Stopped 三种状态。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onRemoteAudioStatusUpdated:streamStatus:reason:extrainfo:

onRemoteAudioStatusUpdated:streamStatus:reason:extrainfo:

- (void)onRemoteAudioStatusUpdated:	(NSString *)userId
streamStatus:	(TRTCAVStatusType)status
reason:	(TRTCAVStatusChangeReason)reason
extrainfo:	(nullable NSDictionary *)extrainfo

远端音频状态变化的事件回调

您可以通过此事件回调获取远端每一路音频的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。
reason	音频状态改变的原因。
status	音频状态：包括 Playing、Loading 和 Stopped 三种状态。
userId	用户标识。

onUserVideoSizeChanged:streamType:newWidth:newHeight:

onUserVideoSizeChanged:streamType:newWidth:newHeight:

- (void)onUserVideoSizeChanged:	(NSString *)userId
streamType:	(TRTCVideoStreamType)streamType
newWidth:	(int)newWidth
newHeight:	(int)newHeight

用户视频大小发生改变回调

当您收到 onUserVideoSizeChanged(userId, streamtype, newWidth, newHeight) 通知时，表示该路画面大小发生了调整，调整的原因可能是该用户调用了 setVideoEncoderParam 或者 setSubStreamEncoderParam 重新设置了画面尺寸。

参数	描述
newHeight	视频流的高度（像素）。
newWidth	视频流的宽度（像素）。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onNetworkQuality:remoteQuality:

onNetworkQuality:remoteQuality:

- (void)onNetworkQuality:	(TRTCQualityInfo*)localQuality

remoteQuality:	(NSArray<TRTCQualityInfo*>*)remoteQuality

网络质量的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 感知到的当前网络的上行和下行质量。
SDK 会使用一组内嵌的自研算法对当前网络的延迟高低、带宽大小以及稳定情况进行评估，并计算出一个评估结果：如果评估结果为 1（Excellent） 代表当前的网络情况非常好，如果评估结果为 6（Down）代表当前网络无法支撑 TRTC 的正常通话。

参数	描述
localQuality	上行网络质量。
remoteQuality	下行网络质量，代表数据流历经“远端->云端->本端”这样一条完整的传输链路后，最终在本端统计到的数据质量。因此，这里的下行网络质量代表的是远端上行链路与本端下行链路共同影响的结果。

注意

暂时无法通过该接口单独判定远端用户的上行质量。

onStatistics:

onStatistics:

- (void)onStatistics:	(TRTCStatistics *)statistics

音视频技术指标的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 内部音频、视频以及网络相关的专业技术指标，这些信息在 TRTCStatistics 均有罗列：
视频统计信息：视频的分辨率（resolution）、帧率（FPS）和比特率（bitrate）等信息。
音频统计信息：音频的采样率（samplerate）、声道（channel）和比特率（bitrate）等信息。
网络统计信息：SDK 和云端一次往返（SDK > Cloud > SDK）的网络耗时（rtt）、丢包率（loss）、上行流量（sentBytes）和下行流量（receivedBytes）等信息。

参数	描述
statistics	统计数据，包括自己本地的统计信息和远端用户的统计信息，详情请参考 TRTCStatistics。

注意

如果您只需要获知当前网络质量的好坏，并不需要花太多时间研究本统计回调，更推荐您使用 onNetworkQuality 来解决问题。

onSpeedTestResult:

onSpeedTestResult:

- (void)onSpeedTestResult:	(TRTCSpeedTestResult *)result

网速测试的结果回调

该统计回调由 startSpeedTest: 触发。

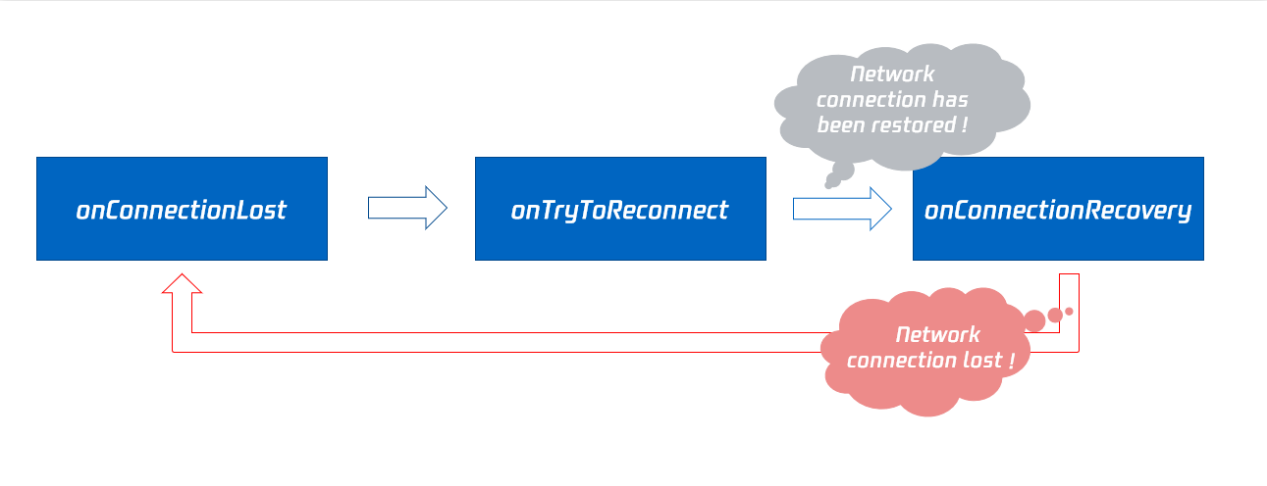
参数	描述
result	网速测试结果，包括丢包、往返延迟、上下行的带宽速率，详情请参见 TRTCSpeedTestResult。

onConnectionLost

onConnectionLost

SDK 与云端的连接已经断开

SDK 会在跟云端的连接断开时抛出此事件回调，导致断开的原因大多是网络不可用或者网络切换所致，比如用户在通话中走进电梯时就可能会遇到此事件。
在抛出此事件之后，SDK 会努力跟云端重新建立连接，重连过程中会抛出 onTryToReconnect，连接恢复后会抛出 onConnectionRecovery 。
所以，SDK 会在如下三个连接相关的事件中按如下规律切换：



onTryToReconnect

onTryToReconnect

SDK 正在尝试重新连接到云端

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出本事件，连接恢复后会抛出 onConnectionRecovery。

onConnectionRecovery

onConnectionRecovery

SDK 与云端的连接已经恢复

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出 onTryToReconnect，连接恢复后会抛出本事件回调。

onCameraDidReady

onCameraDidReady

摄像头准备就绪

当您调用 startLocalPreivew 之后，SDK 会尝试启动摄像头，如果摄像头能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问摄像头的权限，或者摄像头当前正在被其他程序独占使用中。

版权所有：腾讯云计算（北京）有限责任公司

您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onMicDidReady

onMicDidReady

麦克风准备就绪

当您调用 startLocalAudio 之后，SDK 会尝试启动麦克风，如果麦克风能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问麦克风的权限，或者麦克风当前正在被其他程序独占使用中。
您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onAudioRouteChanged:fromRoute:

onAudioRouteChanged:fromRoute:

- (void)onAudioRouteChanged:	(TRTCAudioRoute)route
fromRoute:	(TRTCAudioRoute)fromRoute

当前音频路由发生变化（仅适用于移动设备）

所谓“音频路由”，是指声音是从手机的扬声器还是从听筒中播放出来，音频路由变化也就是声音的播放位置发生了变化。
当音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
当音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。
当音频路由为有线耳机。
当音频路由为蓝牙耳机。
当音频路由为USB专业声卡设备。

参数	描述
fromRoute	变更前的音频路由。
route	音频路由，即声音由哪里输出（扬声器、听筒）。

onUserVoiceVolume:totalVolume:

onUserVoiceVolume:totalVolume:

- (void)onUserVoiceVolume:	(NSArray<TRTCVolumeInfo *> *)userVolumes

totalVolume:	(NSInteger)totalVolume

音量大小的反馈回调

SDK 可以评估每一路音频的音量大小，并每隔一段时间抛出该事件回调，您可以根据音量大小在 UI 上做出相应的提示，比如 波形图 或 音量槽 。
要完成这个功能， 您需要先调用 enableAudioVolumeEvaluation 开启这个能力并设定事件抛出的时间间隔。
需要补充说明的是，无论当前房间中是否有人说话，SDK 都会按照您设定的时间间隔定时抛出此事件回调。

参数	描述
totalVolume	所有远端用户的总音量大小，取值范围 0 - 100。
userVolumes	是一个数组，用于承载所有正在说话的用户的音量大小，取值范围 0 - 100。

注意

userVolumes 为一个数组，对于数组中的每一个元素，当 userId 为空时表示本地麦克风采集的音量大小，当 userId 不为空时代表远端用户的音量大小。

onDevice:type:stateChanged:

onDevice:type:stateChanged:

- (void)onDevice:	(NSString *)deviceId
type:	(TRTCMediaDeviceType)deviceType
stateChanged:	(NSInteger)state

本地设备的通断状态发生变化（仅适用于桌面系统）

当本地设备（包括摄像头、麦克风以及扬声器）被插入或者拔出时，SDK 便会抛出此事件回调。

参数	描述
deviceId	设备 ID。
deviceType	设备类型。
state	通断状态，0：设备断开；1：设备连接。

onAudioDeviceCaptureVolumeChanged:muted:

onAudioDeviceCaptureVolumeChanged:muted:

- (void)onAudioDeviceCaptureVolumeChanged:	(NSInteger)volume
muted:	(BOOL)muted

当前麦克风的系统采集音量发生变化

在 Mac 或 Windows 这样的桌面操作系统上，用户可以在设置中心找到声音相关的设置面板，并设置麦克风的采集音量大小。
用户将麦克风的采集音量设置得越大，麦克风采集到的声音的原始音量也就会越大，反之就会越小。
在有些型号的键盘以及笔记本电脑上，用户还可以通过按下 `禁用麦克风` 按钮（图标是一个 `话筒` 叠加了一道代表禁用的斜线）来将麦克风静音。

当用户通过系统设置界面或者通过键盘上的快捷键设定操作系统的麦克风采集音量时，SDK 便会抛出此事件。

参数	描述
muted	麦克风是否被用户禁用了：YES 被禁用，NO 被启用。
volume	系统采集音量，取值范围 0 - 100，用户可以在系统的声音设置面板上进行拖拽调整。

注意

您需要调用 enableAudioVolumeEvaluation 接口并设定 interval 大于 0 开启次事件回调，设定 interval 等于 0 关闭此事件回调。

onAudioDevicePlayoutVolumeChanged:muted:

onAudioDevicePlayoutVolumeChanged:muted:

- (void)onAudioDevicePlayoutVolumeChanged:	(NSInteger)volume
muted:	(BOOL)muted

当前系统的播放音量发生变化

在 Mac 或 Windows 这样的桌面操作系统上，用户可以在设置中心找到声音相关的设置面板，并设置系统的播放音量大小。
在有些型号的键盘以及笔记本电脑上，用户还可以通过按下“静音”按钮（图标是一个喇叭上叠加了一道代表禁用的斜线）来将系统静音。

当用户通过系统设置界面或者通过键盘上的快捷键设定操作系统的播放音量时，SDK 便会抛出此事件。

参数	描述
muted	系统是否被用户静音了：YES 被静音，NO 已恢复。
volume	系统播放音量，取值范围 0 - 100，用户可以在系统的声音设置面板上进行拖拽调整。

注意

您需要调用 enableAudioVolumeEvaluation 接口并设定 interval 大于 0 开启次事件回调，设定 interval 等于 0 关闭此事件回调。

onSystemAudioLoopbackError:

onSystemAudioLoopbackError:

- (void)onSystemAudioLoopbackError:	(TXLiteAVError)err

系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）

在 Mac 系统上，您可以通过调用 startSystemAudioLoopback 为当前系统安装一个音频驱动，并让 SDK 通过该音频驱动捕获当前 Mac 系统播放出的声音。

当用于播片教学或音乐直播中，比如老师端可以使用此功能，让 SDK 能够采集老师所播放的电影中的声音，使同房间的学生端也能听到电影中的声音。

SDK 会将系统声音采集是否被成功开启的结果，通过本事件回调抛出，需要您关注参数中的错误码。

参数	描述
err	ERR_NULL 表示成功，其余值表示失败。

onRecvCustomCmdMsgUserId:cmdID:seq:message:

onRecvCustomCmdMsgUserId:cmdID:seq:message:

- (void)onRecvCustomCmdMsgUserId:	(NSString *)userId
cmdID:	(NSInteger)cmdID
seq:	(UInt32)seq
message:	(NSData *)message

收到自定义消息的事件回调

当房间中的某个用户使用 sendCustomCmdMsg 发送自定义 UDP 消息时，房间中的其他用户可以通过 onRecvCustomCmdMsg 事件回调接收到该条消息。

参数	描述
cmdID	命令 ID。
message	消息数据。
seq	消息序号。
userId	用户标识。

onMissCustomCmdMsgUserId:cmdID:errCode:missed:

onMissCustomCmdMsgUserId:cmdID:errCode:missed:

- (void)onMissCustomCmdMsgUserId:	(NSString *)userId
cmdID:	(NSInteger)cmdID
errCode:	(NSInteger)errCode
missed:	(NSInteger)missed

自定义消息丢失的事件回调

当您使用 sendCustomCmdMsg 发送自定义 UDP 消息时，即使设置了可靠传输（reliable），也无法确 100% 不丢失，只是丢消息概率极低，能满足常规可靠性要求。
在发送端设置了可靠运输（reliable）后，SDK 都会通过此回调通知过去时间段内（通常为5s）传输途中丢失的自定义消息数量统计信息。

参数	描述
cmdID	命令 ID。
errCode	错误码。
missed	丢失的消息数量。
userId	用户标识。

注意

只有在发送端设置了可靠传输（reliable），接收方才能收到消息的丢失回调。

版权所有：腾讯云计算（北京）有限责任公司

onRecvSEIMsg:message:

onRecvSEIMsg:message:

- (void)onRecvSEIMsg:	(NSString *)userId
message:	(NSData*)message

收到 SEI 消息的回调

当房间中的某个用户使用 sendSEIMsg 借助视频数据帧发送 SEI 消息时，房间中的其他用户可以通过 onRecvSEIMsg 事件回调接收到该条消息。

参数	描述
message	数据。
userId	用户标识。

onStartPublishing:errMsg:

onStartPublishing:errMsg:

- (void)onStartPublishing:	(int)err
errMsg:	(NSString*)errMsg

开始向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 startPublishing 开始向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStopPublishing:errMsg:

onStopPublishing:errMsg:

- (void)onStopPublishing:	(int)err

errMsg:	(NSString*)errMsg

停止向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 stopPublishing 停止向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStartPublishCDNStream:errMsg:

onStartPublishCDNStream:errMsg:

- (void)onStartPublishCDNStream:	(int)err
errMsg:	(NSString *)errMsg

开始向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 startPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

注意

当您收到成功的事件回调时，只是说明您的发布指令已经同步到腾讯云后台服务器，但如果目标 CDN 厂商的服务器不接收该条视频流，依然可能导致发布失败。

onStopPublishCDNStream:errMsg:

onStopPublishCDNStream:errMsg:

- (void)onStopPublishCDNStream:	(int)err
errMsg:	(NSString *)errMsg

版权所有：腾讯云计算（北京）有限责任公司

停止向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 stopPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onSetMixTranscodingConfig:errMsg:

onSetMixTranscodingConfig:errMsg:

- (void)onSetMixTranscodingConfig:	(int)err
errMsg:	(NSString*)errMsg

设置云端混流的排版布局和转码参数的事件回调

当您调用 setMixTranscodingConfig 调整云端混流的排版布局和转码参数时，SDK 会立刻将这一调整指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	错误码：0表示成功，其余值表示失败。
errMsg	具体的错误原因。

onStartPublishMediaStream:code:message:extraInfo:

onStartPublishMediaStream:code:message:extraInfo:

- (void)onStartPublishMediaStream:	(NSString*)taskId
code:	(int)code
message:	(NSString*)message
extraInfo:	(nullable NSDictionary *)extraInfo

开始发布媒体流的事件回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	当请求成功时，TRTC 后台会在回调中提供给您这项任务的 taskId，后续您可以通过该 taskId 结合 updatePublishMediaStream 和 stopPublishMediaStream 进行更新和停止。

onUpdatePublishMediaStream:code:message:extraInfo:

onUpdatePublishMediaStream:code:message:extraInfo:

- (void)onUpdatePublishMediaStream:	(NSString*)taskId
code:	(int)code
message:	(NSString*)message
extraInfo:	(nullable NSDictionary *)extraInfo

更新媒体流的事件回调

当您调用媒体流发布接口 (updatePublishMediaStream) 开始向 TRTC 后台服务更新媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用媒体流发布接口 (updatePublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次更新请求。

onStopPublishMediaStream:code:message:extraInfo:

onStopPublishMediaStream:code:message:extraInfo:

- (void)onStopPublishMediaStream:	(NSString*)taskId
code:	(int)code
message:	(NSString*)message
extraInfo:	(nullable NSDictionary *)extraInfo

停止媒体流的事件回调

当您调用停止发布媒体流 (stopPublishMediaStream) 开始向 TRTC 后台服务停止媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用停止发布媒体流 (stopPublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次停止请求。

onCdnStreamStateChanged:status:code:msg:extraInfo:

onCdnStreamStateChanged:status:code:msg:extraInfo:

- (void)onCdnStreamStateChanged:	(NSString*)cdnUrl
status:	(int)status
code:	(int)code
msg:	(NSString*)msg
extraInfo:	(nullable NSDictionary *)info

RTMP/RTMPS 推流状态发生改变回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务。

若您在目标推流配置 (TRTCPublishTarget) 设置了向腾讯或者第三方 CDN 上发布音视频流的 URL 配置，则具体 RTMP 或者 RTMPS 推流状态将通过此回调同步给您。

参数	描述
cdnUrl	您调用 startPublishMediaStream 时通过目标推流配置 (TRTCPublishTarget) 传入的 url，在推流状态变更时，会通过此回调同步给您。
code	推流结果，0 表示成功，其余值表示出错。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体推流信息。
status	推流状态。 0：推流未开始或者已结束。在您调用 stopPublishMediaStream 时会返回该状态。 1：正在连接 TRTC 服务器和 CDN 服务器。若无法立刻成功，TRTC 后台服务会多次重试尝试推流，并返回该状态（5s回调一次）。如成功进行推流，则进入状态 2；如服务器出错或 60 秒内未成功推流，则进入状态 4。 2：CDN 推流正在进行。在成功推流后，会返回该状态。 3：TRTC 服务器和 CDN 服务器推流中断，正在恢复。当 CDN 出现异常，或推流短暂中断时，TRTC 后台服务会自动尝试恢复推流，并返回该状态（5s回调一次）。如成功恢复推流，则进入状态 2；如服务器出错或 60 秒内未成功恢复，则进入状态 4。 4：TRTC 服务器和 CDN 服务器推流中断，且恢复或连接超时。即此时推流失败，你可以再次调用 updatePublishMediaStream 尝试推流。 5：正在断开 TRTC 服务器和 CDN 服务器。在您调用 stopPublishMediaStream 时，TRTC 后台服务会依次同步状态 5 和状态 0。

onScreenCaptureStarted

onScreenCaptureStarted

屏幕分享开启的事件回调

当您通过 startScreenCapture 等相关接口启动屏幕分享时，SDK 便会抛出此事件回调。

onScreenCapturePaused:

onScreenCapturePaused:

- (void)onScreenCapturePaused:	(int)reason

屏幕分享暂停的事件回调

当您通过 pauseScreenCapture 暂停屏幕分享时，SDK 便会抛出此事件回调。

参数	描述

reason	原因。 0：用户主动暂停。 1：注意此字段的含义在 MAC 和 Windows 平台有稍微差异。屏幕窗口不可见暂停（Mac）。表示设置屏幕分享参数导致的暂停（Windows）。 2：表示屏幕分享窗口被最小化导致的暂停（仅 Windows）。 3：表示屏幕分享窗口被隐藏导致的暂停（仅 Windows）。

onScreenCaptureResumed:

onScreenCaptureResumed:

- (void)onScreenCaptureResumed:	(int)reason

屏幕分享恢复的事件回调

当您通过 resumeScreenCapture 恢复屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	恢复原因。 0：用户主动恢复。 1：注意此字段的含义在 MAC 和 Windows 平台有稍微差异。屏幕窗口恢复可见从而恢复分享（Mac）。屏幕分享参数设置完毕后自动恢复（Windows）。 2：表示屏幕分享窗口从最小化被恢复（仅 Windows）。 3：表示屏幕分享窗口从隐藏被恢复（仅 Windows）。

onScreenCaptureStoped:

onScreenCaptureStoped:

- (void)onScreenCaptureStoped:	(int)reason

屏幕分享停止的事件回调

当您通过 stopScreenCapture 停止屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	停止原因，0：用户主动停止；1：屏幕窗口关闭导致停止；2：表示屏幕分享的显示屏状态变更（如接口被拔出、投影模式变更等）。

onLocalRecordBegin:storagePath:

onLocalRecordBegin:storagePath:

- (void)onLocalRecordBegin:	(NSInteger)errCode
storagePath:	(NSString *)storagePath

本地录制任务已经开始的事件回调

当您调用 startLocalRecording 启动本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务是否已经顺利启动。

参数	描述
errCode	状态码。 0：录制任务启动成功。 -1：内部错误导致录制任务启动失败。 -2：文件后缀名有误（比如不支持的录制格式）。 -6：录制已经启动，需要先停止录制。 -7：录制文件已存在，需要先删除文件。 -8：录制目录无写入权限，请检查目录权限问题。
storagePath	录制文件存储路径。

onLocalRecording:storagePath:

onLocalRecording:storagePath:

- (void)onLocalRecording:	(NSInteger)duration
storagePath:	(NSString *)storagePath

本地录制任务正在进行中的进展事件回调

当您调用 startLocalRecording 成功启动本地媒体录制任务后，SDK 会定时地抛出本事件回调。
您可通过捕获该事件回调来获知录制任务的健康状况。
您可以在 startLocalRecording 时设定本事件回调的抛出间隔。

参数	描述
duration	已经录制的累计时长，单位毫秒。
storagePath	录制文件存储路径。

onLocalRecordFragment:

onLocalRecordFragment:

- (void)onLocalRecordFragment:	(NSString *)storagePath

本地录制分片的事件回调

当您开启分片录制时，每完成一个分片，都会通过此接口回调。

参数	描述
storagePath	分片文件存储路径。

onLocalRecordComplete:storagePath:

onLocalRecordComplete:storagePath:

- (void)onLocalRecordComplete:	(NSInteger)errCode
storagePath:	(NSString *)storagePath

本地录制任务已经结束的事件回调

当您调用 stopLocalRecording 停止本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务的最终结果。

参数	描述
errCode	状态码。 0：结束录制任务成功。 -1：录制失败。 -2：切换分辨率或横竖屏导致录制结束。 -3：录制时间太短，或未采集到任何视频或音频数据，请检查录制时长，或是否已开启音、视频采集。
storagePath	录制文件存储路径。

onUserEnter:

onUserEnter:

- (void)onUserEnter:	(NSString *)userId

有主播加入当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserEnterRoom 替代之。

onUserExit:reason:

onUserExit:reason:

- (void)onUserExit:	(NSString *)userId
reason:	(NSInteger)reason

有主播离开当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserLeaveRoom 替代之。

onAudioEffectFinished:code:

onAudioEffectFinished:code:

- (void)onAudioEffectFinished:	(int) effectId
code:	(int) code

音效播放已结束（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXAudioEffectManager 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onRenderVideoFrame:userId:streamType:

onRenderVideoFrame:userId:streamType:

- (void) onRenderVideoFrame:	(TRTCVideoFrame * _Nonnull)frame
userId:	(NSString* __nullable)userId
streamType:	(TRTCVideoStreamType)streamType

自定义视频渲染回调

当您设置了本地或者远端的视频自定义渲染回调之后，SDK 就会将原本要交给渲染控件进行渲染的视频帧通过此回调接口抛送给您，以便于您进行自定义渲染。

参数	描述
frame	待渲染的视频帧信息。
streamType	频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	视频源的 userId，如果是本地视频回调（setLocalVideoRenderDelegate），该参数可以忽略。

onGLContextCreated

onGLContextCreated

SDK 内部 OpenGL 环境已经创建的通知

onProcessVideoFrame:dstFrame:

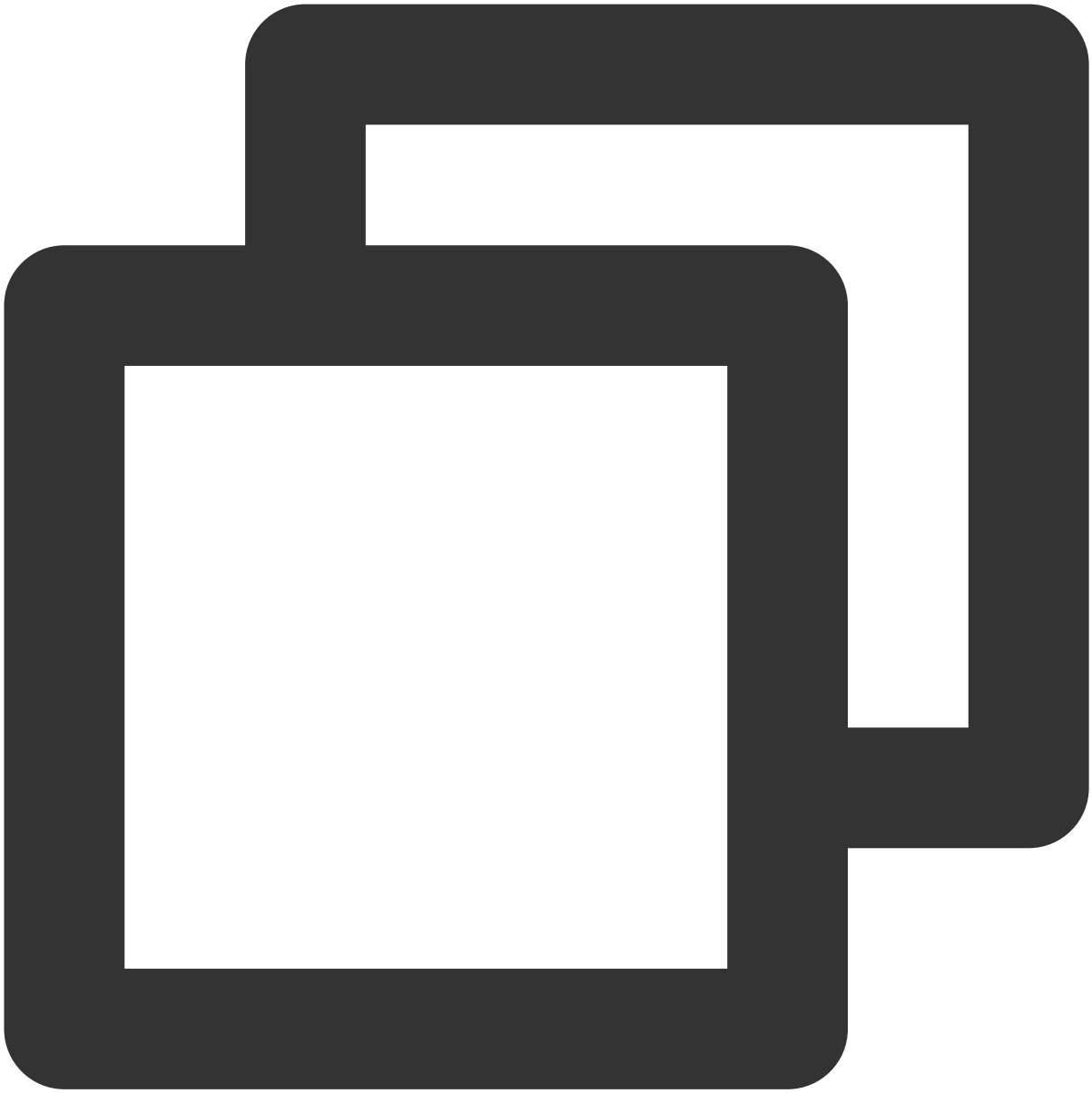
onProcessVideoFrame:dstFrame:

- (uint32_t)onProcessVideoFrame:	(TRTCVideoFrame * _Nonnull)srcFrame
dstFrame:	(TRTCVideoFrame * _Nonnull)dstFrame

用于对接第三方美颜组件的视频处理回调

如果您选购了第三方美颜组件，就需要在 TRTCCloud 中设置第三方美颜回调，之后 TRTC 就会将原本要进行预处理的视频帧通过此回调接口抛送给您。
之后您就可以将 TRTC 抛出的视频帧交给第三方美颜组件进行图像处理，由于抛出的数据是可读且可写的，因此第三方美颜的处理结果也可以同步给 TRTC 进行后续的编码和发送。

情况一：美颜组件自身会产生新的纹理。
如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函数中将 dstFrame.textureId 设置为新纹理的 ID：

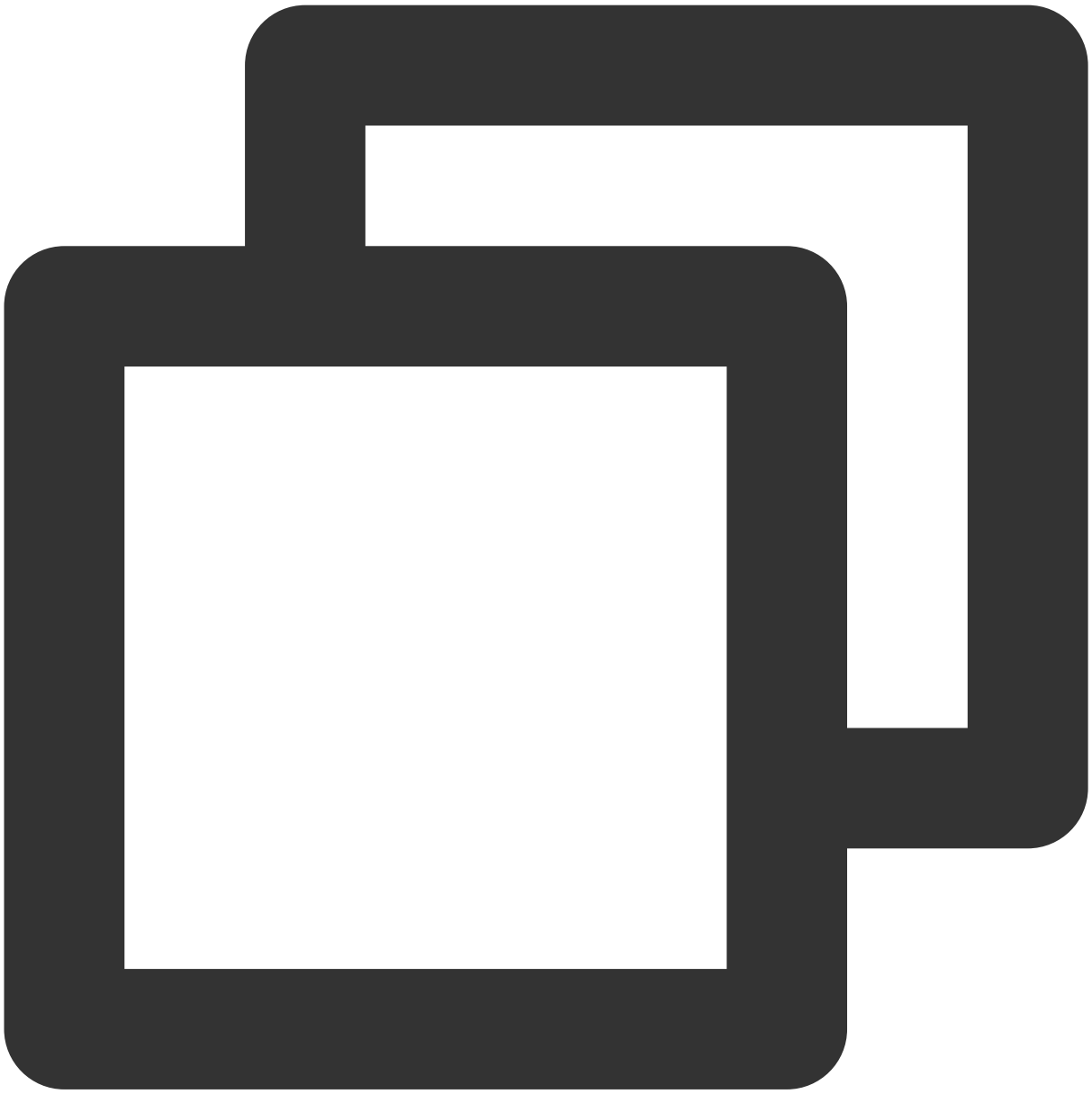


```
uint32_t onProcessVideoFrame(TRTCVideoFrame * _Nonnull)srcFrame dstFrame:(TRTCVideo
    self.frameID += 1;
    dstFrame.pixelBuffer = [[FURenderer shareRenderer] renderPixelBuffer:srcFrame.p
                                                                withFrameId:self.frame
                                                                      items:self.rende
                                                                  itemCount:self.rende

    return 0;
}
```

情况二：美颜组件需要您提供目标纹理。

如果您使用的第三方美颜模块并不生成新的纹理，而是需要您设置给该模块一个输入纹理和一个输出纹理，则可以考虑如下方案：



```
uint32_t onProcessVideoFrame(TRTCVideoFrame * _Nonnull)srcFrame dstFrame:(TRTCVideo
    thirdparty_process(srcFrame.textureId, srcFrame.width, srcFrame.height, dstFram
    return 0;
}
```

参数	描述
dstFrame	用于接收第三方美颜处理过的视频画面。

srcFrame	用于承载 TRTC 采集到的摄像头画面。

注意

目前仅支持 OpenGL 纹理方案（PC 仅支持 TRTCVideoBufferType_Buffer 格式）。

onGLContextDestory

onGLContextDestory

SDK 内部 OpenGL 环境被销毁的通知

onCapturedAudioFrame:

onCapturedAudioFrame:

- (void) onCapturedAudioFrame:	(TRTCAudioFrame *)frame

本地采集并经过音频模块前处理后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理(ANS、AEC、AGC）之后的数据，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的音频数据已经经过了前处理(ANS、AEC、AGC），但**不包含**背景音、音效、混响等前处理效果，延迟较低。

onLocalProcessedAudioFrame:

onLocalProcessedAudioFrame:

- (void) onLocalProcessedAudioFrame:	(TRTCAudioFrame *)frame

本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理、音效处理和混 BGM 之后的数据，在最终进行网络编码之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

特殊说明：

您可以通过设置接口中的 `TRTCAudioFrame.extraData` 字段，达到传输信令的目的。由于音频帧头部的数据块不能太大，建议您写入 `extraData` 时，尽量将信令控制在几个字节的大小，如果超过 100 个字节，写入的数据不会被发送。

房间内其他用户可以通过 TRTCAudioFrameDelegate 中的 `onRemoteUserAudioFrame` 中的 `TRTCAudioFrame.extraData` 字段回调接收数据。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的数据已经经过了前处理(ANS、AEC、AGC）、音效和混 BGM 处理，声音的延迟相比于 onCapturedAudioFrame 要高一些。

onRemoteUserAudioFrame:userId:

onRemoteUserAudioFrame:userId:

- (void) onRemoteUserAudioFrame:	(TRTCAudioFrame *)frame
userId:	(NSString *)userId

混音前的每一路远程用户的音频数据

当您设置完音频数据自定义回调之后，SDK 内部会把远端的每一路原始数据，在最终混音之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。
userId	用户标识。

注意

此接口回调出的音频数据是只读的，不支持修改。

onMixedPlayAudioFrame:

onMixedPlayAudioFrame:

- (void) onMixedPlayAudioFrame:	(TRTCAudioFrame *)frame

将各路待播放音频混合之后并在最终提交系统播放之前的数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把各路待播放的音频混合之后的音频数据，在提交系统播放之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。

3. 此接口回调出的是对各路待播放音频数据的混合，但其中并不包含耳返的音频数据。

onMixedAllAudioFrame:

onMixedAllAudioFrame:

- (void) onMixedAllAudioFrame:	(TRTCAudioFrame *)frame

SDK 所有音频混合后的音频数据（包括采集到的和待播放的）

当您设置完音频数据自定义回调之后，SDK 内部会把所有采集到的和待播放的音频数据混合起来，以 PCM 格式的形式通过本接口回调给您，便于您进行自定义录制。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 此接口回调出的是SDK所有音频数据的混合数据，包括：经过 3A 前处理、特效叠加以及背景音乐混音后的本地音频，所有远端音频，但不包括耳返音频。
2. 此接口回调出的音频数据不支持修改。

onVoiceEarMonitorAudioFrame:

onVoiceEarMonitorAudioFrame:

- (void) onVoiceEarMonitorAudioFrame:	(TRTCAudioFrame *)frame

耳返的音频数据

当您设置完音频数据自定义回调之后，SDK 内部会把耳返的音频数据在播放之前以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长不固定，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式 48000 采样率、单声道、16采样点位宽为例，0.02s 的音频数据字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，否则会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。

onLog:LogLevel:WhichModule:

onLog:LogLevel:WhichModule:

-(void) onLog:	(nullable NSString*)log
LogLevel:	(TRTCLogLevel)level
WhichModule:	(nullable NSString*)module

本地 LOG 的打印回调

如果您希望捕获 SDK 的本地日志打印行为，可以通过设置日志回调，让 SDK 将要打印的日志都通过本回调接口抛送给您。

参数	描述
level	日志等级请参见 TRTC_LOG_LEVEL。
log	日志内容。
module	保留字段，暂无具体意义，目前为固定值 TXLiteAVSDK。

TRTCStatistics

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 音视频统计指标（只读）

Function: TRTC SDK 会以两秒钟一次的频率向您汇报当前实时的音视频指标（帧率、码率、卡顿情况等）

TRTCStatistics

结构体类型

函数列表	描述
TRTCLocalStatistics	本地的音视频统计指标
TRTCRemoteStatistics	远端的音视频统计指标
TRTCStatistics	网络和性能的汇总统计指标

TRTCLocalStatistics

TRTCLocalStatistics

本地的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，即每秒钟新产生音频数据的多少，单位 Kbps。
audioCaptureState	【字段含义】音频设备采集状态（用于检测音频外设的健康度）。 0：采集设备状态正常；1：检测到长时间静音；2：检测到破音；3：检测到声音异常间断。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
frameRate	【字段含义】本地视频的帧率，即每秒钟会有多少视频帧，单位：FPS。
height	【字段含义】本地视频的高度，单位 px。

streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
videoBitrate	【字段含义】本地视频的码率，即每秒钟新产生视频数据的多少，单位 Kbps。
width	【字段含义】本地视频的宽度，单位 px。

TRTCRemoteStatistics

TRTCRemoteStatistics

远端的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，单位 Kbps
audioBlockRate	【字段含义】音频播放卡顿率，单位 (%) 音频播放卡顿率（audioBlockRate） = 音频播放的累计卡顿时长（audioTotalBlockTime） / 音频播放的总时长
audioPacketLoss	【字段含义】音频流的总丢包率（%）。 audioPacketLoss 代表音频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 audioPacketLoss 越小越好，丢包率为0即表示该路音频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 audioPacketLoss != 0 的情况，说明该路音频流在“云端=>观众”这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
audioTotalBlockTime	【字段含义】音频播放的累计卡顿时长，单位 ms
finalLoss	【字段含义】该路音视频流的总丢包率（%）。已废弃，不推荐使用；建议使用 audioPacketLoss、videoPacketLoss 替代。
frameRate	【字段含义】远端视频的帧率，单位：FPS。
height	【字段含义】远端视频的高度，单位 px。
jitterBufferDelay	【字段含义】播放延迟，单位 ms 为了避免网络抖动和网络包乱序导致的声音和画面卡顿，TRTC 会在播放端管理一个播放缓冲区，用于对接收到的网络数据包进行整理，该缓冲区的大小会根据当前的网络质量进行自适应调整，该缓冲区的大小折算成以毫秒为单位的时间长度，也就是 jitterBufferDelay。

版权所有：腾讯云计算（北京）有限责任公司

point2PointDelay	【字段含义】端到端延迟，单位 ms point2PointDelay 代表 “主播=>云端=>观众” 的延迟，更准确地说，它代表了“采集=>编码=>网络传输=>接收=>缓冲=>解码=>播放” 全链路的延迟。 point2PointDelay 需要本地和远端的 SDK 均为 8.5 及以上的版本才生效，若远端用户为 8.5 以前的版本，此数值会一直为0，代表无意义。
remoteNetworkRTT	【字段含义】该用户从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。 该数值越小越好：如果 remoteNetworkRTT < 50ms，意味着较低的音视频通话延迟；如果 remoteNetworkRTT > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，remoteNetworkRTT 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 remoteNetworkUpRTT 和 remoteNetworkDownRTT
remoteNetworkUplinkLoss	【字段含义】该用户从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 remoteNetworkUplinkLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 remoteNetworkUplinkLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。
streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
userId	【字段含义】用户 ID
videoBitrate	【字段含义】远端视频的码率，单位 Kbps。
videoBlockRate	【字段含义】视频播放卡顿率，单位 (%) 视频播放卡顿率（videoBlockRate） = 视频播放的累计卡顿时长（videoTotalBlockTime） / 视频播放的总时长。
videoPacketLoss	【字段含义】该路视频流的总丢包率（%）。 videoPacketLoss 代表该路视频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 videoPacketLoss 越小越好，丢包率为0即表示该路视频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 videoPacketLoss != 0 的情况，说明该路视频流在 `云端>观众` 这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
videoTotalBlockTime	【字段含义】视频播放的累计卡顿时长，单位 ms
width	【字段含义】远端视频的宽度，单位 px。

版权所有：腾讯云计算（北京）有限责任公司

TRTCStatistics

TRTCStatistics

网络和性能的汇总统计指标

枚举类型	描述
appCpu	【字段含义】当前应用的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。
downLoss	【字段含义】从云端到 SDK 的下行丢包率，单位 (%) 该数值越小越好，如果 downLoss 为 0%，则意味着下行链路的网络质量很好，从云端接收的数据包基本不发生丢失。 如果 downLoss 为 30%，则意味着云端向 SDK 传输的音视频数据包中，会有 30% 丢失在传输链路中。
gatewayRtt	【字段含义】从 SDK 到本地路由器的往返时延，单位 ms 该数值代表从 SDK 发送一个网络包到本地路由器网关，再从网关回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 【SDK>网关>SDK【 的总耗时。 该数值越小越好：如果 gatewayRtt < 50ms，意味着较低的音视频通话延迟；如果 gatewayRtt > 200ms，则意味着较高的音视频通话延迟。 当网络类型为蜂窝网时，该值无效。
localStatistics	【字段含义】本地的音视频统计信息 由于本地可能有三路音视频流（即高清大画面，低清小画面，以及辅流画面），因此本地的音视频统计信息是一个数组。
receivedBytes	【字段含义】总接收字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
remoteStatistics	【字段含义】远端的音视频统计信息 因为同时可能有多个远端用户，而且每个远端用户同时可能有多路音视频流（即高清大画面，低清小画面，以及辅流画面），因此远端的音视频统计信息是一个数组。
rtt	【字段含义】从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。 该数值越小越好：如果 rtt < 50ms，意味着较低的音视频通话延迟；如果 rtt > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，rtt 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 upRtt 和 downRtt。
sentBytes	【字段含义】总发送字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
systemCpu	【字段含义】当前系统的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。

版权所有：腾讯云计算（北京）有限责任公司

upLoss	【字段含义】从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 upLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 upLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。

TXAudioEffectManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 背景音乐、短音效和人声特效的管理类

Function: 用于对背景音乐、短音效和人声特效进行设置的管理类

TXAudioEffectManager

TXAudioEffectManager

函数列表	描述
enableVoiceEarMonitor:	开启耳返
setVoiceEarMonitorVolume:	设置耳返音量
setVoiceReverbType:	设置人声的混响效果
setVoiceChangerType:	设置人声的变声特效
setVoiceVolume:	设置语音音量
setVoicePitch:	设置语音音调
startPlayMusic:onStart:onProgress:onComplete:	开始播放背景音乐
stopPlayMusic:	停止播放背景音乐
pausePlayMusic:	暂停播放背景音乐
resumePlayMusic:	恢复播放背景音乐
setAllMusicVolume:	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume:volume:	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume:volume:	设置某一首背景音乐的本地音量的大小
setMusicPitch:pitch:	调整背景音乐的音调高低

setMusicSpeedRate:speedRate:	调整背景音乐的变速效果
getMusicCurrentPosInMS:	获取背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS:	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInMS:pts:	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate:speedRate:	调整搓碟的变速效果
preloadMusic:onProgress:onError:	预加载背景音乐
getMusicTrackCount:	获取背景音乐的音轨数量
setMusicTrack:track:	指定背景音乐的播放音轨

结构体类型

函数列表	描述
TXAudioMusicParam	背景音乐的播放控制信息

枚举类型

枚举类型	描述
TXVoiceReverbType	混响特效
TXVoiceChangeType	变声特效

enableVoiceEarMonitor:

enableVoiceEarMonitor:

- (void)enableVoiceEarMonitor:	(BOOL)enable

开启耳返

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

需要您注意的是，由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。
同时也需要注意，并非所有的手机开启此特效后都能达到优秀的耳返效果，我们已经对部分耳返效果不佳的手机屏蔽了该特效。

参数	描述
enable	YES：开启；NO：关闭。

注意
仅在主播佩戴耳机时才能开启此特效，同时请您提示主播佩戴有线耳机。

setVoiceEarMonitorVolume:

setVoiceEarMonitorVolume:

- (void)setVoiceEarMonitorVolume:	(NSInteger)volume

设置耳返音量

通过该接口您可以设置耳返特效中声音的音量大小。

参数	描述
volume	音量大小，取值范围为 0 - 100，默认值：100。

注意
如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoiceReverbType:

setVoiceReverbType:

- (void)setVoiceReverbType:	(TXVoiceReverbType)reverbType

设置人声的混响效果

通过该接口您可以设置人声的混响效果，具体特效请参见枚举定义 TXVoiceReverbType。
注意
设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceChangerType:

setVoiceChangerType:

- (void)setVoiceChangerType:	(TXVoiceChangeType)changerType

设置人声的变声特效

通过该接口您可以设置人声的变声特效，具体特效请参见枚举定义 TXVoiceChangeType。

注意

设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceVolume:

setVoiceVolume:

- (void)setVoiceVolume:	(NSInteger)volume

设置语音音量

该接口可以设置语音音量的大小，一般配合音乐音量的设置接口 setAllMusicVolume 协同使用，用于调谐语音和音乐在混音前各自的音量占比。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoicePitch:

setVoicePitch:

-(void)setVoicePitch:	(double)pitch

设置语音音调

该接口可以设置语音音调，用于实现变调不变速的目的。

参数	描述

pitch	音调，取值范围为-1.0f~1.0f，默认值：0.0f。

startPlayMusic:onStart:onProgress:onComplete:

startPlayMusic:onStart:onProgress:onComplete:

- (void)startPlayMusic:	(TXAudioMusicParam *)musicParam
onStart:	(TXAudioMusicStartBlock _Nullable)startBlock
onProgress:	(TXAudioMusicProgressBlock _Nullable)progressBlock
onComplete:	(TXAudioMusicCompleteBlock _Nullable)completeBlock

开始播放背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
completeBlock	播放结束回调。
musicParam	音乐参数。
progressBlock	播放进度回调。
startBlock	播放开始回调。

注意

1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

stopPlayMusic:

stopPlayMusic:

- (void)stopPlayMusic:	(int32_t)id

停止播放背景音乐

参数	描述

id	音乐 ID。

pausePlayMusic:

pausePlayMusic:

- (void)pausePlayMusic:	(int32_t)id

暂停播放背景音乐

参数	描述
id	音乐 ID。

resumePlayMusic:

resumePlayMusic:

- (void)resumePlayMusic:	(int32_t)id

恢复播放背景音乐

参数	描述
id	音乐 ID。

setAllMusicVolume:

setAllMusicVolume:

- (void)setAllMusicVolume:	(NSInteger)volume

设置所有背景音乐的本地音量和远端音量的大小

该接口可以设置所有背景音乐的本地音量和远端音量。
本地音量：即主播本地可以听到的背景音乐的音量大小。
远端音量：即观众端可以听到的背景音乐的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPublishVolume:volume:

setMusicPublishVolume:volume:

- (void)setMusicPublishVolume:	(int32_t)id
volume:	(NSInteger)volume

设置某一首背景音乐的远端音量的大小

该接口可以细粒度地控制每一首背景音乐的远端音量，也就是观众端可听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。
volume	音量大小，取值范围为0 - 100；默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPlayoutVolume:volume:

setMusicPlayoutVolume:volume:

- (void)setMusicPlayoutVolume:	(int32_t)id
volume:	(NSInteger)volume

设置某一首背景音乐的本地音量的大小

该接口可以细粒度地控制每一首背景音乐的本地音量，也就是主播本地可以听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。

volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPitch:pitch:

setMusicPitch:pitch:

- (void)setMusicPitch:	(int32_t)id
pitch:	(double)pitch

调整背景音乐的音调高低

参数	描述
id	音乐 ID。
pitch	音调，默认值是0.0f，范围是：[-1 ~ 1] 之间的浮点数。

setMusicSpeedRate:speedRate:

setMusicSpeedRate:speedRate:

- (void)setMusicSpeedRate:	(int32_t)id
speedRate:	(double)speedRate

调整背景音乐的变速效果

参数	描述
id	音乐 ID。
speedRate	速度，默认值是1.0f，范围是：[0.5 ~ 2] 之间的浮点数。

getMusicCurrentPosInMS:

getMusicCurrentPosInMS:

- (NSInteger)getMusicCurrentPosInMS:	(int32_t)id

获取背景音乐的播放进度（单位：毫秒）

参数	描述
id	音乐 ID。

返回值说明：

成功返回当前播放时间，单位：毫秒，失败返回 -1。

getMusicDurationInMS:

getMusicDurationInMS:

- (NSInteger)getMusicDurationInMS:	(NSString *)path

获取背景音乐的总时长（单位：毫秒）

参数	描述
path	音乐文件路径。

返回值说明：

成功返回时长，失败返回 -1。

seekMusicToPosInMS:pts:

seekMusicToPosInMS:pts:

- (void)seekMusicToPosInMS:	(int32_t)id
pts:	(NSInteger)pts

设置背景音乐的播放进度（单位：毫秒）

参数	描述
id	音乐 ID。

pts	单位: 毫秒。

注意

请尽量避免过度频繁地调用该接口，因为该接口可能会再次读写音乐文件，耗时稍高。

因此，当用户拖拽音乐的播放进度条时，请在用户完成拖拽操作后再调用本接口。

因为 UI 上的进度条控件往往会以很高的频率反馈用户的拖拽进度，如不做频率限制，会导致较差的用户体验。

setMusicScratchSpeedRate:speedRate:

setMusicScratchSpeedRate:speedRate:

- (void)setMusicScratchSpeedRate:	(int32_t)id
speedRate:	(double)scratchSpeedRate

调整搓碟的变速效果

参数	描述
id	音乐 ID。
scratchSpeedRate	搓碟速度，默认值是1.0f，范围是：[-12.0 ~ 12.0] 之间的浮点数, 速度值正/负表示方向正/反，绝对值大小表示速度快慢。

注意

前置条件 preloadMusic 成功。

preloadMusic:onProgress:onError:

preloadMusic:onProgress:onError:

- (void)preloadMusic:	(TXAudioMusicParam *)preloadParam
onProgress:	(TXMusicPreloadProgressBlock _Nullable)progressBlock
onError:	(TXMusicPreloadErrorBlock _Nullable)errorBlock

预加载背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
preloadParam	预加载音乐参数。

注意

1. 预先加载最多同时支持2个不同 ID 的预加载，且预加载时长不超过10分钟，使用完需 stopPlayMusic，否则内存不释放。
2. 若该ID对应的音乐正在播放中，预加载会失败，需先调用 stopPlayMusic。
3. 当 musicParam 和传入 startPlayMusic 的 musicParam 完全相同时，预加载有效。

getMusicTrackCount:

getMusicTrackCount:

- (NSInteger)getMusicTrackCount:	(int32_t)id

获取背景音乐的音轨数量

参数	描述
id	音乐 ID。

setMusicTrack:track:

setMusicTrack:track:

- (void)setMusicTrack:	(int32_t)id
track:	(NSInteger)track

指定背景音乐的播放音轨

参数	描述
id	音乐 ID。
index	默认播放第一个音轨。取值范围[0, 音轨总数)。

注意

音轨总数量可通过 getMusicTrackCount 接口获取。

TXVoiceReverbType

TXVoiceReverbType

混响特效

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受，目前支持如下几种混响效果：
0：关闭；1：KTV；2：小房间；3：大会堂；4：低沉；5：洪亮；6：金属声；7：磁性；8：空灵；9：录音棚；10：悠扬 11：录音棚2。

枚举	取值	描述
TXVoiceReverbType_0	0	关闭特效
TXVoiceReverbType_1	1	KTV
TXVoiceReverbType_2	2	小房间
TXVoiceReverbType_3	3	大会堂
TXVoiceReverbType_4	4	低沉
TXVoiceReverbType_5	5	洪亮
TXVoiceReverbType_6	6	金属声
TXVoiceReverbType_7	7	磁性
TXVoiceReverbType_8	8	空灵
TXVoiceReverbType_9	9	录音棚
TXVoiceReverbType_10	10	悠扬
TXVoiceReverbType_11	11	录音棚2

TXVoiceChangeType

TXVoiceChangeType

变声特效

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色，目前支持如下几种变声特效：

版权所有：腾讯云计算（北京）有限责任公司

0：关闭；1：熊孩子；2：萝莉；3：大叔；4：重金属；5：感冒；6：外语腔；7：困兽；8：肥宅；9：强电流；10：重机械；11：空灵。

枚举	取值	描述
TXVoiceChangeType_0	0	关闭
TXVoiceChangeType_1	1	熊孩子
TXVoiceChangeType_2	2	萝莉
TXVoiceChangeType_3	3	大叔
TXVoiceChangeType_4	4	重金属
TXVoiceChangeType_5	5	感冒
TXVoiceChangeType_6	6	外语腔
TXVoiceChangeType_7	7	困兽
TXVoiceChangeType_8	8	肥宅
TXVoiceChangeType_9	9	强电流
TXVoiceChangeType_10	10	重机械
TXVoiceChangeType_11	11	空灵

TXAudioMusicParam

TXAudioMusicParam

背景音乐的播放控制信息

该信息用于在接口 startPlayMusic 中指定背景音乐的相关信息，包括播放 ID、文件路径和循环次数等：

1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

枚举类型	描述
ID	【字段含义】音乐 ID。 【特殊说明】SDK 允许播放多路音乐，因此需要使用 ID 进行标记，用于控制音乐的开始、停止、音量等。
endTimeMS	【字段含义】音乐结束播放时间点，单位毫秒，0表示播放至文件结尾。

isShortFile	【字段含义】播放的是否为短音乐文件。 【推荐取值】YES：需要重复播放的短音乐文件；NO：正常的音乐文件。默认值：NO。
loopCount	【字段含义】音乐循环播放的次数。 【推荐取值】取值范围为0 - 任意正整数，默认值：0。0 表示播放音乐一次；1 表示播放音乐两次；以此类推。
path	【字段含义】音效文件的完整路径或 URL 地址。支持的音频格式包括 MP3、AAC、M4A、WAV。
publish	【字段含义】是否将音乐传到远端。 【推荐取值】YES：音乐在本地播放的同时，远端用户也能听到该音乐；NO：主播只能在本地听到该音乐，远端观众听不到。默认值：NO。
startTimeMS	【字段含义】音乐开始播放时间点，单位：毫秒。

版权所有：腾讯云计算（北京）有限责任公司

TXBeautyManager

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: 美颜与图像处理参数设置类

Function: 修改美颜、滤镜、绿幕等参数

TXBeautyManager

TXBeautyManager

函数列表	描述
setBeautyStyle:	设置美颜（磨皮）算法
setBeautyLevel:	设置美颜级别
setWhitenessLevel:	设置美白级别
enableSharpnessEnhancement:	开启清晰度增强
setRuddyLevel:	设置红润级别
setFilter:	设置色彩滤镜效果
setFilterStrength:	设置色彩滤镜的强度
setGreenScreenFile:	设置绿幕背景视频
setEyeScaleLevel:	设置大眼级别
setFaceSlimLevel:	设置瘦脸级别
setFaceVLevel:	设置 V 脸级别
setChinLevel:	设置下巴拉伸或收缩
setFaceShortLevel:	设置短脸级别
setFaceNarrowLevel:	设置窄脸级别

setNoseSlimLevel:	设置瘦鼻级别
setEyeLightenLevel:	设置亮眼级别
setToothWhitenLevel:	设置牙齿美白级别
setWrinkleRemoveLevel:	设置祛皱级别
setPounchRemoveLevel:	设置祛眼袋级别
setSmileLinesRemoveLevel:	设置法令纹去除级别
setForeheadLevel:	设置发际线调整级别
setEyeDistanceLevel:	设置眼距
setEyeAngleLevel:	设置眼角调整级别
setMouthShapeLevel:	设置嘴型调整级别
setNoseWingLevel:	设置鼻翼调整级别
setNosePositionLevel:	设置鼻子位置
setLipsThicknessLevel:	设置嘴唇厚度
setFaceBeautyLevel:	设置脸型
setMotionTmpl:inDir:	选择 AI 动效挂件
setMotionMute:	是否在动效素材播放时静音

枚举类型

枚举类型	描述
TXBeautyStyle	美颜（磨皮）算法

setBeautyStyle:

setBeautyStyle:

- (void)setBeautyStyle:	(TXBeautyStyle)beautyStyle

设置美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案：

参数	描述
beautyStyle	美颜风格，TXBeautyStyleSmooth：光滑；TXBeautyStyleNature：自然；TXBeautyStylePitu：优图。

setBeautyLevel:

setBeautyLevel:

- (void)setBeautyLevel:	(float)beautyLevel

设置美颜级别

参数	描述
beautyLevel	美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显。

setWhitenessLevel:

setWhitenessLevel:

- (void)setWhitenessLevel:	(float)whitenessLevel

设置美白级别

参数	描述
whitenessLevel	美白级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

enableSharpnessEnhancement:

enableSharpnessEnhancement:

- (void)enableSharpnessEnhancement:	(BOOL)enable

开启清晰度增强

setRuddyLevel:

setRuddyLevel:

- (void)setRuddyLevel:	(float)ruddyLevel

设置红润级别

参数	描述
ruddyLevel	红润级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

setFilter:

setFilter:

- (void)setFilter:	(nullable TXImage *)image

设置色彩滤镜效果

色彩滤镜，是一副包含色彩映射关系的颜色查找表图片，您可以在我们提供的官方 Demo 中找到预先准备好的几张滤镜图片。
SDK 会根据该查找表中的映射关系，对摄像头采集出的原始视频画面进行二次处理，以达到预期的滤镜效果。

参数	描述
image	包含色彩映射关系的颜色查找表图片，必须是 png 格式。

setFilterStrength:

setFilterStrength:

- (void)setFilterStrength:	(float)strength

设置色彩滤镜的强度

该数值越高，色彩滤镜的作用强度越明显，经过滤镜处理后的视频画面跟原画面的颜色差异越大。
我默认的滤镜浓度是 0.5，如果您觉得默认的滤镜效果不明显，可以设置为 0.5 以上的数字，最大值为 1。

参数	描述
strength	从 0 到 1，数值越大滤镜效果越明显，默认值为 0.5。

版权所有：腾讯云计算（北京）有限责任公司

setGreenScreenFile:

setGreenScreenFile:

- (int)setGreenScreenFile:	(nullable NSString *)path

设置绿幕背景视频

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。此接口所开启的绿幕功能不具备智能去除背景的能力，需要被拍摄者的背后有一块绿色的幕布来辅助产生特效。

参数	描述
path	MP4格式的视频文件路径; 设置空值表示关闭特效。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeScaleLevel:

setEyeScaleLevel:

- (int)setEyeScaleLevel:	(float)eyeScaleLevel

设置大眼级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeScaleLevel	大眼级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceSlimLevel:

setFaceSlimLevel:

- (int)setFaceSlimLevel:	(float)faceSlimLevel

设置瘦脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceSlimLevel	瘦脸级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceVLevel:

setFaceVLevel:

- (int)setFaceVLevel:	(float)faceVLevel

设置 V 脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceVLevel	V 脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setChinLevel:

setChinLevel:

- (int)setChinLevel:	(float)chinLevel

设置下巴拉伸或收缩

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
chinLevel	下巴拉伸或收缩级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示收缩，大于 0 表示拉伸。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceShortLevel:

setFaceShortLevel:

- (int)setFaceShortLevel:	(float)faceShortLevel

设置短脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceShortLevel	短脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceNarrowLevel:

setFaceNarrowLevel:

- (int)setFaceNarrowLevel:	(float)faceNarrowLevel

设置窄脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
level	窄脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNoseSlimLevel:

setNoseSlimLevel:

- (int)setNoseSlimLevel:	(float)noseSlimLevel

设置瘦鼻级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
noseSlimLevel	瘦鼻级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeLightenLevel:

setEyeLightenLevel:

- (int)setEyeLightenLevel:	(float)eyeLightenLevel

设置亮眼级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeLightenLevel	亮眼级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setToothWhitenLevel:

setToothWhitenLevel:

- (int)setToothWhitenLevel:	(float)toothWhitenLevel

设置牙齿美白级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
toothWhitenLevel	白牙级别，取值范围 0 - 9；0表示关闭，9 表示效果最明显。

版权所有：腾讯云计算（北京）有限责任公司

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setWrinkleRemoveLevel:

setWrinkleRemoveLevel:

- (int)setWrinkleRemoveLevel:	(float)wrinkleRemoveLevel

设置祛皱级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
wrinkleRemoveLevel	祛皱级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setPounchRemoveLevel:

setPounchRemoveLevel:

- (int)setPounchRemoveLevel:	(float)pounchRemoveLevel

设置祛眼袋级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
pounchRemoveLevel	祛眼袋级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setSmileLinesRemoveLevel:

setSmileLinesRemoveLevel:

- (int)setSmileLinesRemoveLevel:	(float)smileLinesRemoveLevel

设置法令纹去除级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
smileLinesRemoveLevel	法令纹级别，取值范围 0 - 9；0表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setForeheadLevel:

setForeheadLevel:

- (int)setForeheadLevel:	(float)foreheadLevel

设置发际线调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
foreheadLevel	发际线级别，取值范围-9 - 9；0表示关闭，9表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeDistanceLevel:

setEyeDistanceLevel:

- (int)setEyeDistanceLevel:	(float)eyeDistanceLevel

设置眼距

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述

eyeDistanceLevel	眼距级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeAngleLevel:

setEyeAngleLevel:

- (int)setEyeAngleLevel:	(float)eyeAngleLevel

设置眼角调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeAngleLevel	眼角调整级别，取值范围-9 - 9；0表示关闭，9表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setMouthShapeLevel:

setMouthShapeLevel:

- (int)setMouthShapeLevel:	(float)mouthShapeLevel

设置嘴型调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
mouthShapeLevel	嘴型级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNoseWingLevel:

setNoseWingLevel:

- (int)setNoseWingLevel:	(float)noseWingLevel

设置鼻翼调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
noseWingLevel	鼻翼调整级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNosePositionLevel:

setNosePositionLevel:

- (int)setNosePositionLevel:	(float)nosePositionLevel

设置鼻子位置

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
nosePositionLevel	鼻子位置级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示抬高，大于 0 表示降低。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setLipsThicknessLevel:

setLipsThicknessLevel:

- (int)setLipsThicknessLevel:	(float)lipsThicknessLevel

设置嘴唇厚度

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
lipsThicknessLevel	嘴唇厚度级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceBeautyLevel:

setFaceBeautyLevel:

- (int)setFaceBeautyLevel:	(float)faceBeautyLevel

设置脸型

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceBeautyLevel	美型级别，取值范围 0 - 9；0 表示关闭，1 - 9 值越大，效果越明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setMotionTmpl:inDir:

setMotionTmpl:inDir:

- (void)setMotionTmpl:	(nullable NSString *)tmplName
inDir:	(nullable NSString *)tmplDir

选择 AI 动效挂件

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述

tmplDir	动效素材文件所在目录。
tmplName	动效挂件名称。

setMotionMute:

setMotionMute:

- (void)setMotionMute:	(BOOL)motionMute

是否在动效素材播放时静音

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。
有些挂件本身会有声音特效，通过此 API 可以关闭这些特效播放时所带的声音效果。

参数	描述
motionMute	YES：静音；NO：不静音。

TXBeautyStyle

TXBeautyStyle

美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案。

枚举	取值	描述
TXBeautyStyleSmooth	0	光滑，算法比较激进，磨皮效果比较明显，适用于秀场直播。
TXBeautyStyleNature	1	自然，算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。
TXBeautyStylePitu	2	优图，由优图实验室提供，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高。

TXDeviceManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 音视频设备管理模块

Function: 用于管理摄像头、麦克风和扬声器等音视频相关的硬件设备

TXDeviceManager

TXDeviceObserver

函数列表	描述
onDeviceChanged:type:state:	本地设备的通断状态发生变化（仅适用于桌面系统）

TXDeviceManager

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera:	切换前置或后置摄像头（仅适用于移动端）
isCameraZoomSupported	查询当前摄像头是否支持缩放（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio:	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus:	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition:	设置摄像头的对焦位置（仅适用于移动端）
isCameraTorchSupported	查询是否支持开启闪光灯（仅适用于移动端）
enableCameraTorch:	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动

	端）
setAudioRoute:	设置音频路由（仅适用于移动端）
setExposureCompensation:	设置摄像头的曝光参数，取值范围从-1到1
getDevicesList:	获取设备列表（仅适用于桌面端）
setCurrentDevice:deviceId:	设置当前要使用的设备（仅适用于桌面端）
getCurrentDevice:	获取当前正在使用的设备（仅适用于桌面端）
setCurrentDeviceVolume:deviceType:	设置当前设备的音量（仅适用于桌面端）
getCurrentDeviceVolume:	获取当前设备的音量（仅适用于桌面端）
setCurrentDeviceMute:deviceType:	设置当前设备的静音状态（仅适用于桌面端）
getCurrentDeviceMute:	获取当前设备的静音状态（仅适用于桌面端）
enableFollowingDefaultAudioDevice:enable:	设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）
startCameraDeviceTest:	开始摄像头测试（仅适用于桌面端）
stopCameraDeviceTest	结束摄像头测试（仅适用于桌面端）
startMicDeviceTest:	开始麦克风测试（仅适用于桌面端）
startMicDeviceTest:playback:	开始麦克风测试（仅适用于桌面端）
stopMicDeviceTest	结束麦克风测试（仅适用于桌面端）
startSpeakerDeviceTest:	开始扬声器测试（仅适用于桌面端）
stopSpeakerDeviceTest	结束扬声器测试（仅适用于桌面端）
setObserver:	设备热插拔回调（仅适用于 Mac 系统）
setCameraCapturerParam:	设置摄像头采集偏好
setSystemVolumeType:	设置系统音量类型（仅适用于移动端）

结构体类型

函数列表	描述

TXCameraCaptureParam	摄像头采集参数
TXMediaDeviceInfo	音视频设备的相关信息（仅适用于桌面平台）

枚举类型

枚举类型	描述
TXSystemVolumeType	系统音量类型（仅适用于移动设备）
TXAudioRoute	音频路由（即声音的播放模式）
TXMediaDeviceType	设备类型（仅适用于桌面平台）
TXMediaDeviceState	设备操作
TXCameraCaptureMode	摄像头采集偏好

onDeviceChanged:type:state:

onDeviceChanged:type:state:

- (void)onDeviceChanged:	(NSString*)deviceId
type:	(TXMediaDeviceType)mediaType
state:	(TXMediaDeviceState)mediaState

本地设备的通断状态发生变化（仅适用于桌面系统）

当本地设备（包括摄像头、麦克风以及扬声器）被插入或者拔出时，SDK 便会抛出此事件回调。

参数	描述
deviceId	设备 ID
state	通断状态，0：设备已添加；1：设备已被移除；2：设备已启用。
type	设备类型

isFrontCamera

isFrontCamera

判断当前是否为前置摄像头（仅适用于移动端）

switchCamera:

switchCamera:

- (NSInteger)switchCamera:	(BOOL)frontCamera

切换前置或后置摄像头（仅适用于移动端）

isCameraZoomSupported

isCameraZoomSupported

查询当前摄像头是否支持缩放（仅适用于移动端）

getCameraZoomMaxRatio

getCameraZoomMaxRatio

获取摄像头的最大缩放倍数（仅适用于移动端）

setCameraZoomRatio:

setCameraZoomRatio:

- (NSInteger)setCameraZoomRatio:	(CGFloat)zoomRatio

设置摄像头的缩放倍数（仅适用于移动端）

参数	描述
zoomRatio	取值范围1 - 5，取值为1表示最远视角（正常镜头），取值为5表示最近视角（放大镜头）。最大值推荐为5，若超过5，视频数据会变得模糊不清。

isAutoFocusEnabled

isAutoFocusEnabled

查询是否支持自动识别人脸位置（仅适用于移动端）

enableCameraAutoFocus:

enableCameraAutoFocus:

- (NSInteger)enableCameraAutoFocus:	(BOOL)enabled

开启自动对焦功能（仅适用于移动端）

开启后，SDK 会自动检测画面中的人脸位置，并将摄像头的焦点始终对焦在人脸位置上。

setCameraFocusPosition:

setCameraFocusPosition:

- (NSInteger)setCameraFocusPosition:	(CGPoint)position

设置摄像头的对焦位置（仅适用于移动端）

您可以通过该接口实现如下交互：
1. 在本地摄像头的预览画面上，允许用户单击操作。
2. 在用户的单击位置显示一个矩形方框，以示摄像头会在此处对焦。
3. 随后将用户点击位置的坐标通过本接口传递给 SDK，之后 SDK 会操控摄像头按照用户期望的位置进行对焦。

参数	描述
position	对焦位置，请传入期望对焦点的坐标值

注意

使用该接口的前提是先通过 enableCameraAutoFocus 关闭自动对焦功能。

返回值说明：

0：操作成功；负数：操作失败。

isCameraTorchSupported

isCameraTorchSupported

查询是否支持开启闪光灯（仅适用于移动端）

enableCameraTorch:

enableCameraTorch:

- (NSInteger)enableCameraTorch:	(BOOL)enabled

开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）

setAudioRoute:

setAudioRoute:

- (NSInteger)setAudioRoute:	(TXAudioRoute)route

设置音频路由（仅适用于移动端）

手机有两个音频播放设备：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

setExposureCompensation:

setExposureCompensation:

- (NSInteger)setExposureCompensation:	(CGFloat)value

设置摄像头的曝光参数，取值范围从-1到1

getDevicesList:

getDevicesList:

- (NSArray<TXMediaDeviceInfo *> * _Nullable)getDevicesList:	(TXMediaDeviceType)type

获取设备列表（仅适用于桌面端）

参数	描述
type	设备类型，指定需要获取哪种设备的列表。详见 TXMediaDeviceType 定义。

注意

使用完毕后请调用 release 方法释放资源，这样可以让 SDK 维护 ITXDeviceCollection 对象的生命周期。
不要使用 delete 释放返回的 Collection 对象，delete ITXDeviceCollection* 指针会导致异常崩溃。
type 只支持 TXMediaDeviceTypeMic、TXMediaDeviceTypeSpeaker、TXMediaDeviceTypeCamera。
此接口只支持 Mac 和 Windows 平台。

setCurrentDevice:deviceId:

setCurrentDevice:deviceId:

- (NSInteger)setCurrentDevice:	(TXMediaDeviceType)type
deviceId:	(NSString *)deviceId

设置当前要使用的设备（仅适用于桌面端）

参数	描述
deviceId	设备ID，您可以通过接口 getDevicesList 获得设备 ID。
type	设备类型，详见 TXMediaDeviceType 定义。

返回值说明：

0：操作成功；负数：操作失败。

getCurrentDevice:

getCurrentDevice:

- (TXMediaDeviceInfo * _Nullable)getCurrentDevice:	(TXMediaDeviceType)type

获取当前正在使用的设备（仅适用于桌面端）

setCurrentDeviceVolume:deviceType:

setCurrentDeviceVolume:deviceType:

- (NSInteger)setCurrentDeviceVolume:	(NSInteger)volume
deviceType:	(TXMediaDeviceType)type

设置当前设备的音量（仅适用于桌面端）

这里的音量指的是麦克风的采集音量或者扬声器的播放音量，摄像头是不支持设置音量的。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

getCurrentDeviceVolume:

getCurrentDeviceVolume:

- (NSInteger)getCurrentDeviceVolume:	(TXMediaDeviceType)type

获取当前设备的音量（仅适用于桌面端）

这里的音量指的是麦克风的采集音量或者扬声器的播放音量，摄像头是不支持获取音量的。

setCurrentDeviceMute:deviceType:

setCurrentDeviceMute:deviceType:

- (NSInteger)setCurrentDeviceMute:	(BOOL)mute
deviceType:	(TXMediaDeviceType)type

设置当前设备的静音状态（仅适用于桌面端）

这里的音量指的是麦克风和扬声器，摄像头是不支持静音操作的。

getCurrentDeviceMute:

getCurrentDeviceMute:

- (BOOL)getCurrentDeviceMute:	(TXMediaDeviceType)type

获取当前设备的静音状态（仅适用于桌面端）

这里的音量指的是麦克风和扬声器，摄像头是不支持静音操作的。

enableFollowingDefaultAudioDevice:enable:

enableFollowingDefaultAudioDevice:enable:

- (NSInteger)enableFollowingDefaultAudioDevice:	(TXMediaDeviceType)type
enable:	(BOOL)enable

设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）

仅支持设置麦克风和扬声器类型，摄像头暂不支持跟随系统默认设备

参数	描述
enable	是否跟随系统默认的音频设备。 true：跟随。当系统默认音频设备发生改变或者有新音频设备插入时，SDK 立即切换音频设备。 false：不跟随。当系统默认音频设备发生改变或者有新音频设备插入时，SDK 不会切换音频设备。
type	设备类型，详见 TXMediaDeviceType 定义。

startCameraDeviceTest:

startCameraDeviceTest:

- (NSInteger)startCameraDeviceTest:	(NSView *)view

开始摄像头测试（仅适用于桌面端）

注意

在测试过程中可以使用 setCurrentDevice 接口切换摄像头。

stopCameraDeviceTest

stopCameraDeviceTest

结束摄像头测试（仅适用于桌面端）

startMicDeviceTest:

startMicDeviceTest:

- (NSInteger)startMicDeviceTest:	(NSInteger)interval

开始麦克风测试（仅适用于桌面端）

该接口可以测试麦克风是否能正常工作，测试到的麦克风采集音量的大小，会以回调的形式通知给您，其中 volume 的取值范围为0 - 100。

参数	描述
interval	麦克风音量的回调间隔。

注意

该接口调用后默认会回播麦克风录制到的声音到扬声器中。

startMicDeviceTest:playback:

startMicDeviceTest:playback:

- (NSInteger)startMicDeviceTest:	(NSInteger)interval
playback:	(BOOL)playback

开始麦克风测试（仅适用于桌面端）

该接口可以测试麦克风是否能正常工作，测试到的麦克风采集音量的大小，会以回调的形式通知给您，其中 volume 的取值范围为0 - 100。

参数	描述
interval	麦克风音量的回调间隔。
playback	是否开启回播麦克风声音，开启后用户测试麦克风时会听到自己的声音。

stopMicDeviceTest

stopMicDeviceTest

结束麦克风测试（仅适用于桌面端）

startSpeakerDeviceTest:

startSpeakerDeviceTest:

- (NSInteger)startSpeakerDeviceTest:	(NSString *)audioFilePath

开始扬声器测试（仅适用于桌面端）

该接口通过播放指定的音频文件，用于测试播放设备是否能正常工作。如果用户在测试时能听到声音，说明播放设备能正常工作。

参数	描述
filePath	声音文件的路径

stopSpeakerDeviceTest

stopSpeakerDeviceTest

结束扬声器测试（仅适用于桌面端）

setObserver:

setObserver:

- (void)setObserver:	(nullable id<TXDeviceObserver>) observer

设备热插拔回调（仅适用于 Mac 系统）

setCameraCapturerParam:

setCameraCapturerParam:

- (void)setCameraCapturerParam:	(TXCameraCaptureParam *)params

设置摄像头采集偏好

setSystemVolumeType:

setSystemVolumeType:

- (NSInteger)setSystemVolumeType:	(TXSystemVolumeType)type

设置系统音量类型（仅适用于移动端）

@deprecated v9.5 版本开始不推荐使用，建议使用 `TRTCCloud` 中的 startLocalAudio(quality) 接口替代之，通过 quality 参数来决策音质。

TXSystemVolumeType(Deprecated)

TXSystemVolumeType(Deprecated)

系统音量类型（仅适用于移动设备）

@deprecated v9.5 版本开始不推荐使用。

现代智能手机中一般都具备两套系统音量类型，即“通话音量”和“媒体音量”。
通话音量：手机专门为接打电话所设计的音量类型，自带回声抵消（AEC）功能，并且支持通过蓝牙耳机上的麦克风进行拾音，缺点是音质比较一般。
当您通过手机侧面的音量按键下调手机音量时，如果无法将其调至零（也就是无法彻底静音），说明您的手机当前处于通话音量。
媒体音量：手机专门为音乐场景所设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。
当您通过手机侧面的音量按键下调手机音量时，如果能够将手机音量调至彻底静音，说明您的手机当前处于媒体音量。

SDK 目前提供了三种系统音量类型的控制模式：自动切换模式、全程通话音量模式、全程媒体音量模式。

枚举	取值	描述
TXSystemVolumeTypeAuto	0	自动切换模式
TXSystemVolumeTypeMedia	1	全程媒体音量
TXSystemVolumeTypeVOIP	2	全程通话音量

版权所有：腾讯云计算（北京）有限责任公司

TXAudioRoute

TXAudioRoute

音频路由（即声音的播放模式）

音频路由，即声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。
手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

枚举	取值	描述
TXAudioRouteSpeakerphone	0	Speakerphone：使用扬声器播放（即“免提”），扬声器位于手机底部，声音偏大，适合外放音乐。
TXAudioRouteEarpiece	1	Earpiece：使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

TXMediaDeviceType

TXMediaDeviceType

设备类型（仅适用于桌面平台）

该枚举值用于定义三种类型的音视频设备，即摄像头、麦克风和扬声器，以便让一套设备管理接口可以操控三种不同类型的设备。

枚举	取值	描述
TXMediaDeviceTypeUnknown	-1	未定义的设备类型
TXMediaDeviceTypeAudioInput	0	麦克风类型设备
TXMediaDeviceTypeAudioOutput	1	扬声器类型设备
TXMediaDeviceTypeVideoCamera	2	摄像头类型设备

TXMediaDeviceState

TXMediaDeviceState

版权所有：腾讯云计算（北京）有限责任公司

设备操作

该枚举值用于本地设备的状态变化通知 onDeviceChanged。

枚举	取值	描述
TXMediaDeviceStateAdd	0	设备已被插入
TXMediaDeviceStateRemove	1	设备已被移除
TXMediaDeviceStateActive	2	设备已启用
TXMediaDefaultDeviceChanged	3	系统默认设备变更

TXCameraCaptureMode

TXCameraCaptureMode

摄像头采集偏好

该枚举类型用于摄像头采集参数设置。

枚举	取值	描述
TXCameraResolutionStrategyAuto	0	自动调整采集参数。 SDK 根据实际的采集设备性能及网络情况，选择合适的摄像头输出参数，在设备性能及视频预览质量之间，维持平衡。
TXCameraResolutionStrategyPerformance	Not Defined	优先保证设备性能。 SDK 根据用户设置编码器的分辨率和帧率，选择最接近的摄像头输出参数，从而保证设备性能。
TXCameraResolutionStrategyHighQuality	Not Defined	优先保证视频预览质量。 SDK选择较高的摄像头输出参数，从而提高预览视频的质量。在这种情况下，会消耗更多的 CPU 及内存做视频前处理。
TXCameraCaptureManual	Not Defined	允许用户设置本地摄像头采集的视频宽高。

TXCameraCaptureParam

TXCameraCaptureParam

摄像头采集参数

该设置能决定本地预览图像画质。

枚举类型	描述
height	**字段含义：** 采集图像宽度
mode	**字段含义：** 摄像头采集偏好，请参见 TXCameraCaptureMode
width	**字段含义：** 采集图像长度

TXMediaDeviceInfo

TXMediaDeviceInfo

音视频设备的相关信息（仅适用于桌面平台）

该结构体用于描述一个音视频设备的关键信息，比如设备 ID、设备名称等等，以便用户能够在用户界面上选择自己期望使用的音视频设备。

枚举类型	描述
deviceId	设备 ID （UTF-8）
deviceName	设备名称 （UTF-8）
deviceProperties	设备属性
type	设备类型

类型定义

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 关键类型定义

Function: 分辨率、质量等级等枚举和常量值的定义

类型定义

结构体类型

函数列表	描述
TRTCParams	进房参数
TRTCVideoEncParam	视频编码参数
TRTCNetworkQosParam	网络流控（Qos）参数集
TRTCRenderParams	视频画面的渲染参数
TRTCQualityInfo	网络质量
TRTCVolumeInfo	音量大小
TRTCSpeedTestParams	测速参数
TRTCSpeedTestResult	网络测速结果
TRTCVideoFrame	视频帧信息
TRTCAudioFrame	音频帧数据
TRTCMixUser	云端混流中各路画面的描述信息
TRTCTranscodingConfig	云端混流的排版布局和转码参数
TRTCPublishCDNParam	向非腾讯云 CDN 上发布音视频流时需设置的转推参数
TRTCAudioRecordingParams	本地音频文件的录制参数

TRTCLocalRecordingParams	本地媒体文件的录制参数
TRTCAudioEffectParam	音效参数（已废弃）
TRTCSwitchRoomConfig	房间切换参数
TRTCAudioFrameDelegateFormat	音频自定义回调的格式参数
TRTCScreenCaptureSourceInfo	屏幕分享的目标信息（仅适用于桌面系统）
TRTCAudioParallelParams	远端音频流智能并发播放策略的参数
TRTCUser	媒体流发布相关配置的用户信息
TRTCPublishCdnUrl	向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置
TRTCPublishTarget	目标推流配置
TRTCVideoLayout	转码视频布局
TRTCWatermark	水印布局
TRTCStreamEncoderParam	媒体流编码输出参数
TRTCStreamMixingConfig	媒体流转码配置参数
TRTCPayloadPrivateEncryptionConfig	媒体流私有加密配置
TRTCAudioVolumeEvaluateParams	音量评估等相关参数设置

枚举类型

枚举类型	描述
TRTCVideoResolution	视频分辨率
TRTCVideoResolutionMode	视频宽高比模式
TRTCVideoStreamType	视频流类型
TRTCVideoFillMode	视频画面填充模式
TRTCVideoRotation	视频画面旋转方向
TRTCBeautyStyle	美颜（磨皮）算法
TRTCVideoPixelFormat	视频像素格式

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoBufferType	视频数据传递方式
TRTCVideoMirrorType	视频的镜像类型
TRTCSnapshotSourceType	本地视频截图的数据源
TRTCAppScene	应用场景
TRTCRoleType	角色
TRTCQosControlMode	流控模式（已废弃）
TRTCVideoQosPreference	画质偏好
TRTCQuality	网络质量
TRTCAVStatusType	音视频状态类型
TRTCAVStatusChangeReason	音视频状态变化原因类型
TRTCAudioSampleRate	音频采样率
TRTCAudioQuality	声音音质
TRTCAudioRoute	音频路由（即声音的播放模式）
TRTCReverbType	声音混响模式
TRTCVoiceChangerType	变声类型
TRTCSystemVolumeType	系统音量类型（仅适用于移动设备）
TRTCAudioFrameOperationMode	音频回调数据读写模式
TRTCLogLevel	Log 级别
TRTCGSensorMode	重力感应开关（仅适用于移动端）
TRTCScreenCaptureSourceType	屏幕分享的目标类型（仅适用于桌面端）
TRTCTranscodingConfigMode	云端混流的排版模式
TRTCRecordType	媒体录制类型
TRTCMixInputType	混流输入类型
TRTCAudioRecordingContent	音频录制内容类型
TRTCPublishMode	媒体流发布模式

版权所有：腾讯云计算（北京）有限责任公司

TRTCEncryptionAlgorithm	加密算法
TRTCSpeedTestScene	测速场景
TRTCGravitySensorAdaptiveMode	设置重力感应的适配模式（仅适用于移动端）

TRTCVideoResolution

TRTCVideoResolution

视频分辨率

此处仅定义横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTCVideoResolution_120_120	1	宽高比 1:1；分辨率 120x120；建议码率（VideoCall）80kbps; 建议码率（LIVE）120kbps。
TRTCVideoResolution_160_160	3	宽高比 1:1 分辨率 160x160；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTCVideoResolution_270_270	5	宽高比 1:1；分辨率 270x270；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_480_480	7	宽高比 1:1；分辨率 480x480；建议码率（VideoCall）350kbps; 建议码率（LIVE）500kbps。
TRTCVideoResolution_160_120	50	宽高比4:3；分辨率 160x120；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTCVideoResolution_240_180	52	宽高比 4:3；分辨率 240x180；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。
TRTCVideoResolution_280_210	54	宽高比 4:3；分辨率 280x210；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_320_240	56	宽高比 4:3；分辨率 320x240；建议码率（VideoCall）250kbps; 建议码率（LIVE）375kbps。
TRTCVideoResolution_400_300	58	宽高比 4:3；分辨率 400x300；建议码率（VideoCall）300kbps; 建议码率（LIVE）450kbps。
TRTCVideoResolution_480_360	60	宽高比 4:3；分辨率 480x360；建议码率（VideoCall）400kbps; 建议码率（LIVE）600kbps。

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoResolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps; 建议码率（LIVE）900kbps。
TRTCVideoResolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps; 建议码率（LIVE）1500kbps。
TRTCVideoResolution_160_90	100	宽高比 16:9；分辨率 160x90；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。
TRTCVideoResolution_256_144	102	宽高比 16:9；分辨率 256x144；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_320_180	104	宽高比 16:9；分辨率 320x180；建议码率（VideoCall）250kbps; 建议码率（LIVE）400kbps。
TRTCVideoResolution_480_270	106	宽高比 16:9；分辨率 480x270；建议码率（VideoCall）350kbps; 建议码率（LIVE）550kbps。
TRTCVideoResolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps; 建议码率（LIVE）900kbps。
TRTCVideoResolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps; 建议码率（LIVE）1300kbps。
TRTCVideoResolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps; 建议码率（LIVE）1800kbps。
TRTCVideoResolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps; 建议码率（LIVE）3000kbps。

TRTCVideoResolutionMode

TRTCVideoResolutionMode

视频宽高比模式

TRTCVideoResolution 中仅定义了横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTCVideoResolutionModeLandscape	0	横屏分辨率，例如：TRTCVideoResolution_640_360

		+ TRTCVideoResolutionModeLandscape = 640 × 360。
TRTCVideoResolutionModePortrait	1	竖屏分辨率，例如：TRTCVideoResolution_640_360 + TRTCVideoResolutionModePortrait = 360 × 640。

TRTCVideoStreamType

TRTCVideoStreamType

视频流类型

TRTC 内部有三种不同的视频流，分别是：
高清大画面：一般用来传输摄像头的视频数据。
低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

注意

不支持单独开启低清小画面，小画面必须依附于大画面而存在，SDK 会自动设定低清小画面的分辨率和码率。

枚举	取值	描述
TRTCVideoStreamTypeBig	0	高清大画面，一般用来传输摄像头的视频数据。
TRTCVideoStreamTypeSmall	1	低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
TRTCVideoStreamTypeSub	2	辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

TRTCVideoFillMode

TRTCVideoFillMode

视频画面填充模式

如果视频显示区域的宽高比不等于视频内容的宽高比时，需要指定画面的填充模式。

枚举	取值	描述
TRTCVideoFillMode_Fill	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完

		整。
TRTCVideoFillMode_Fit	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边。

TRTCVideoRotation

TRTCVideoRotation

视频画面旋转方向

TRTC 提供了对本地和远程画面的旋转角度设置 API，下列的旋转角度都是指顺时针方向的。

枚举	取值	描述
TRTCVideoRotation_0	0	不旋转。
TRTCVideoRotation_90	1	顺时针旋转90度。
TRTCVideoRotation_180	2	顺时针旋转180度。
TRTCVideoRotation_270	3	顺时针旋转270度。

TRTCBeautyStyle

TRTCBeautyStyle

美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案。

枚举	取值	描述
TRTCBeautyStyleSmooth	0	光滑，算法比较激进，磨皮效果比较明显，适用于秀场直播。
TRTCBeautyStyleNature	1	自然，算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。
TRTCBeautyStylePitu	2	优图，由优图实验室提供，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高。

TRTCVideoPixelFormat

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoPixelFormat

视频像素格式

TRTC 提供针对视频的自定义采集和自定义渲染功能：
在自定义采集功能中，您可以用下列枚举值描述您采集的视频像素格式。
在自定义渲染功能中，您可以指定您期望 SDK 回调出的视频像素格式。

枚举	取值	描述
TRTCVideoPixelFormat_Unknown	0	未定义的格式。
TRTCVideoPixelFormat_I420	1	YUV420P(I420) 格式。
TRTCVideoPixelFormat_Texture_2D	7	OpenGL 2D 纹理格式。
TRTCVideoPixelFormat_32BGRA	6	BGRA 格式
TRTCVideoPixelFormat_NV12	5	YUV420SP（NV12）格式。

TRTCVideoBufferType

TRTCVideoBufferType

视频数据传递方式

在自定义采集和自定义渲染功能，您需要用到下列枚举值来指定您希望以什么方式传递视频数据：
方案一：使用内存 Buffer 传递视频数据，该方案在 iOS 效率尚可，但在 Android 系统上效率较差，Windows 暂时仅支持内存 Buffer 的传递方式。
方案二：使用 Texture 纹理传递视频数据，该方案在 iOS 和 Android 系统下均有较高的效率，Windows 暂不支持，需要您有一定的 OpenGL 编程基础。

枚举	取值	描述
TRTCVideoBufferType_Unknown	0	未定义的传递方式。
TRTCVideoBufferType_PixelBuffer	1	使用内存 Buffer 传递视频数据，iOS：PixelBuffer；Android：用于 JNI 层的 Direct Buffer；Win：内存数据块。
TRTCVideoBufferType_NSData	2	使用内存 Buffer 传递视频数据，iOS：经过一次额外整理后更加紧凑的 NSData 类型的内存块； Android：用于 JAVA 层的 byte[]。 该传递的方式的性能是几种方案中效率较差的一种。

TRTCVideoBufferType_Texture	3	使用 OpengGL 纹理传递视频数据。

TRTCVideoMirrorType

TRTCVideoMirrorType

视频的镜像类型

视频的镜像是指对视频内容进行左右翻转，尤其是对本地的摄像头预览视频，开启镜像后能给主播带来熟悉的“照镜子”体验。

枚举	取值	描述
TRTCVideoMirrorTypeAuto	0	自动模式：如果正使用前置摄像头则开启镜像，如果是后置摄像头则不开启镜像（仅适用于移动设备）。
TRTCVideoMirrorTypeEnable	1	强制开启镜像，不论当前使用的是前置摄像头还是后置摄像头。
TRTCVideoMirrorTypeDisable	2	强制关闭镜像，不论当前使用的是前置摄像头还是后置摄像头。

TRTCSnapshotSourceType

TRTCSnapshotSourceType

本地视频截图的数据源

SDK 支持从如下两种数据源中截取图片并保存成本地文件：
视频流：从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
渲染层：从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
采集层：从采集控件中截取采集到的视频内容，可以截取采集出来的高清截图。

枚举	取值	描述
TRTCSnapshotSourceTypeStream	0	从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
TRTCSnapshotSourceTypeView	1	从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
TRTCSnapshotSourceTypeCapture	2	从采集控件中截取采集到的视频内容，可以截取采集

		出来的高清截图。

TRTCAppScene

TRTCAppScene

应用场景

TRTC 针对常见的音视频应用场景都进行了定向优化，以满足各种垂直场景下的差异化要求，主要场景可以分为如下两类：
直播（LIVE）场景：包括 LIVE 和 VoiceChatRoom，前者是音频+视频，后者是纯音频。
直播场景下，用户被分成“主播”和“观众”两种角色，单个房间中同时最多支持10万人在线，适合于观众人数众多的直播场景。
实时（RTC）场景：包括 VideoCall 和 AudioCall，前者是音频+视频，后者是纯音频。
实时场景下，用户没有角色的差异，但单个房间中同时最多支持 300 人在线，适合于小范围实时通信的场景。

枚举	取值	描述
TRTCAppSceneVideoCall	0	视频通话场景，支持720P、1080P高清画质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于[1对1视频通话]、[300人视频会议]、[在线问诊]、[教育小班课]、[远程面试]等业务场景。
TRTCAppSceneLIVE	1	视频互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于300ms；支持十万级别观众同时播放，播放延时低至1000ms。 适用于[低延时互动直播]、[大班课]、[主播PK]、[视频相亲]、[在线互动课堂]、[远程培训]、[超大型会议]等业务场景。 **注意** 此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。
TRTCAppSceneAudioCall	2	语音通话场景，默认采用 SPEECH 音质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于 [1对1语音通话]、[300人语音会议]、[语音聊天]、[语音会议]、[在线狼人杀] 等业务场景。
TRTCAppSceneVoiceChatRoom	3	语音互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于 300ms；支持十万级别观众同时播放，播放延时低至 1000ms。 适用于 [语音俱乐部]、[在线K歌房]、[音乐直播间]、[FM电台] 等业务场景。 **注意**

		此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。

TRTCRoleType

TRTCRoleType

角色

仅适用于直播类场景（即 TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom），把用户区分成两种不同的身份：

主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。

观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

枚举	取值	描述
TRTCRoleAnchor	20	主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。
TRTCRoleAudience	21	观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

TRTCQosControlMode(Deprecated)

TRTCQosControlMode(Deprecated)

流控模式（已废弃）

枚举	取值	描述
TRTCQosControlModeClient	0	本地控制，用于 SDK 开发内部调试，客户请勿使用。
TRTCQosControlModeServer	1	云端控制，默认模式，推荐选择。

TRTCVideoQosPreference

TRTCVideoQosPreference

画质偏好

TRTC 在弱网络环境下有两种调控模式：“优先保证画面清晰”或“优先保证画面流畅”，两种模式均会优先保障声音数据的传输。

枚举	取值	描述
TRTCVideoQosPreferenceSmooth	1	流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。
TRTCVideoQosPreferenceClear	2	清晰优先（默认值）：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。

TRTCQuality

TRTCQuality

网络质量

TRTC 会每隔两秒对当前的网络质量进行评估，评估结果为六个等级：Excellent 表示最好，Down 表示最差。

枚举	取值	描述
TRTCQuality_Unknown	0	未定义。
TRTCQuality_Excellent	1	当前网络非常好。
TRTCQuality_Good	2	当前网络比较好。
TRTCQuality_Poor	3	当前网络一般。
TRTCQuality_Bad	4	当前网络较差。
TRTCQuality_Vbad	5	当前网络很差。
TRTCQuality_Down	6	当前网络不满足 TRTC 的最低要求。

TRTCAVStatusType

TRTCAVStatusType

音视频状态类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态。

枚举	取值	描述
TRTCAVStatusStopped	0	停止播放。
TRTCAVStatusPlaying	1	正在播放。
TRTCAVStatusLoading	2	正在加载。

TRTCAVStatusChangeReason

TRTCAVStatusChangeReason

音视频状态变化原因类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态变化原因。

枚举	取值	描述
TRTCAVStatusChangeReasonInternal	0	缺省值。
TRTCAVStatusChangeReasonBufferingBegin	1	网络缓冲。
TRTCAVStatusChangeReasonBufferingEnd	2	结束缓冲。
TRTCAVStatusChangeReasonLocalStarted	3	本地启动音频或视频流播放。
TRTCAVStatusChangeReasonLocalStopped	4	本地停止音频或视频流播放。
TRTCAVStatusChangeReasonRemoteStarted	5	远端音频或视频流开始（或继续）。
TRTCAVStatusChangeReasonRemoteStopped	6	远端音频或视频流停止（或中断）。

TRTCAudioSampleRate

TRTCAudioSampleRate

音频采样率

音频采样率用来衡量声音的保真程度，采样率越高保真程度越好，如果您的应用场景有音乐的存在，推荐使用 TRTCAudioSampleRate48000。

枚举	取值	描述
TRTCAudioSampleRate16000	16000	16k 采样率。

TRTCAudioSampleRate32000	32000	32k 采样率。
TRTCAudioSampleRate44100	44100	44.1k 采样率。
TRTCAudioSampleRate48000	48000	48k采样率

TRTCAudioQuality

TRTCAudioQuality

声音音质

TRTC 提供了三种精心校调好的模式，用来满足各种垂直场景下对音质的差异化追求：
人声模式（Speech）：适用于以人声沟通为主的应用场景，该模式下音频传输的抗性较强，TRTC 会通过各种人声处理技术保障在弱网络环境下的流畅度最佳。
音乐模式（Music）：适用于对声乐要求很苛刻的场景，该模式下音频传输的数据量很大，TRTC 会通过各项技术确保音乐信号在各频段均能获得高保真的细节还原度。
默认模式（Default）：介于 Speech 和 Music 之间的档位，对音乐的还原度比人声模式要好，但传输数据量比音乐模式要低很多，对各种场景均有不错的适应性。

枚举	取值	描述
TRTCAudioQualitySpeech	1	人声模式：采样率：16k；单声道；编码码率：16kbps；具备几个模式中最强的网络抗性，适合语音通话为主的场景，比如在线会议，语音通话等。
TRTCAudioQualityDefault	2	默认模式：采样率：48k；单声道；编码码率：50kbps；介于 Speech 和 Music 之间的档位，SDK 默认档位，推荐选择。
TRTCAudioQualityMusic	3	音乐模式：采样率：48k；全频带立体声；编码码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

TRTCAudioRoute

TRTCAudioRoute

音频路由（即声音的播放模式）

音频路由，即声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。
手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。

设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。
音频路由为有线耳机。
音频路由为蓝牙耳机。
音频路由为USB专业声卡设备。

枚举	取值	描述
TRTCAudioModeSpeakerphone	0	Speakerphone：使用扬声器播放（即“免提”），扬声器位于手机底部，声音偏大，适合外放音乐。
TRTCAudioModeEarpiece	1	Earpiece：使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。
TRTCAudioModeWiredHeadset	2	WiredHeadset：使用有线耳机播放。
TRTCAudioModeBluetoothHeadset	3	BluetoothHeadset：使用蓝牙耳机播放。
TRTCAudioModeSoundCard	4	SoundCard：使用 USB 声卡播放。

TRTCReverbType

TRTCReverbType

声音混响模式

该枚举值应用于设定直播场景中的混响模式，常用于秀场直播中。

枚举	取值	描述
TRTCReverbType_0	0	关闭混响。
TRTCReverbType_1	1	KTV。
TRTCReverbType_2	2	小房间。
TRTCReverbType_3	3	大会堂。
TRTCReverbType_4	4	低沉。
TRTCReverbType_5	5	洪亮。
TRTCReverbType_6	6	金属声。
TRTCReverbType_7	7	磁性。

版权所有：腾讯云计算（北京）有限责任公司

TRTCVoiceChangerType

TRTCVoiceChangerType

变声类型

该枚举值应用于设定直播场景中的变声模式，常用于秀场直播中。

枚举	取值	描述
TRTCVoiceChangerType_0	0	关闭变声。
TRTCVoiceChangerType_1	1	熊孩子。
TRTCVoiceChangerType_2	2	萝莉。
TRTCVoiceChangerType_3	3	大叔。
TRTCVoiceChangerType_4	4	重金属。
TRTCVoiceChangerType_5	5	感冒。
TRTCVoiceChangerType_6	6	外国人。
TRTCVoiceChangerType_7	7	困兽。
TRTCVoiceChangerType_8	8	死肥仔。
TRTCVoiceChangerType_9	9	强电流。
TRTCVoiceChangerType_10	10	重机械。
TRTCVoiceChangerType_11	11	空灵。

TRTCSystemVolumeType

TRTCSystemVolumeType

系统音量类型（仅适用于移动设备）

现代智能手机中一般都具备两套系统音量类型，即“通话音量”和“媒体音量”。
通话音量：手机专门为接打电话所设计的音量类型，自带回声抵消（AEC）功能，并且支持通过蓝牙耳机上的麦克风进行拾音，缺点是音质比较一般。当您通过手机侧面的音量按键下调手机音量时，如果无法将其调至零（也就是无法彻底静音），说明您的手机当前处于通话音量。

媒体音量：手机专门为音乐场景所设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。当您通过手机侧面的音量按键下调手机音量时，如果能够将手机音量调至彻底静音，说明您的手机当前处于媒体音量。

SDK 目前提供了三种系统音量类型的控制模式：自动切换模式、全程通话音量模式、全程媒体音量模式。

枚举	取值	描述
TRTCSystemVolumeTypeAuto	0	自动切换模式： 也被称为“麦上通话，麦下媒体”，即主播上麦时使用通话音量，观众不上麦则使用媒体音量，适合在线直播场景。 如果您在 enterRoom 时选择的场景为 TRTCAppSceneLIVE 或 TRTCAppSceneVoiceChatRoom，SDK 会自动使用该模式。
TRTCSystemVolumeTypeMedia	1	全程媒体音量： 通话全程使用媒体音量，并不是非常常用的音量类型，适用于对音质要求比较苛刻的音乐场景中。 如果您的用户大都使用外接设备（比如外接声卡）为主，可以使用该模式，否则请慎用。
TRTCSystemVolumeTypeVOIP	2	全程通话音量： 该方案的优势在于用户在上下麦时音频模块无需切换工作模式，可以做到无缝上下麦，适合于用户需要频繁上下麦的应用场景。 如果您在 enterRoom 时选择的场景为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，SDK 会自动使用该模式。

TRTCAudioFrameOperationMode

TRTCAudioFrameOperationMode

音频回调数据读写模式

TRTC 提供了两种音频回调数据的操作模式。

读写模式（ReadWrite）：可以获取并修改回调的音频数据，默认模式。

只读模式（ReadOnly）：仅从回调中获取音频数据。

枚举	取值	描述
TRTCAudioFrameOperationModeReadWrite	0	读写模式：可以获取并修改回调的音频数据。
TRTCAudioFrameOperationModeReadOnly	1	只读模式：仅从回调中获取音频数据。

TRTCLogLevel

TRTCLogLevel

Log 级别

不同的日志等级定义了不同的详实程度和日志数量，推荐一般情况下将日志等级设置为：TRTCLogLevelInfo。

枚举	取值	描述
TRTCLogLevelVerbose	0	输出所有级别的 Log。
TRTCLogLevelDebug	1	输出 DEBUG，INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelInfo	2	输出 INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelWarn	3	输出WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelError	4	输出ERROR 和 FATAL 级别的 Log。
TRTCLogLevelFatal	5	仅输出 FATAL 级别的 Log。
TRTCLogLevelNone	6	不输出任何 SDK Log。

TRTCGSensorMode

TRTCGSensorMode

重力感应开关（仅适用于移动端）

枚举	取值	描述
TRTCGSensorMode_Disable	0	不适配重力感应，该模式是桌面平台上的默认值，该模式下，当前用户发布出去的视频画面不受重力感应方向变化的影响。
TRTCGSensorMode_UIAutoLayout	1	适配重力感应，该模式是移动平台上的默认值，该模式下，当前用户发布出去的视频画面会跟随设备的重力感应方向进行相应的调整，同时本地预览画面保持方向不变。 SDK 目前支持的一种适配模式是：当手机或 Pad 上下颠倒时，为了保证远端用户看到的画面方向正常，SDK 会自动将发布出去的画面上下旋转180度。如果您的 APP 的界面层开启了重力感应自适应，推荐使用 UIFixLayout 模式。

TRTCGSensorMode_UIFixLayout	2	适配重力感应 该模式下，当前用户发布出去的视频画面会跟随设备的重力感应方向进行相应的调整，同时本地预览画面也会进行相应的旋转适配。 目前支持的一种特性是：当手机或 Pad 上下颠倒时，为了保证远端用户看到的画面方向正常，SDK 会自动将发布出去的画面上下旋转180度。 如果您的 APP 的界面层不支持重力感应自适应，并且希望 SDK 的视频画面能够适配重力感应方向，推荐使用 UIFixLayout 模式。 @deprecated 从v11.5版本开始，不再支持 TRTCGSensorMode_UIFixLayout，只支持上面两种模式。

TRTCScreenCaptureSourceType

TRTCScreenCaptureSourceType

屏幕分享的目标类型（仅适用于桌面端）

枚举	取值	描述
TRTCScreenCaptureSourceTypeUnknown	-1	未定义。
TRTCScreenCaptureSourceTypeWindow	0	该分享目标是某一个应用的窗口。
TRTCScreenCaptureSourceTypeScreen	1	该分享目标是某一台显示器的屏幕。

TRTCTranscodingConfigMode

TRTCTranscodingConfigMode

云端混流的排版模式

TRTC 的云端混流服务能够将房间中的多路音视频流混合成一路，因此您需要指定画面的排版方案，我们提供了如下几种排版模式。

枚举	取值	描述
TRTCTranscodingConfigMode_Unknown	0	未定义。
TRTCTranscodingConfigMode_Manual	1	全手动排版模式。

		该模式下，您需要指定每一路画面的精确排版位置。该模式的自由度最高，但易用性也最差： 您需要填写 TRTCTranscodingConfig 中的所有参数，包括每一路画面（TRTCMixUser）的位置坐标。 您需要监听 TRTCCloudDelegate 中的 onUserVideoAvailable() 和 onUserAudioAvailable() 事件回调，并根据当前房间中各个麦上用户的音视频状态不断地调整 mixUsers 参数。
TRTCTranscodingConfigMode_Template_PureAudio	2	纯音频模式。 该模式适用于语音通话（AudioCall）和语音聊天室（VoiceChatRoom）等纯音频的应用场景。 您只需要在进入房间后，通过 setMixTranscodingConfig 接口设置一次，之后 SDK 就会自动把房间内所有上麦用户的声音混流到当前用户的直播流上。 您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，只需设置 audioSampleRate、audioBitrate 和 audioChannels 等参数即可。
TRTCTranscodingConfigMode_Template_PresetLayout	3	预排版模式。 最受欢迎的排版模式，因为该模式支持您通过占位符提前对各路画面的位置进行设定，之后 SDK 会自动根据房间中画面的路数动态进行适配调整。 此模式下，您依然需要设置 mixUsers 参数，但可以将 userId 设置为“占位符”，可选的占位符有： "$PLACE_HOLDER_REMOTE$" ：指代远程用户的画面，可以设置多个。 "$PLACE_HOLDER_LOCAL_MAIN$" ：指代本地摄像头画面，只允许设置一个。 "$PLACE_HOLDER_LOCAL_SUB$" ：指代本地屏幕分享画面，只允许设置一个。 此模式下，您不需要监听 TRTCCloudDelegate 中的 onUserVideoAvailable 和 onUserAudioAvailable 回调进行实时调

		整，只需要在进房成功后调用一次 setMixTranscodingConfig 即可，之后 SDK 会自动将真实的 userId 补位到您设置的占位符上。
TRTCTranscodingConfigMode_Template_ScreenSharing	4	屏幕分享模式。 适用于在线教育场景等以屏幕分享为主的应用场景，仅支持 Windows 和 Mac 两个平台的 SDK。 该模式下，SDK 会先根据您通过 videoWidth 和 videoHeight 参数设置的目标分辨率构建一张画布， 当老师未开启屏幕分享时，SDK 会将老师的摄像头画面等比例拉伸绘制到该画布上； 当老师开启屏幕分享之后，SDK 会将屏幕分享画面绘制到同样的画布上。 此种排版模式的目的是为了确保混流模块的输出分辨率一致，避免课程回放和网页观看的花屏问题（网页播放器不支持可变分辨率）。 同时，连麦学生的声音也会被默认混合到老师的音视频流中。 由于教学模式下的视频内容以屏幕分享为主，因此同时传输摄像头画面和屏幕分享画面是非常浪费带宽的。 推荐的做法是直接将摄像头画面通过 setLocalVideoRenderCallback 接口自定义绘制到当前屏幕上。 在该模式下，您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，SDK 不会混合学生的画面，以免干扰屏幕分享的效果。 您可以将 TRTCTranscodingConfig 中的 width x height 设为 0px × 0px，SDK 会自动根据用户当前屏幕的宽高比计算出一个合适的分辨率： 如果老师当前屏幕宽度 <= 1920px，SDK 会使用老师当前屏幕的实际分辨率。 如果老师当前屏幕宽度 > 1920px，SDK 会根据当前屏幕宽高比，选择 1920x1080(16:9)、1920x1200(16:10)、1920x1440(4:3) 三种分辨率中的一种。

TRTCRecordType

TRTCRecordType

媒体录制类型

该枚举类型用于本地媒体录制接口 startLocalRecording，用于指定是录制音视频文件还是纯音频文件。

枚举	取值	描述
TRTCRecordTypeAudio	0	仅录制音频。
TRTCRecordTypeVideo	1	仅录制视频。
TRTCRecordTypeBoth	2	同时录制音频和视频。

TRTCMixInputType

TRTCMixInputType

混流输入类型

枚举	取值	描述
TRTCMixInputTypeUndefined	0	默认值，考虑到针对老版本的兼容性，如果您指定了 inputType 为 Undefined，SDK 会根据另一个参数 pureAudio 的数值决定混流输入类型。
TRTCMixInputTypeAudioVideo	1	混入音频和视频。
TRTCMixInputTypePureVideo	2	只混入视频。
TRTCMixInputTypePureAudio	3	只混入音频。
TRTCMixInputTypeWatermark	4	混入水印，此时您无需指定 userId 字段，但需要指定 image 字段，推荐使用 png 格式的图片。

TRTCAudioRecordingContent

TRTCAudioRecordingContent

音频录制内容类型

该枚举类型用于音频录制接口 startAudioRecording，用于指定录制音频的内容。

枚举	取值	描述
TRTCAudioRecordingContentAll	0	录制本地和远端所有音频。
TRTCAudioRecordingContentLocal	1	仅录制本地音频。
TRTCAudioRecordingContentRemote	2	仅录制远端音频。

TRTCPublishMode

TRTCPublishMode

媒体流发布模式

该枚举类型用于媒体流发布接口 startPublishMediaStream TRTC 的媒体流发布服务能够将房间中的多路音视频流混合成一路发布至 CDN 或者回推到房间内，也可以将您当前的这路音视频发布到腾讯或者第三方 CDN 因此您需要指定对应媒体流的发布模式，我们提供了如下几种模式。

枚举	取值	描述
TRTCPublishModeUnknown	0	未定义。
TRTCPublishBigStreamToCdn	1	您可以通过设置该参数将您房间内的主路流（TRTCVideoStreamTypeBig）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishSubStreamToCdn	2	您可以通过设置该参数将您房间内的辅路流（TRTCVideoStreamTypeSub）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishMixStreamToCdn	3	您可以通过设置该参数，配合编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishMixStreamToRoom	4	您可以通过设置该参数，配合媒体流编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您指定的房间中。 通过 TRTCPublishTarget 中的 TRTCUser 进行指定回推房间的机器人信息。

TRTCEncryptionAlgorithm

TRTCEncryptionAlgorithm

加密算法

该枚举类型用于媒体流私有加密算法选择。

枚举	取值	描述
TRTCEncryptionAlgorithmAes128Gcm	0	AES GCM 128。
TRTCEncryptionAlgorithmAes256Gcm	1	AES GCM 256。

TRTCSpeedTestScene

TRTCSpeedTestScene

测速场景

该枚举类型用于测速场景选择。

枚举	取值	描述
TRTCSpeedTestScene_DelayTesting	1	延迟测试。
TRTCSpeedTestScene_DelayAndBandwidthTesting	2	延迟与带宽测试。
TRTCSpeedTestScene_OnlineChorusTesting	3	在线合唱测试。

TRTCGravitySensorAdaptiveMode

TRTCGravitySensorAdaptiveMode

设置重力感应的适配模式（仅适用于移动端）

枚举	取值	描述
TRTCGravitySensorAdaptiveMode_Disable	0	关闭重力感应，根据当前采集分辨率与设置的编码分辨率决策，如果两者不一致，则通过旋转90度，保证最大画幅。
TRTCGravitySensorAdaptiveMode_FillByCenterCrop	1	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用居中裁剪模式。

版权所有：腾讯云计算（北京）有限责任公司

TRTCGravitySensorAdaptiveMode_FitWithBlackBorder	2	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用叠加黑边模式。

TRTCParams

TRTCParams

进房参数

作为 TRTC SDK 的进房参数，只有该参数填写正确，才能顺利进入 roomId 或者 strRoomId 所指定的音视频房间。
由于历史原因，TRTC 支持数字和字符串两种类型的房间号，分别是 roomId 和 strRoomId。
请注意：不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 “123” 在 TRTC 看来是两个完全不同的房间。

枚举类型	描述
bussInfo	【字段含义】业务数据字段（选填），部分高级特性才需要用到此字段。 【推荐取值】请不要自行设置该字段。
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
role	【字段含义】直播场景下的角色，仅适用于直播场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom），通话场景下指定该参数是无效的。 【推荐取值】默认值：主播（TRTCRoleAnchor）。
roomId	【字段含义】数字房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为 0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。
sdkAppId	【字段含义】应用标识（必填），腾讯云基于 sdkAppId 完成计费统计。 【推荐取值】在 **实时音视频控制台** 创建应用后可以在账号信息页面中得到该 ID。
strRoomId	【字段含义】字符串房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。

	【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、`!` 、`#` 、`$` 、`%` 、`&` 、`(` 、`)` 、`+` 、`-` 、`:` 、`;` 、`<` 、`=` 、`.` 、`>` 、`?` 、`@` 、`[` 、`]` 、`^` 、`_` 、`{` 、`}` 、`|` 、`~` 、`,` 。
streamId	【字段含义】用于指定在腾讯云直播平台上的 streamId（选填），设置之后，您可以在腾讯云直播 CDN 上通过标准拉流方案（FLV 或 HLS）播放该用户的音视频流。 【推荐取值】限制长度为64字节，可以不填写，一种推荐的方案是使用 `sdkappid_roomid_userid_main` 作为 streamid，这种命名方式容易辨认且不会在您的多个应用中发生冲突。 【特殊说明】要使用腾讯云直播 CDN，您需要先在 控制台 中的功能配置页开启“启动自动旁路直播”开关。 【参考文档】**CDN 旁路直播**。
userDefineRecordId	【字段含义】云端录制开关（选填），用于指定是否要在云端将该用户的音视频流录制下来。 【参考文档】云端录制。 【推荐取值】限制长度为64字节，只允许包含大小写英文字母（a-zA-Z）、数字（0-9）及下划线和连词符。 方案一：手动录制方案： 1. 在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `手动录制` 。 3. 设置手动录制后，在一个 TRTC 房间中只有设置了 userDefineRecordId 参数的用户才会在云端录制出视频文件，不指定该参数的用户不会产生录制行为。 4. 云端会以 “userDefineRecordId_起始时间_结束时间” 的格式命名录制下来的文件。 方案二：自动录制方案： 1. 需要在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `自动录制` 。 3. 设置自动录制后，在一个 TRTC 房间中的任何一个有音视频上行的用户，均会在云端录制出视频文件。 4. 文件会以 `userDefineRecordId_起始时间_结束时间` 的格式命名，如果不指定 userDefineRecordId，则文件会以 `streamId_起始时间_结束时间` 命名。
userId	【字段含义】用户标识（必填），当前用户的 userId，相当于用户名，使用 UTF-8 编码。

	【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。
userSig	【字段含义】用户签名（必填），当前 userId 对应的验证签名，相当于使用云服务的登录密码。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCVideoEncParam

TRTCVideoEncParam

视频编码参数

该设置决定远端用户看到的画面质量，同时也决定了云端录制出的视频文件的画面质量。

枚举类型	描述
enableAdjustRes	【字段含义】是否允许动态调整分辨率（开启后会对云端录制产生影响）。 【推荐取值】该功能适用于不需要云端录制的场景，开启后 SDK 会根据当前网络情况，智能选择出一个合适的分辨率，避免出现“大分辨率+小码率”的低效编码模式。 【特别说明】默认值：关闭。如有云端录制的需求，请不要开启此功能，因为如果视频分辨率发生变化后，云端录制出的 MP4 在普通的播放器上无法正常播放。
minVideoBitrate	【字段含义】最低视频码率，SDK 会在网络不佳的情况下主动降低视频码率以保持流畅度，最低会降至 minVideoBitrate 所设定的数值。 【特别说明】 默认值：0，此时最低码率由 SDK 会根据您指定的分辨率，自动计算出合适的数值。 【推荐取值】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求 `弱网络下允许卡顿但要保持清晰` 的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求 `弱网络下允许模糊但要保持流畅` 的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
resMode	【字段含义】分辨率模式（横屏分辨率 or 竖屏分辨率）。 【推荐取值】手机平台（iOS、Android）建议选择 Portrait，桌面平台（Windows、Mac）建议选择 Landscape。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如：640 × 360 + Portrait = 360 × 640。
videoBitrate	【字段含义】目标视频码率，SDK 会按照目标码率进行编码，只有在弱网络环境下才会主动降低视频码率。

	【推荐取值】请参见本 TRTCVideoResolution 在各档位注释的最佳码率，也可以在此基础上适当调高。比如：TRTCVideoResolution_1280_720 对应 1200kbps 的目标码率，您也可以设置为 1500kbps 用来获得更好的观感清晰度。 【特别说明】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求“弱网络下允许卡顿但要保持清晰”的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求“弱网络下允许模糊但要保持流畅”的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
videoFps	【字段含义】视频采集帧率。 【推荐取值】15fps或20fps。5fps以下，卡顿感明显。10fps以下，会有轻微卡顿感。20fps以上，会浪费带宽（电影的帧率为24fps）。 【特别说明】部分 Android 手机的前置摄像头并不支持15fps以上的采集帧率，部分主打美颜功能的 Android 手机的前置摄像头的采集帧率可能低于10fps。
videoResolution	【字段含义】视频分辨率。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如：640 × 360 + Portrait = 360 × 640。 【推荐取值】 手机视频通话：建议选择 360 × 640 及以下分辨率，resMode 选择 Portrait，即竖屏分辨率。 手机在线直播：建议选择 540 × 960，resMode 选择 Portrait，即竖屏分辨率。 桌面平台（Win + Mac）：建议选择 640 × 360 及以上分辨率，resMode 选择 Landscape，即横屏分辨率。

TRTCNetworkQosParam

TRTCNetworkQosParam

网络流控（Qos）参数集

网络流控相关参数，该设置决定 SDK 在弱网络环境下的调控策略（例如：“清晰优先”或“流畅优先”）。

枚举类型	描述
controlMode	【字段含义】流控模式（已废弃）。 【推荐取值】云端控制。 【特别说明】请设置为云端控制模式（TRTCQosControlModeServer）。
preference	【字段含义】清晰优先还是流畅优先。 【推荐取值】清晰优先。 【特别说明】该参数主要影响 TRTC 在较差网络环境下的音视频表现：

	流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。参见 TRTCVideoQosPreferenceSmooth 清晰优先（默认值）：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。参见 TRTCVideoQosPreferenceClear

TRTCRenderParams

TRTCRenderParams

视频画面的渲染参数

您可以通过设置此参数来控制画面的旋转角度、填充模式和左右镜像模式。

枚举类型	描述
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTCVideoFillMode_Fill。
mirrorType	【字段含义】画面镜像模式。 【推荐取值】默认值：TRTCVideoMirrorTypeAuto。
rotation	【字段含义】图像的顺时针旋转角度。 【推荐取值】支持 90、180 以及 270 旋转角度，默认值：TRTCVideoRotation_0。

TRTCQualityInfo

TRTCQualityInfo

网络质量

表征网络质量的好坏，您可以通过该数值在用户界面上展示每个用户的网络质量。

枚举类型	描述
quality	网络质量
userId	用户 ID

TRTCVolumeInfo

TRTCVolumeInfo

音量大小

表征语音音量的评估值，您可以通过该数值在用户界面上展示每个用户的音量大小。

枚举类型	描述
pitch	本地用户的人声频率（单位：Hz），取值范围[0 - 4000]，对于远端用户，该值始终为0。
spectrumData	音频频谱数据是将音频数据在频率域中的分布，划分为 256 个频率段，使用 spectrumData 记录各个频率段的能量值，每个能量值的取值范围为 [-300, 0]，单位为 dBFS。 **注意** 本地频谱使用编码前的音频数据计算，会受到本地采集音量、BGM等影响；远端频谱使用接收到的音频数据计算，本地调整远端播放音量等操作不会对其产生影响。
userId	说话者的 userId, 如果 userId 为空则代表是当前用户自己。
vad	是否检测到人声，0：非人声 1：人声。
volume	说话者的音量大小, 取值范围[0 - 100]。

TRTCSpeedTestParams

TRTCSpeedTestParams

测速参数

您可以在用户进入房间前通过 startSpeedTest 接口测试网速（注意：请不要在通话中调用）。

枚举类型	描述
expectedDownBandwidth	预期的下行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
expectedUpBandwidth	预期的上行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
scene	测速场景。
sdkAppId	应用标识，请参见 TRTCParams 中的相关说明。
userId	用户标识，请参见 TRTCParams 中的相关说明。

userSig	用户签名，请参见 TRTCParams 中的相关说明。

TRTCSpeedTestResult

TRTCSpeedTestResult

网络测速结果

您可以在用户进入房间前通过 startSpeedTest 接口进行测速（注意：请不要在通话中调用）。

枚举类型	描述
availableDownBandwidth	下行带宽（kbps，-1：无效值）。
availableUpBandwidth	上行带宽（kbps，-1：无效值）。
downJitter	下行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。
downLostRate	下行丢包率，取值范围是 [0 - 1.0]，例如 0.2 表示每从服务器收取 10 个数据包可能会在中途丢失 2 个。
errMsg	带宽测试错误信息。
ip	服务器 IP 地址。
quality	内部通过评估算法测算出的网络质量，更多信息请参见 TRTCQuality。
rtt	延迟（毫秒），指当前设备到 TRTC 服务器的一次网络往返时间，该值越小越好，正常数值范围是10ms - 100ms。
success	测试是否成功。
upJitter	上行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。
upLostRate	上行丢包率，取值范围是 [0 - 1.0]，例如 0.3 表示每向服务器发送 10 个数据包可能会在中途丢失 3 个。

TRTCVideoFrame

TRTCVideoFrame

视频帧信息

TRTCVideoFrame 用来描述一帧视频画面的裸数据，也就是编码前或者解码后的视频画面数据。

枚举类型	描述
bufferType	【字段含义】视频数据结构类型。
data	【字段含义】bufferType 为 TRTCVideoBufferType_NSData 时的视频数据，承载 NSData 类型的内存数据块。
height	【字段含义】视频高度。
pixelBuffer	【字段含义】bufferType 为 TRTCVideoBufferType_PixelBuffer 时的视频数据，承载 iOS 平台专用的 PixelBuffer。
pixelFormat	【字段含义】视频的像素格式。
rotation	【字段含义】视频像素的顺时针旋转角度。
textureId	【字段含义】视频纹理 ID,bufferType 为 TRTCVideoBufferType_Texture 时的视频数据，承载用于 OpenGL 渲染的纹理数据。
timestamp	【字段含义】视频帧的时间戳，单位毫秒。 【推荐取值】自定义视频采集时可以设置为0。若该参数为0，SDK 会自定填充 timestamp 字段，但请“均匀”地控制 sendCustomVideoData 的调用间隔。
width	【字段含义】视频宽度。

TRTCAudioFrame

TRTCAudioFrame

音频帧数据

枚举类型	描述
channels	【字段含义】声道数。
data	【字段含义】音频数据。
extraData	【字段含义】音频额外数据，远端用户通过 `onLocalProcessedAudioFrame` 写入的数据会通过该字段回调。
sampleRate	【字段含义】采样率。
timestamp	【字段含义】时间戳，单位ms。

版权所有：腾讯云计算（北京）有限责任公司

TRTCMixUser

TRTCMixUser

云端混流中各路画面的描述信息

TRTCMixUser 用于指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
image	【字段含义】占位图或水印图 占位图是指当对应 userId 混流内容为纯音频时，混合后的画面中显示的是占位图片。 水印图是指一张贴在混合后画面中的半透明图片，这张图片会一直覆盖于混合后的画面上。 当指定 inputType 为 TRTCMixInputTypePureAudio 时，image 为占位图，此时需要您指定 userId。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，image 为水印图，此时不需要您指定 userId。 【推荐取值】默认值：空值，即不设置占位图或者水印图。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 “**控制台** => 应用管理 => 功能配置 => 素材管理” 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的“图片ID”，然后将“图片ID”转换成字符串类型并设置给 image 字段即可（比如假设“图片ID” 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。 image 仅在 inputType 为 TRTCMixInputTypePureAudio 或者 TRTCMixInputTypeWatermark 时才生效。
inputType	【字段含义】指定该路流的混合内容（只混音频、只混视频、混合音视频、混入水印）。 【默认取值】默认值：TRTCMixInputTypeUndefined。 【特别说明】 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 YES 时，等效于设置 inputType 为 TRTCMixInputTypePureAudio。 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 NO 时，等效于设置 inputType 为 TRTCMixInputTypeAudioVideo。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，您可以不指定 userId 字段，但需要指定 image 字段。
pureAudio	【字段含义】指定该路流是不是只混合声音。 【推荐取值】默认值：NO。 【特别说明】已废弃，推荐使用8.5版本开始新引入的字段：inputType。
rect	【字段含义】指定该路画面的坐标区域（单位：像素）。

renderMode	【字段含义】该画面在输出时的显示模式。 【推荐取值】默认值：视频流默认为0。0为裁剪，1为缩放，2为缩放并显示黑底。 【特别说明】水印图和占位图暂时不支持设置 renderMode，默认强制拉伸处理。
roomID	【字段含义】该路音视频流所在的房间号（设置为空值代表当前用户所在的房间号）。
soundLevel	【字段含义】该路音频参与混音时的音量等级（取值范围：0 - 100）。 【默认取值】默认值：100。
streamType	【字段含义】指定该路画面是主路画面（TRTCVideoStreamTypeBig）还是辅路画面（TRTCVideoStreamTypeSub）。
userId	【字段含义】用户 ID
zOrder	【字段含义】指定该路画面的层级（取值范围：1 - 15，不可重复）。

TRTCTranscodingConfig

TRTCTranscodingConfig

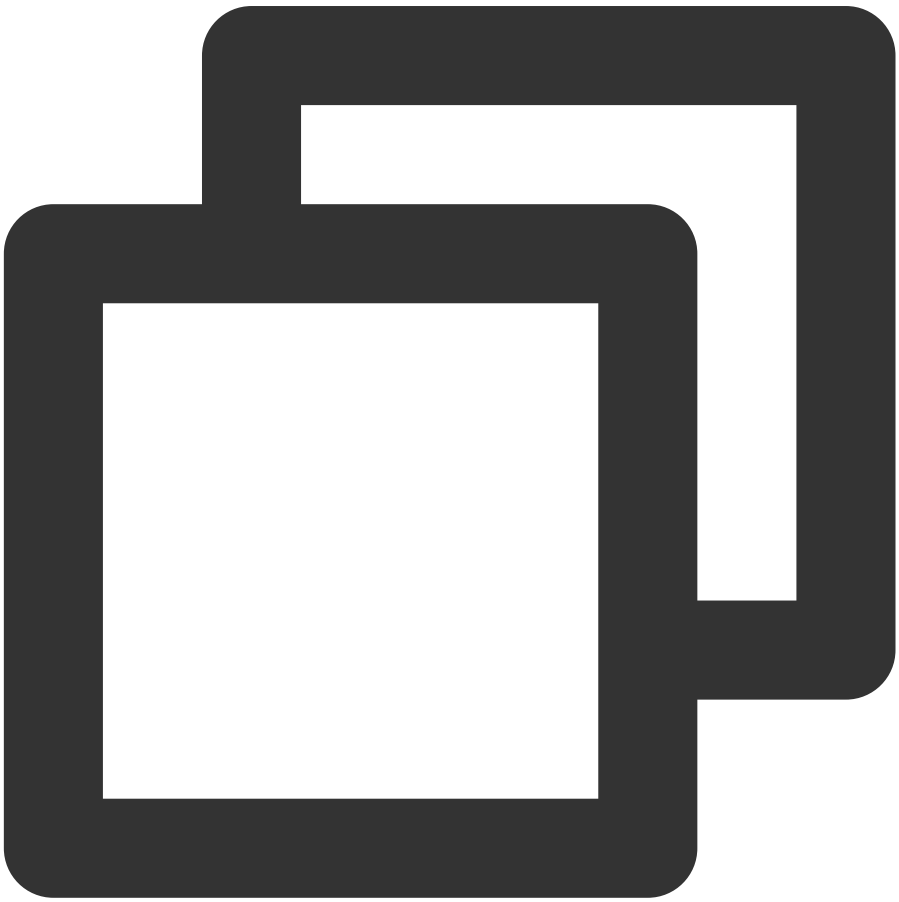
云端混流的排版布局和转码参数

用于指定混流时各路画面的排版位置信息和云端转码的编码参数。

枚举类型	描述
appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
audioBitrate	【字段含义】指定云端转码的目标音频码率。 【推荐取值】默认值：64kbps，取值范围是 [32，192]。
audioChannels	【字段含义】指定云端转码的音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioCodec	【字段含义】指定云端转码的输出流音频编码类型。 【推荐取值】默认值：0，代表LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。 【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。 【特别说明】HE-AAC 和 HE-AACv2 取值仅在输出流为您额外设置的 streamId 上时才生效。

audioSampleRate	【字段含义】指定云端转码的目标音频采样率。 【推荐取值】默认值：48000Hz。支持12000HZ、16000HZ、22050HZ、24000HZ、32000HZ、44100HZ、48000HZ。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如："0x61B9F1" 代表 RGB 分别为(97,158,241)。
backgroundImage	【字段含义】指定混合画面的背景图片。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 "**控制台** => 应用管理 => 功能配置 => 素材管理" 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的"图片ID"，然后将"图片ID"转换成字符串类型并设置给 image 字段即可（比如假设"图片ID" 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
mixUsers	【字段含义】指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。 【推荐取值】该字段是一个 TRTCMixUser 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
mode	【字段含义】排版模式。 【推荐取值】请根据您的业务场景要求自行选择，预排版模式是适用性较好的一种模式。
streamId	【字段含义】输出到 CDN 上的直播流 ID。 【推荐取值】默认值：空值，即房间里的多路音视频流最终会混合到接口调用者的那一路音视频流上。 如不设置该参数，SDK 会执行默认逻辑，即房间里的多路音视频流会混合到该接口调用者的那一路音视频流上，也就是 A + B => A。 如您设置该参数，SDK 会将房间里的多路音视频流混合到您指定的直播流上，也就是 A + B => C（C 代表您指定的 streamId）。
videoBitrate	【字段含义】指定云端转码的目标视频码率（kbps）。 【推荐取值】如果填0，TRTC 会根据 videoWidth 和 videoHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoFramerate	【字段含义】指定云端转码的目标视频帧率（FPS）。 【推荐取值】默认值：15fps，取值范围是 (0,30]。

版权所有：腾讯云计算（北京）有限责任公司

videoGOP	【字段含义】指定云端转码的目标视频关键帧间隔（GOP）。 【推荐取值】默认值：2，单位为秒，取值范围是 [1,8]。
videoHeight	【字段含义】指定云端转码的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：  `{ "payLoadContent":"xxx", "payloadType":5, "payloadUuid":"1234567890abcdef1234567890abcdef", "interval":1000, "followIdr":false }` 当前支持的字段及含义： payloadContent: 必填。透传 sei 的 payload 内容，不能为空；

	payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）； payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字； interval: 选填，默认 1000。sei 的发送间隔，单位毫秒； followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。
videoWidth	【字段含义】指定云端转码的目标分辨率（宽度）。 【推荐取值】单位：像素值，推荐值：360，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。

TRTCPublishCDNParam

TRTCPublishCDNParam

向非腾讯云 CDN 上发布音视频流时需设置的转推参数

TRTC 的后台服务支持通过标准 RTMP 协议，将其中的音视频流发布到第三方直播 CDN 服务商。
如果您使用腾讯云直播 CDN 服务，可无需关注此参数，直接使用 startPublish 接口即可。

枚举类型	描述
appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
streamId	【字段含义】需要转推的 streamId。 【推荐取值】默认值：空值。如果不填写，则默认转推调用者的旁路流。
url	【字段含义】指定该路音视频流在第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。 【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCAudioRecordingParams

版权所有：腾讯云计算（北京）有限责任公司

TRTCAudioRecordingParams

本地音频文件的录制参数

该参数用于在音频录制接口 startAudioRecording 中指定录制参数。

枚举类型	描述
filePath	【字段含义】录音文件的保存路径（必填）。 【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录音文件的格式，目前支持的格式有 PCM、WAV 和 AAC。 例如：假如您指定路径为 "mypath/record/audio.aac"，代表您希望 SDK 生成一个 AAC 格式的音频录制文件。 请您指定一个有读写权限的合法路径，否则录音文件无法生成。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值 10000。默认值为0，表示不分片。
recordingContent	【字段含义】音频录制内容类型。 【特别说明】默认录制所有本地和远端音频。

TRTCLocalRecordingParams

TRTCLocalRecordingParams

本地媒体文件的录制参数

该参数用于在本地媒体文件的录制接口 startLocalRecording 中指定录制相关参数。

接口 startLocalRecording 是接口 startAudioRecording 的能力加强版本，前者可以录制视频文件，后者只能录制音频文件。

枚举类型	描述
filePath	【字段含义】录制的文件地址（必填），请确保路径有读写权限且合法，否则录制文件无法生成。 【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录制出的文件格式，目前支持的格式暂时只有 MP4。 例如：假如您指定路径为 "mypath/record/test.mp4"，代表您希望 SDK 生成一个 MP4 格式的本地视频文件。 请您指定一个有读写权限的合法路径，否则录制文件无法生成。
interval	【字段含义】interval 录制信息更新频率，单位毫秒，有效范围：1000-10000。默认值为-1，表示不回调。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值 10000。默认值为0，表示不分片。

recordType	【字段含义】媒体录制类型，默认值：TRTCRecordTypeBoth，即同时录制音频和视频。

TRTCSwitchRoomConfig

TRTCSwitchRoomConfig

房间切换参数

该参数用于切换房间接口switchRoom，可以让用户从一个房间快速切换到另一个房间。

枚举类型	描述
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
roomId	【字段含义】数字房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
strRoomId	【字段含义】字符串房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
userSig	【字段含义】用户签名 [选填]，当前 userId 对应的验证签名，相当于登录密码。 如果您在切换房间时不指定新计算出的 userSig，SDK 会继续使用您在进入房间时（enterRoom）时所指定的 userSig。 这就需要您必须保证旧的 userSig 在切换房间的那一刻仍在签名允许的效期内，否则会导致房间切换失败。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCAudioFrameDelegateFormat

TRTCAudioFrameDelegateFormat

音频自定义回调的格式参数

该参数用于在音频自定义回调相关的接口中，设置 SDK 回调出来的音频数据的相关格式（包括采样率、声道数等）。

枚举类型	描述
channels	【字段含义】声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
mode	【字段含义】回调数据读写模式 【推荐取值】TRTCAudioFrameOperationModeReadOnly：仅从回调中获取音频数据。可设定的模式有 TRTCAudioFrameOperationModeReadOnly，TRTCAudioFrameOperationModeReadWrite。
sampleRate	【字段含义】采样率。 【推荐取值】默认值：48000Hz。支持 16000, 32000, 44100, 48000。
samplesPerCall	【字段含义】采样点数。 【推荐取值】取值必须是 sampleRate/100 的整数倍。

TRTCScreenCaptureSourceInfo

TRTCScreenCaptureSourceInfo

屏幕分享的目标信息（仅适用于桌面系统）

在用户进行屏幕分享时，可以选择抓取整个桌面，也可以仅抓取某个程序的窗口。
TRTCScreenCaptureSourceInfo 用于描述待分享目标的信息，包括 ID、名称、缩略图等，该结构体中的字段信息均是只读的。

枚举类型	描述
extInfo	【字段含义】窗口的扩展信息。
icon	【字段含义】分享窗口的图标。
sourceId	【字段含义】采集源的ID，对于窗口，该字段代表窗口的 ID；对于屏幕，该字段代表显示器的 ID。
sourceName	【字段含义】采集源名称（采用 UTF8 编码）。
thumbnail	【字段含义】分享窗口的缩略图。
type	【字段含义】采集源类型（是分享整个屏幕？还是分享某个窗口？）。

TRTCAudioParallelParams

版权所有：腾讯云计算（北京）有限责任公司

TRTCAudioParallelParams

远端音频流智能并发播放策略的参数

该参数用于设置远端音频流智能并发播放策略。

枚举类型	描述
includeUsers	【字段含义】指定用户必定能并发播放。 【特殊说明】指定必定并发播放的用户 ID 列表。这些用户不参与智能选择。 includeUsers 的数量必须小于 maxCount，否则本次并发播放设置失效。 includeUsers 仅在 maxCount > 0 时有效。当 includeUsers 生效时，参与智能并发选择的最大播放数 = maxCount - 有效 includeUsers 的数量。
maxCount	【字段含义】最大并发播放数。默认值：0。 如果 maxCount > 0，且实际人数 > maxCount，会实时智能选出 maxCount 路数据进行播放，这会极大的降低性能消耗。 如果 maxCount = 0，SDK 不限制并发播放数，在上麦人数比较多的房间可能会引发性能问题。

TRTCUser

TRTCUser

媒体流发布相关配置的用户信息

您可以通过设置该参数，配合媒体流目标发布参数 (TRTCPublishTarget) 和混流转码参数 (TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您填写的目标发布地址中

枚举类型	描述
intRoomId	【字段含义】数字房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 strRoomId，则 intRoomId 需要填写为0。若两者都填，SDK 将优先选用 intRoomId。
strRoomId	【字段含义】字符串房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 roomId，则 strRoomId 无需填写。若两者都填，SDK 将优先选用 intRoomId。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、 ! 、 # 、 $ 、 % 、 & 、 (、) 、 + 、 - 、 : 、 ; 、 < 、 = 、 . 、 > 、 ? 、 @ 、 [、] 、 ^ 、 _ 、 { 、 } 、 | 、 ~ 、 , 。

userId	【字段含义】用户标识，当前用户的 userId，相当于用户名，使用 UTF-8 编码。 【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。

TRTCPublishCdnUrl

TRTCPublishCdnUrl

向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置

该配置用于媒体流发布接口 startPublishMediaStream 中的目标推流配置 (TRTCPublishTarget)

枚举类型	描述
isInternalLine	【字段含义】指定该路音视频流是否发布至腾讯云。 【推荐取值】默认值：true。 【特别说明】若您的目标直播服务商为腾讯，请将此参数设置为 true，此时后台计费系统不会对此计算转推服务费。
rtmpUrl	【字段含义】指定该路音视频流在腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。 【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCPublishTarget

TRTCPublishTarget

目标推流配置

该配置用于媒体流发布接口 startPublishMediaStream。

枚举类型	描述
cdnUrlList	【字段含义】发布至腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【特别说明】若您的 mode 选择为 TRTCPublishMixStreamToRoom，此时您不需要设置该参数。
mixStreamIdentity	【字段含义】回推房间机器人信息。 【特别说明】仅当您的 mode 选择为 TRTCPublishMixStreamToRoom 时，您需要设置该参数。

版权所有：腾讯云计算（北京）有限责任公司

	【特别说明】设置后，该路转码音视频数据将回推到您指定的房间中。建议设置为特殊的 userId，以避免难以区分回推机器人和您通过 TRTC SDK 进房的主播。 【特别说明】参与混流的用户不支持订阅该转码流。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为手动时，您需要自行管理您想要拉取的音视频流（通常当您拉取回推房间的转码流时，您应该取消订阅参与转码的对应音视频单流）。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为自动时，不参与转码的用户将自动收到后台下发的转码流并不再继续接收参与转码的音视频单流。除非您明确进行取消订阅（muteRemoteVideoStream 和 muteRemoteAudio），否则转码流数据将持续下发。
mode	【字段含义】媒体流发布模式。 【推荐取值】请根据您的业务场景要求自行选择，TRTC 支持转推、转码和回推到 RTC 房间的模式。 【特别说明】若您的业务场景需要多个发布模式，您可以通过多次调用媒体流发布接口 (startPublishMediaStream) 并分别设置不同的 TRTCPublishTarget。 【特别说明】同一个 mode 请对应一个媒体流发布接口 (startPublishMediaStream)，并在后续需要调整时，使用 updatePublishCDNStream 进行更新。

TRTCVideoLayout

TRTCVideoLayout

转码视频布局

该配置用于媒体流发布接口（startPublishMediaStream）中的转码配置（TRTCStreamMixingConfig）。
用于指定转码流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为 (97,158,241)。
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTCVideoFillMode_Fill。
fixedVideoStreamType	【字段含义】指定该路画面是主路画面（TRTCVideoStreamTypeBig）还是辅路画面（TRTCVideoStreamTypeSub）。
fixedVideoUser	【字段含义】参与转码的用户信息。 【特别说明】用户信息 (TRTCUser) 支持不填写（即 userId、intRoomId 和 strRoomId 均不填写）。此时当发起混流的房间中有主播上行音视频数据时，

	TRTC 后台服务器将自动将对应主播音视频填充到您指定的布局中。
placeHolderImage	【字段含义】占位图 URL，即当指定用户暂时仅上行音频时，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置占位图。 【特别说明】此时需要您指定用户信息 fixedVideoUser 中的 userId。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为占位图。
rect	【字段含义】指定该路画面的坐标区域（单位：像素）。
zOrder	【字段含义】指定该路画面的层级（取值范围：0 - 15，不可重复）。

TRTCWatermark

TRTCWatermark

水印布局

该配置用于媒体流发布接口 (startPublishMediaStream) 中的转码配置 (TRTCStreamMixingConfig)

枚举类型	描述
rect	【字段含义】指定该路水印画面的坐标区域（单位：像素）。
watermarkUrl	【字段含义】水印 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。
zOrder	【字段含义】指定该路水印画面的层级（取值范围：0 - 15，不可重复）。

TRTCStreamEncoderParam

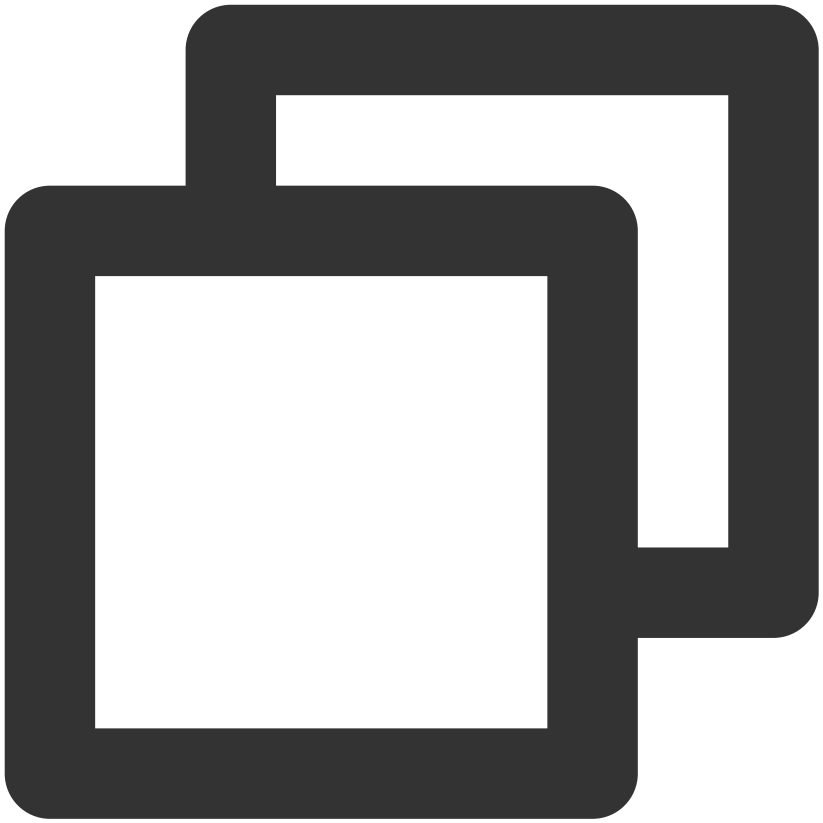
TRTCStreamEncoderParam

媒体流编码输出参数

【字段含义】该配置用于媒体流发布接口（startPublishMediaStream）。

【特别说明】当您的发布目标（TRTCPublishTarget）中的 mode 配置为 TRTCPublish_MixStream_ToCdn 或者 TRTCPublish_MixStream_ToRoom 时，该参数为必填。
【特别说明】当您使用转推服务（mode 为 TRTCPublish_BigStream_ToCdn 或者 TRTCPublish_SubStream_ToCdn）时，为了更好的转推稳定性以及更好的 CDN 播放兼容性，也建议您填写该配置的具体参数。

枚举类型	描述
audioEncodedChannelNum	【字段含义】指定媒体发布流的目标音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioEncodedCodecType	【字段含义】指定媒体发布流的目标音频编码类型。 【推荐取值】默认值：0，代表 LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。 【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。
audioEncodedKbps	【字段含义】指定媒体发布流的目标音频码率（kbps）。 【推荐取值】默认值：50kbps，取值范围是 [32，192]。
audioEncodedSampleRate	【字段含义】指定媒体发布流的目标音频采样率。 【推荐取值】默认值：48000Hz。取值为 [48000, 44100, 32000, 24000, 16000, 8000]，单位是 Hz。
videoEncodedCodecType	【字段含义】指定媒体发布流的目标视频编码类型。 【推荐取值】默认值：0，代表 H264。可设定的数值只有两个数字：0 - H264，1 - H265。
videoEncodedFPS	【字段含义】指定媒体发布流的目标视频帧率（FPS）。 【推荐取值】推荐值：20fps，取值范围是 (0,30]。
videoEncodedGOP	【字段含义】指定媒体发布流的目标视频关键帧间隔（GOP）。 【推荐取值】推荐值：3，单位为秒，取值范围是 [1,5]。
videoEncodedHeight	【字段含义】指定媒体发布流的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoEncodedKbps	【字段含义】指定媒体发布流的目标视频码率（kbps）。 【推荐取值】如果填 0，TRTC 会根据 videoEncodedWidth 和 videoEncodedHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoEncodedWidth	【字段含义】指定媒体发布流的目标分辨率（宽度）。

	【推荐取值】单位：像素值，推荐值：368，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：  `{ "payLoadContent":"xxx", "payloadType":5, "payloadUuid":"1234567890abcdef1234567890abcdef", "interval":1000, "followIdr":false }` 当前支持的字段及含义： payloadContent: 必填。透传 sei 的 payload 内容，不能为空； payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）； payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字； interval: 选填，默认 1000。sei 的发送间隔，单位毫秒； followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。

TRTCStreamMixingConfig

TRTCStreamMixingConfig

媒体流转码配置参数

该配置用于媒体流发布接口（startPublishMediaStream）。

用于指定转码时各路画面的排版位置信息和输入的音频信息。

枚举类型	描述
audioMixUserList	【字段含义】指定转码流中的每一路输入音频的信息。 【推荐取值】该字段是一个 TRTCUser 类型的数组，数组中的每一个元素都用来代表每一路输入音频的信息。 【特别说明】用户信息支持不填写（即 audioMixUserList 为空）。此时若设置了 TRTCStreamEncoderParam 中音频相关编码输出参数，TRTC 后台服务器将自动将所有主播的音频混合输出（当前仅支持最高 16 路音视频输入）。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为 （97、158、241）。
backgroundImage	【字段含义】指定混合画面的背景图 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为背景图。
videoLayoutList	【字段含义】指定混合画面中的每一路视频画面的位置、大小、图层以及流类型等信息。 【推荐取值】该字段是一个 TRTCVideoLayout 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
watermarkList	【字段含义】指定混合画面中的每一路水印画面的位置、大小、图层等信息。 【推荐取值】该字段是一个 TRTCWatermark 类型的数组，数组中的每一个元素都用来代表每一路水印的信息。

TRTCPayloadPrivateEncryptionConfig

TRTCPayloadPrivateEncryptionConfig

媒体流私有加密配置

该配置用于设置媒体流私有加密的算法和密钥。

枚举类型	描述
encryptionAlgorithm	【字段含义】加密算法，默认为 TRTCEncryptionAlgorithmAes128Gcm。
encryptionKey	【字段含义】加密用密钥，字符串类型。 【推荐取值】若加密算法为 TRTCEncryptionAlgorithmAes128Gcm，密匙长度需为 16 字节，若加密算法为 TRTCEncryptionAlgorithmAes256Gcm，密匙长度需为 32 字节。
encryptionSalt	【字段含义】盐，加密用初始向量。 【推荐取值】需确保填入该参数的数组不为空、不全为 0 且数据长度为 32 字节。

TRTCAudioVolumeEvaluateParams

TRTCAudioVolumeEvaluateParams

音量评估等相关参数设置

该设置用于开启人声检测、声音频谱计算。

枚举类型	描述
enablePitchCalculation	【字段含义】是否开启本地人声频率计算
enableSpectrumCalculation	【字段含义】是否开启声音频谱计算。
enableVadDetection	【字段含义】是否开启本地人声检测。 【请您注意】在 startLocalAudio 之前调用才可以生效。
interval	【字段含义】设置 onUserVoiceVolume 回调的触发间隔，单位为毫秒，最小间隔为 100ms，如果小于等于 0 则会关闭回调。 【推荐取值】推荐值：300，单位为毫秒。 【特别说明】interval 大于 0 时，音量提示将默认开启，无需另外设置。

版权所有：腾讯云计算（北京）有限责任公司

废弃接口

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2022 Tencent. All rights reserved.

Module: DeprecatedTRTCCloud @ TXLiteAVSDK

Function: TRTC 废弃接口

废弃接口

TRTCCloud

函数列表	描述
destroySharedIntance	销毁 TRTCCloud 实例（单例模式）
delegate	设置 TRTC 事件回调
setBeautyStyle:beautyLevel:whitenessLevel:ruddinessLevel:	设置美颜、美白以及红润效果级别
setEyeScaleLevel:	设置大眼级别
setFaceScaleLevel:	设置瘦脸级别
setFaceVLevel:	设置 V 脸级别
setChinLevel:	设置下巴拉伸或收缩幅度
setFaceShortLevel:	设置短脸级别
setNoseSlimLevel:	设置瘦鼻级别
selectMotionTmpl:	设置动效贴纸
setMotionMute:	设置动效静音
setFilter:	设置色彩滤镜效果
setFilterConcentration:	设置色彩滤镜浓度
setGreenScreenFile:	设置绿幕背景视频

setReverbType:	设置混响效果
setVoiceChangerType:	设置变声类型
enableAudioEarMonitoring:	开启（或关闭）耳返
enableAudioVolumeEvaluation:	启用音量大小提示
enableAudioVolumeEvaluation:enable_vad:	启用音量大小提示
switchCamera	切换摄像头
isCameraZoomSupported	查询当前摄像头是否支持缩放
setZoom:	设置摄像头缩放倍数（焦距）
isCameraTorchSupported	查询是否支持开关闪光灯
enbaleTorch:	开关/关闭闪光灯
isCameraFocusPositionInPreviewSupported	查询摄像头是否支持设置焦点
setFocusPosition:	设置摄像头焦点坐标位置
isCameraAutoFocusFaceModeSupported	查询是否支持自动识别人脸位置
enableAutoFaceFoucs:	开启/关闭人脸跟踪对焦
setSystemVolumeType:	设置系统音量类型
snapshotVideo:type:	视频截图
startScreenCaptureByReplaykit:appGroup:	开始全系统的屏幕分享（iOS）
startLocalAudio	设置音频质量
startRemoteView:view:	开始显示远端视频画面
stopRemoteView:	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode:	设置本地图像的渲染模式
setLocalViewRotation:	设置本地图像的顺时针旋转角度
setLocalViewMirror:	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode:mode:	设置远端图像的渲染模式
setRemoteViewRotation:rotation:	设置远端图像的顺时针旋转角度

版权所有：腾讯云计算（北京）有限责任公司

startRemoteSubStreamView:view:	开始显示远端用户的辅路画面
stopRemoteSubStreamView:	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode:mode:	设置辅路画面的填充模式
setRemoteSubStreamViewRotation:rotation:	设置辅路画面的顺时针旋转角度
setAudioQuality:	设置音频质量
setPriorRemoteVideoStreamType:	设定优先观看大画面还是小画面
setMicVolumeOnMixing:	设置麦克风音量大小
playBGM:	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration:	获取背景音乐总时长（单位：毫秒）
setBGMPosition:	设置背景音乐的播放进度
setBGMVolume:	设置背景音乐的音量大小
setBGMPlayoutVolume:	设置背景音乐的本地播放音量
setBGMPublishVolume:	设置背景音乐的远端播放音量
playAudioEffect:	播放音效
setAudioEffectVolume:volume:	设置音效音量
stopAudioEffect:	停止播放音效
stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume:	设置所有音效音量
pauseAudioEffect:	暂停音效
resumeAudioEffect:	暂停音效
enableCustomVideoCapture:	启用视频自定义采集模式
sendCustomVideoData:	投送自己采集的视频数据

版权所有：腾讯云计算（北京）有限责任公司

muteLocalVideo:	暂停/恢复发布本地的视频流
muteRemoteVideoStream:mute:	暂停 / 恢复订阅远端用户的视频流
startSpeedTest:userId:userSig:	开始进行网络测速（进入房间前使用）
startScreenCapture:	启动屏幕分享
getCameraDevicesList	获取摄像头设备列表
setCurrentCameraDevice:	选定当前要使用的摄像头
getCurrentCameraDevice	获取当前使用的摄像头
getMicDevicesList	获取麦克风设备列表
getCurrentMicDevice	获取当前的麦克风设备
setCurrentMicDevice:	选定当前使用的麦克风
getCurrentMicDeviceVolume	获取当前麦克风的设备音量
setCurrentMicDeviceVolume:	设置当前麦克风的设备音量
setCurrentMicDeviceMute:	设置系统当前麦克风设备的静音状态
getCurrentMicDeviceMute	获取系统当前麦克风设备是否静音
getSpeakerDevicesList	获取扬声器设备列表
getCurrentSpeakerDevice	获取当前的扬声器设备
setCurrentSpeakerDevice:	设置要使用的扬声器
getCurrentSpeakerDeviceVolume	获取当前扬声器的设备音量
setCurrentSpeakerDeviceVolume:	设置当前扬声器的设备音量
getCurrentSpeakerDeviceMute	获取系统当前扬声器设备是否静音
setCurrentSpeakerDeviceMute:	设置系统当前扬声器设备的静音状态
startCameraDeviceTestInView:	开始进行摄像头测试
stopCameraDeviceTest	停止进行摄像头测试
startMicDeviceTest:	开始进行麦克风测试
stopMicDeviceTest	开始进行麦克风测试

版权所有：腾讯云计算（北京）有限责任公司

startSpeakerDeviceTest:	开始进行扬声器测试
stopSpeakerDeviceTest	停止进行扬声器测试
startScreenCaptureInApp:	开始应用内的屏幕分享（iOS）
setVideoEncoderRotation:	设置视频编码器输出的画面方向
setVideoEncoderMirror:	设置编码器输出的画面镜像模式
setGSensorMode:	设置重力感应的适配模式

destroySharedIntance

destroySharedIntance

销毁 TRTCCloud 实例（单例模式）

@deprecated 11.5 版本不推荐使用 建议使用 destroySharedInstance

delegate

delegate

设置 TRTC 事件回调

@deprecated v11.4 版本开始不推荐使用，建议使用 addDelegate 替代之。

setBeautyStyle:beautyLevel:whitenessLevel:ruddinessLevel:

setBeautyStyle:beautyLevel:whitenessLevel:ruddinessLevel:

- (void)setBeautyStyle:	(TRTCBeautyStyle)beautyStyle
beautyLevel:	(NSInteger)beautyLevel
whitenessLevel:	(NSInteger)whitenessLevel
ruddinessLevel:	(NSInteger)ruddinessLevel

设置美颜、美白以及红润效果级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

版权所有：腾讯云计算（北京）有限责任公司

setEyeScaleLevel:

setEyeScaleLevel:

- (void)setEyeScaleLevel:	(float)eyeScaleLevel

设置大眼级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFaceScaleLevel:

setFaceScaleLevel:

- (void)setFaceScaleLevel:	(float)faceScaleLevel

设置瘦脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFaceVLevel:

setFaceVLevel:

- (void)setFaceVLevel:	(float)faceVLevel

设置 V 脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setChinLevel:

setChinLevel:

- (void)setChinLevel:	(float)chinLevel

设置下巴拉伸或收缩幅度

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

版权所有：腾讯云计算（北京）有限责任公司

setFaceShortLevel:

setFaceShortLevel:

- (void)setFaceShortLevel:	(float)faceShortlevel

设置短脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setNoseSlimLevel:

setNoseSlimLevel:

- (void)setNoseSlimLevel:	(float)noseSlimLevel

设置瘦鼻级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

selectMotionTmpl:

selectMotionTmpl:

- (void)selectMotionTmpl:	(NSString *)tmplPath

设置动效贴纸

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setMotionMute:

setMotionMute:

- (void)setMotionMute:	(BOOL)motionMute

设置动效静音

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

版权所有：腾讯云计算（北京）有限责任公司

setFilter:

setFilter:

- (void)setFilter:	(TXImage *)image

设置色彩滤镜效果

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFilterConcentration:

setFilterConcentration:

- (void)setFilterConcentration:	(float)concentration

设置色彩滤镜浓度

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setGreenScreenFile:

setGreenScreenFile:

- (void)setGreenScreenFile:	(NSURL *)file

设置绿幕背景视频

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setReverbType:

setReverbType:

- (void)setReverbType:	(TRTCReverbType)reverbType

设置混响效果

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceReverbType 替代之。

版权所有：腾讯云计算（北京）有限责任公司

setVoiceChangerType:

setVoiceChangerType:

- (void)setVoiceChangerType:	(TRTCVoiceChangerType)voiceChangerType

设置变声类型

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceChangerType 替代之。

enableAudioEarMonitoring:

enableAudioEarMonitoring:

- (void)enableAudioEarMonitoring:	(BOOL)enable

开启（或关闭）耳返

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceEarMonitor 替代之。

enableAudioVolumeEvaluation:

enableAudioVolumeEvaluation:

- (void)enableAudioVolumeEvaluation:	(NSUInteger)interval

启用音量大小提示

@deprecated v10.1 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

enableAudioVolumeEvaluation:enable_vad:

enableAudioVolumeEvaluation:enable_vad:

- (void)enableAudioVolumeEvaluation:	(NSUInteger)interval
enable_vad:	(BOOL)enable_vad

启用音量大小提示

@deprecated v11.2 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

switchCamera

switchCamera

切换摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 switchCamera 接口替代之。

isCameraZoomSupported

isCameraZoomSupported

查询当前摄像头是否支持缩放

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isCameraZoomSupported 接口替代之。

setZoom:

setZoom:

- (void)setZoom:	(CGFloat)distance

设置摄像头缩放倍数（焦距）

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCameraZoomRatio 接口替代之。

isCameraTorchSupported

isCameraTorchSupported

查询是否支持开关闪光灯

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isCameraTorchSupported 接口替代之。

enbaleTorch:

enbaleTorch:

- (BOOL)enbaleTorch:	(BOOL)enable

版权所有：腾讯云计算（北京）有限责任公司

开关/关闭闪光灯

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 enableCameraTorch 接口替代之。

isCameraFocusPositionInPreviewSupported

isCameraFocusPositionInPreviewSupported

查询摄像头是否支持设置焦点

@deprecated v8.0 版本开始不推荐使用。

setFocusPosition:

setFocusPosition:

- (void)setFocusPosition:	(CGPoint)touchPoint

设置摄像头焦点坐标位置

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCameraFocusPosition 接口替代之。

isCameraAutoFocusFaceModeSupported

isCameraAutoFocusFaceModeSupported

查询是否支持自动识别人脸位置

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isAutoFocusEnabled 接口替代之。

enableAutoFaceFoucs:

enableAutoFaceFoucs:

- (void)enableAutoFaceFoucs:	(BOOL)enable

开启/关闭人脸跟踪对焦

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 enableCameraAutoFocus 接口替代之。

setSystemVolumeType:

setSystemVolumeType:

- (void)setSystemVolumeType:	(TRTCSystemVolumeType)type

设置系统音量类型

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio(quality) 替代之，通过 quality 参数来决策音质。

snapshotVideo:type:

snapshotVideo:type:

- (void)snapshotVideo:	(NSString *)userId
type:	(TRTCVideoStreamType)streamType

视频截图

@deprecated v8.2 版本开始不推荐使用，建议使用 snapshotVideo 替代之。

startScreenCaptureByReplaykit:appGroup:

startScreenCaptureByReplaykit:appGroup:

- (void)startScreenCaptureByReplaykit:	(TRTCVideoEncParam *)encParams
appGroup:	(NSString *)appGroup

开始全系统的屏幕分享（iOS）

@deprecated v8.6 版本开始不推荐使用，建议使用 startScreenCaptureByReplaykit 接口替代之。

startLocalAudio

startLocalAudio

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

startRemoteView:view:

startRemoteView:view:

- (void)startRemoteView:	(NSString *)userId
view:	(TXView *)view

开始显示远端视频画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteView:

stopRemoteView:

- (void)stopRemoteView:	(NSString *)userId

停止显示远端视频画面，同时不再拉取该远端用户的视频数据流

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setLocalViewFillMode:

setLocalViewFillMode:

- (void)setLocalViewFillMode:	(TRTCVideoFillMode)mode

设置本地图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setLocalViewRotation:

setLocalViewRotation:

- (void)setLocalViewRotation:	(TRTCVideoRotation)rotation

设置本地图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

版权所有：腾讯云计算（北京）有限责任公司

setLocalViewMirror:

setLocalViewMirror:

- (void)setLocalViewMirror:	(TRTCLocalVideoMirrorType)mirror

设置本地摄像头预览画面的镜像模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setRemoteViewFillMode:mode:

setRemoteViewFillMode:mode:

- (void)setRemoteViewFillMode:	(NSString*)userId
mode:	(TRTCVideoFillMode)mode

设置远端图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteViewRotation:rotation:

setRemoteViewRotation:rotation:

- (void)setRemoteViewRotation:	(NSString*)userId
rotation:	(TRTCVideoRotation)rotation

设置远端图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

startRemoteSubStreamView:view:

startRemoteSubStreamView:view:

- (void)startRemoteSubStreamView:	(NSString *)userId
view:	(TXView *)view

开始显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteSubStreamView:

stopRemoteSubStreamView:

- (void)stopRemoteSubStreamView:	(NSString *)userId

停止显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setRemoteSubStreamViewFillMode:mode:

setRemoteSubStreamViewFillMode:mode:

- (void)setRemoteSubStreamViewFillMode:	(NSString *)userId
mode:	(TRTCVideoFillMode)mode

设置辅路画面的填充模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteSubStreamViewRotation:rotation:

setRemoteSubStreamViewRotation:rotation:

- (void)setRemoteSubStreamViewRotation:	(NSString*)userId
rotation:	(TRTCVideoRotation)rotation

设置辅路画面的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams:streamType:params: 替代之。

setAudioQuality:

setAudioQuality:

- (void)setAudioQuality:	(TRTCAudioQuality)quality

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

setPriorRemoteVideoStreamType:

setPriorRemoteVideoStreamType:

- (void)setPriorRemoteVideoStreamType:	(TRTCVideoStreamType)streamType

设定优先观看大画面还是小画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView:streamType:view: 替代之。

setMicVolumeOnMixing:

setMicVolumeOnMixing:

- (void)setMicVolumeOnMixing:	(NSInteger)volume

设置麦克风音量大小

playBGM:

playBGM:

- (void) playBGM:	(NSString *)path

启动播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

stopBGM

stopBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

pauseBGM

pauseBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

resumeBGM

resumeBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

getBGMDuration:

getBGMDuration:

- (NSInteger)getBGMDuration:	(NSString *)path

获取背景音乐总时长（单位：毫秒）

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 getMusicDurationInMS 替代之。

setBGMPosition:

setBGMPosition:

- (int)setBGMPosition:	(NSInteger)pos

设置背景音乐的播放进度

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 seekMusicToPosInMS 替代之。

setBGMVolume:

setBGMVolume:

- (void)setBGMVolume:	(NSInteger)volume

设置背景音乐的音量大小

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicVolume 替代之。

setBGMPlayoutVolume:

setBGMPlayoutVolume:

- (void)setBGMPlayoutVolume:	(NSInteger)volume

设置背景音乐的本地播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPlayoutVolume 替代之。

setBGMPublishVolume:

setBGMPublishVolume:

- (void)setBGMPublishVolume:	(NSInteger)volume

设置背景音乐的远端播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setBGMPublishVolume 替代之。

playAudioEffect:

playAudioEffect:

- (void)playAudioEffect:	(TRTCAudioEffectParam*)effect

播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 startPlayMusic 替代之。

setAudioEffectVolume:volume:

setAudioEffectVolume:volume:

- (void)setAudioEffectVolume:	(int)effectId
volume:	(int) volume

设置音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

stopAudioEffect:

stopAudioEffect:

- (void)stopAudioEffect:	(int)effectId

停止播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

stopAllAudioEffects

stopAllAudioEffects

停止所有音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

setAllAudioEffectsVolume:

setAllAudioEffectsVolume:

- (void)setAllAudioEffectsVolume:	(int)volume

设置所有音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

pauseAudioEffect:

pauseAudioEffect:

- (void)pauseAudioEffect:	(int)effectId

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 pauseAudioEffect 替代之。

resumeAudioEffect:

resumeAudioEffect:

- (void)resumeAudioEffect:	(int)effectId

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 resumePlayMusic 替代之。

enableCustomVideoCapture:

enableCustomVideoCapture:

- (void)enableCustomVideoCapture:	(BOOL)enable

启用视频自定义采集模式

@deprecated v8.5 版本开始不推荐使用，建议使用 enableCustomVideoCapture(streamType, enable) 接口替代之。

sendCustomVideoData:

sendCustomVideoData:

- (void)sendCustomVideoData:	(TRTCVideoFrame *)frame

投送自己采集的视频数据

@deprecated v8.5 版本开始不推荐使用，建议使用 sendCustomVideoData(streamType, TRTCVideoFrame) 接口替代之。

版权所有：腾讯云计算（北京）有限责任公司

muteLocalVideo:

muteLocalVideo:

- (void)muteLocalVideo:	(BOOL)mute

暂停/恢复发布本地的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteLocalVideo(streamType, mute) 接口替代之。

muteRemoteVideoStream:mute:

muteRemoteVideoStream:mute:

- (void)muteRemoteVideoStream:	(NSString*)userId
mute:	(BOOL)mute

暂停 / 恢复订阅远端用户的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteRemoteVideoStream(userId, streamType, mute) 接口替代之。

startSpeedTest:userId:userSig:

startSpeedTest:userId:userSig:

- (void)startSpeedTest:	(uint32_t)sdkAppId
userId:	(NSString *)userId
userSig:	(NSString *)userSig

开始进行网络测速（进入房间前使用）

@deprecated v9.2 版本开始不推荐使用，建议使用 startSpeedTest(params) 接口替代之。

startScreenCapture:

startScreenCapture:

- (void)startScreenCapture:	(nullable NSView *)view

启动屏幕分享

@deprecated v7.2 版本开始不推荐使用，建议使用 startScreenCapture 替代之。

getCameraDevicesList

getCameraDevicesList

获取摄像头设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

setCurrentCameraDevice:

setCurrentCameraDevice:

- (int)setCurrentCameraDevice:	(NSString *)deviceId

选定当前要使用的摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentCameraDevice

getCurrentCameraDevice

获取当前使用的摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

getMicDevicesList

getMicDevicesList

获取麦克风设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

版权所有：腾讯云计算（北京）有限责任公司

getCurrentMicDevice

getCurrentMicDevice

获取当前的麦克风设备

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

setCurrentMicDevice:

setCurrentMicDevice:

- (int)setCurrentMicDevice:	(NSString*)deviceId

选定当前使用的麦克风

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentMicDeviceVolume

getCurrentMicDeviceVolume

获取当前麦克风的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceVolume 接口替代之。

setCurrentMicDeviceVolume:

setCurrentMicDeviceVolume:

- (void)setCurrentMicDeviceVolume:	(NSInteger)volume

设置当前麦克风的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceVolume 接口替代之。

setCurrentMicDeviceMute:

setCurrentMicDeviceMute:

- (void)setCurrentMicDeviceMute:	(BOOL)mute

设置系统当前麦克风设备的静音状态

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceMute 接口替代之。

getCurrentMicDeviceMute

getCurrentMicDeviceMute

获取系统当前麦克风设备是否静音

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceMute 接口替代之。

getSpeakerDevicesList

getSpeakerDevicesList

获取扬声器设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

getCurrentSpeakerDevice

getCurrentSpeakerDevice

获取当前的扬声器设备

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

setCurrentSpeakerDevice:

setCurrentSpeakerDevice:

- (int)setCurrentSpeakerDevice:	(NSString*)deviceId

设置要使用的扬声器

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentSpeakerDeviceVolume

getCurrentSpeakerDeviceVolume

获取当前扬声器的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceVolume 接口替代之。

setCurrentSpeakerDeviceVolume:

setCurrentSpeakerDeviceVolume:

- (int)setCurrentSpeakerDeviceVolume:	(NSInteger)volume

设置当前扬声器的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceVolume 接口替代之。

getCurrentSpeakerDeviceMute

getCurrentSpeakerDeviceMute

获取系统当前扬声器设备是否静音

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceMute 接口替代之。

setCurrentSpeakerDeviceMute:

setCurrentSpeakerDeviceMute:

- (void)setCurrentSpeakerDeviceMute:	(BOOL)mute

设置系统当前扬声器设备的静音状态

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceMute 接口替代之。

startCameraDeviceTestInView:

startCameraDeviceTestInView:

- (void)startCameraDeviceTestInView:	(NSView *)view

开始进行摄像头测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startCameraDeviceTest 接口替代之。

stopCameraDeviceTest

stopCameraDeviceTest

停止进行摄像头测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopCameraDeviceTest 接口替代之。

startMicDeviceTest:

startMicDeviceTest:

- (void)startMicDeviceTest:	(NSInteger)interval

开始进行麦克风测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startMicDeviceTest 接口替代之。

stopMicDeviceTest

stopMicDeviceTest

开始进行麦克风测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopMicDeviceTest 接口替代之。

startSpeakerDeviceTest:

startSpeakerDeviceTest:

- (void)startSpeakerDeviceTest:	(NSString*)audioFilePath

开始进行扬声器测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startSpeakerDeviceTest 接口替代之。

版权所有：腾讯云计算（北京）有限责任公司

stopSpeakerDeviceTest

stopSpeakerDeviceTest

停止进行扬声器测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopSpeakerDeviceTest 接口替代之。

startScreenCaptureInApp:

startScreenCaptureInApp:

- (void)startScreenCaptureInApp:	(TRTCVideoEncParam *)encParams

开始应用内的屏幕分享（iOS）

@deprecated v8.6 版本开始不推荐使用，建议使用 startScreenCaptureInApp 接口替代之。

setVideoEncoderRotation:

setVideoEncoderRotation:

- (void)setVideoEncoderRotation:	(TRTCVideoRotation)rotation

设置视频编码器输出的画面方向

@deprecated v11.7 版本开始不推荐使用。

setVideoEncoderMirror:

setVideoEncoderMirror:

- (void)setVideoEncoderMirror:	(BOOL)mirror

设置编码器输出的画面镜像模式

@deprecated v11.7 版本开始不推荐使用。

setGSensorMode:

setGSensorMode:

- (void)setGSensorMode:	(TRTCGSensorMode) mode

设置重力感应的适配模式

@deprecated v11.7 版本开始不推荐使用。建议使用 setGravitySensorAdaptiveMode 接口替代之。

错误码表

最近更新时间：2024-03-07 15:33:58

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 错误码表

Function: 用于通知客户 TRTC 在使用过程中出现的警告和错误

iOS&Mac 的错误码，请查阅：全平台 C++ 错误码表

Android API 概览

最近更新时间：2024-06-06 15:26:14

API OVERVIEW

创建实例和事件回调

函数列表	描述
sharedInstance	创建 TRTCCloud 实例（单例模式）
destroySharedInstance	销毁 TRTCCloud 实例（单例模式）
addListener	添加 TRTC 事件回调
removeListener	移除 TRTC 事件回调
setListenerHandler	设置驱动 TRTCCloudListener 事件回调的队列

房间相关接口函数

函数列表	描述
enterRoom	进入房间
exitRoom	离开房间
switchRole	切换角色
switchRoom	切换房间
ConnectOtherRoom	请求跨房通话
DisconnectOtherRoom	退出跨房通话
setDefaultStreamRecvMode	设置订阅模式（需要在进入房前设置才能生效）
createSubCloud	创建子房间实例（用于多房间并发观看）
destroySubCloud	销毁子房间实例

版权所有：腾讯云计算（北京）有限责任公司

updateOtherRoomForwardMode	更改跨房主播在本房间的上行能力

CDN 相关接口函数

函数列表	描述
startPublishing	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig	设置云端混流的排版布局和转码参数
startPublishMediaStream	开始发布媒体流
updatePublishMediaStream	更新发布媒体流
stopPublishMediaStream	停止发布媒体流

视频相关接口函数

函数列表	描述
startLocalPreview	开启本地摄像头的预览画面（移动端）
updateLocalView	更新本地摄像头的预览画面
stopLocalPreview	停止摄像头预览
muteLocalVideo	暂停/恢复发布本地的视频流
setVideoMuteImage	设置本地画面被暂停期间的替代图片
startRemoteView	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView	更新远端用户的视频渲染控件
stopRemoteView	停止订阅远端用户的视频流，并释放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源

版权所有：腾讯云计算（北京）有限责任公司

muteRemoteVideoStream	暂停/恢复订阅远端用户的视频流
muteAllRemoteVideoStreams	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam	设置视频编码器的编码参数
setNetworkQosParam	设置网络质量控制的相关参数
setLocalRenderParams	设置本地画面的渲染参数
setRemoteRenderParams	设置远端画面的渲染模式
enableEncSmallVideoStream	开启大小画面双路编码模式
setRemoteVideoStreamType	切换指定远端用户的大小画面
snapshotVideo	视频画面截图
setPerspectiveCorrectionPoints	视频画面透视校正坐标设置
setGravitySensorAdaptiveMode	设置重力感应的适配模式（11.7 及以上版本）

音频相关接口函数

函数列表	描述
startLocalAudio	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布
muteLocalAudio	暂停/恢复发布本地的音频流
muteRemoteAudio	暂停/恢复播放远端的音频流
muteAllRemoteAudio	暂停/恢复播放所有远端用户的音频流
setAudioRoute	设置音频路由
setRemoteAudioVolume	设定某一个远端用户的声音播放音量
setAudioCaptureVolume	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量

版权所有：腾讯云计算（北京）有限责任公司

enableAudioVolumeEvaluation	启用音量大小提示
startAudioRecording	开始录音
stopAudioRecording	停止录音
startLocalRecording	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition	设置 3D 音效中远端用户坐标信息
set3DSpatialReceivingRange	设置指定用户所发出声音的可被接收范围

设备管理相关接口

函数列表	描述
getDeviceManager	获取设备管理类（TXDeviceManager）

美颜特效和图像水印

函数列表	描述
getBeautyManager	获取美颜管理类（TXBeautyManager）
setWatermark	添加水印

背景音乐和声音特效

函数列表	描述
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集

版权所有：腾讯云计算（北京）有限责任公司

stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）

屏幕分享相关接口

函数列表	描述
startScreenCapture	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
setSubStreamEncoderParam	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）

自定义采集和自定义渲染

函数列表	描述
enableCustomVideoCapture	启用/关闭视频自定义采集模式
sendCustomVideoData	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture	启用音频自定义采集模式
sendCustomAudioData	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame	启用/关闭自定义音轨
mixExternalAudioFrame	向 SDK 混入自定义音轨
setMixExternalAudioVolume	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳
setLocalVideoProcessListener	设置第三方美颜的视频数据回调
setLocalVideoRenderListener	设置本地视频自定义渲染回调
setRemoteVideoRenderListener	设置远端视频自定义渲染回调

版权所有：腾讯云计算（北京）有限责任公司

setAudioFrameListener	设置音频数据自定义回调
setCapturedAudioFrameCallbackFormat	设置本地麦克风采集出的音频帧回调格式
setLocalProcessedAudioFrameCallbackFormat	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameCallbackFormat	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering	开启音频自定义播放
getCustomAudioRenderingFrame	获取可播放的音频数据

自定义消息发送接口

函数列表	描述
sendCustomCmdMsg	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg	使用 SEI 通道发送自定义消息给房间内所有用户

网络测试接口

函数列表	描述
startSpeedTest	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速

调试相关接口

函数列表	描述
getSDKVersion	获取 SDK 版本信息
setLogLevel	设置 Log 输出级别
setConsoleEnabled	启用/禁用控制台日志打印
setLogCompressEnabled	启用/禁用日志的本地压缩
setLogDirPath	设置本地日志的保存路径

版权所有：腾讯云计算（北京）有限责任公司

setLogListener	设置日志回调
showDebugView	显示仪表盘
TRTCViewMargin	设置仪表盘的边距
callExperimentalAPI	调用实验性接口

加密接口

函数列表	描述
enablePayloadPrivateEncryption	开启或关闭媒体流私有加密

错误和警告事件

函数列表	描述
onError	错误事件回调
onWarning	警告事件回调

房间相关事件回调

函数列表	描述
onEnterRoom	进入房间成功与否的事件回调
onExitRoom	离开房间的事件回调
onSwitchRole	切换角色的事件回调
onSwitchRoom	切换房间的结果回调
onConnectOtherRoom	请求跨房通话的结果回调
onDisConnectOtherRoom	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode	更改跨房主播上行能力的结果回调

版权所有：腾讯云计算（北京）有限责任公司

用户相关事件回调

函数列表	描述
onRemoteUserEnterRoom	有用户加入当前房间
onRemoteUserLeaveRoom	有用户离开当前房间
onUserVideoAvailable	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable	某远端用户发布/取消了辅路视频画面
onUserAudioAvailable	某远端用户发布/取消了自己的音频
onFirstVideoFrame	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated	远端音频状态变化的事件回调
onUserVideoSizeChanged	用户视频大小发生改变回调

网络和技术指标统计回调

函数列表	描述
onNetworkQuality	网络质量的实时统计回调
onStatistics	音视频技术指标的实时统计回调
onSpeedTestResult	网速测试的结果回调

与云端连接情况的事件回调

函数列表	描述
onConnectionLost	SDK 与云端的连接已经断开

版权所有：腾讯云计算（北京）有限责任公司

onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复

硬件设备相关事件回调

函数列表	描述
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onAudioRouteChanged	当前音频路由发生变化（仅适用于移动设备）
onUserVoiceVolume	音量大小的反馈回调

自定义消息的接收事件回调

函数列表	描述
onRecvCustomCmdMsg	收到自定义消息的事件回调
onMissCustomCmdMsg	自定义消息丢失的事件回调
onRecvSEIMsg	收到 SEI 消息的回调

CDN 相关事件回调

函数列表	描述
onStartPublishing	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig	设置云端混流的排版布局和转码参数的事件回调

版权所有：腾讯云计算（北京）有限责任公司

onStartPublishMediaStream	开始发布媒体流的事件回调
onUpdatePublishMediaStream	更新媒体流的事件回调
onStopPublishMediaStream	停止媒体流的事件回调
onCdnStreamStateChanged	RTMP/RTMPS 推流状态发生改变回调

屏幕分享相关事件回调

函数列表	描述
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused	屏幕分享暂停的事件回调
onScreenCaptureResumed	屏幕分享恢复的事件回调
onScreenCaptureStopped	屏幕分享停止的事件回调

本地录制和本地截图的事件回调

函数列表	描述
onLocalRecordBegin	本地录制任务已经开始的事件回调
onLocalRecording	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment	本地录制分片的事件回调
onLocalRecordComplete	本地录制任务已经结束的事件回调
onSnapshotComplete	本地截图完成的事件回调

废弃的事件回调

函数列表	描述
onUserEnter	有主播加入当前房间（已废弃）
onUserExit	有主播离开当前房间（已废弃）

版权所有：腾讯云计算（北京）有限责任公司

onAudioEffectFinished	音效播放已结束（已废弃）
onSpeedTest	服务器测速的结果回调（已废弃）

视频数据自定义回调

函数列表	描述
onRenderVideoFrame	自定义视频渲染回调
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame	用于对接第三方美颜组件的视频处理回调
onGLContextDestory	SDK 内部 OpenGL 环境被销毁的通知

音频数据自定义回调

函数列表	描述
onCapturedAudioFrame	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onRemoteUserAudioFrame	混音前的每一路远程用户的音频数据
onMixedPlayAudioFrame	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）
onVoiceEarMonitorAudioFrame	耳返的音频数据

更多事件回调接口

函数列表	描述
onLog	本地 LOG 的打印回调

版权所有：腾讯云计算（北京）有限责任公司

背景音乐预加载事件回调

函数列表	描述
onLoadProgress	背景音乐预加载进度
onLoadError	背景音乐预加载出错

背景音乐的播放事件回调

函数列表	描述
onStart	背景音乐开始播放
onPlayProgress	背景音乐的播放进度
onComplete	背景音乐已经播放完毕

人声相关的特效接口

函数列表	描述
enableVoiceEarMonitor	开启耳返
setVoiceEarMonitorVolume	设置耳返音量
setVoiceReverbType	设置人声的混响效果
setVoiceChangerType	设置人声的变声特效
setVoiceCaptureVolume	设置语音音量
setVoicePitch	设置语音音调

背景音乐的相关接口

函数列表	描述
setMusicObserver	设置背景音乐的事件回调接口

版权所有：腾讯云计算（北京）有限责任公司

startPlayMusic	开始播放背景音乐
stopPlayMusic	停止播放背景音乐
pausePlayMusic	暂停播放背景音乐
resumePlayMusic	恢复播放背景音乐
setAllMusicVolume	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume	设置某一首背景音乐的本地音量的大小
setMusicPitch	调整背景音乐的音调高低
setMusicSpeedRate	调整背景音乐的变速效果
getMusicCurrentPosInMS	获取背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInMS	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate	调整搓碟的变速效果
setPreloadObserver	设置预加载事件回调
preloadMusic	预加载背景音乐
getMusicTrackCount	获取背景音乐的音轨数量
setMusicTrack	指定背景音乐的播放音轨

美颜相关接口

函数列表	描述
setBeautyStyle	设置美颜（磨皮）算法
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别
enableSharpnessEnhancement	开启清晰度增强
setRuddyLevel	设置红润级别

版权所有：腾讯云计算（北京）有限责任公司

setFilter	设置色彩滤镜效果
setFilterStrength	设置色彩滤镜的强度
setGreenScreenFile	设置绿幕背景视频
setEyeScaleLevel	设置大眼级别
setFaceSlimLevel	设置瘦脸级别
setFaceVLevel	设置 V 脸级别
setChinLevel	设置下巴拉伸或收缩
setFaceShortLevel	设置短脸级别
setFaceNarrowLevel	设置窄脸级别
setNoseSlimLevel	设置瘦鼻级别
setEyeLightenLevel	设置亮眼级别
setToothWhitenLevel	设置牙齿美白级别
setWrinkleRemoveLevel	设置祛皱级别
setPounchRemoveLevel	设置祛眼袋级别
setSmileLinesRemoveLevel	设置法令纹去除级别
setForeheadLevel	设置发际线调整级别
setEyeDistanceLevel	设置眼距
setEyeAngleLevel	设置眼角调整级别
setMouthShapeLevel	设置嘴型调整级别
setNoseWingLevel	设置鼻翼调整级别
setNosePositionLevel	设置鼻子位置
setLipsThicknessLevel	设置嘴唇厚度
setFaceBeautyLevel	设置脸型
setMotionTmpl	选择 AI 动效挂件
setMotionMute	是否在动效素材播放时静音

版权所有：腾讯云计算（北京）有限责任公司

设备操作接口

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera	切换前置或后置摄像头（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition	设置摄像头的对焦位置（仅适用于移动端）
enableCameraTorch	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）
setAudioRoute	设置音频路由（仅适用于移动端）
setExposureCompensation	设置摄像头的曝光参数，取值范围从-1到1
setCameraCapturerParam	设置摄像头采集偏好

弃用接口

函数列表	描述
setSystemVolumeType	设置系统音量类型（仅适用于移动端）

废弃接口

函数列表	描述
setListener	设置 TRTC 事件回调
setBeautyStyle	设置美颜、美白以及红润效果级别
setEyeScaleLevel	设置大眼级别
setFaceSlimLevel	设置瘦脸级别

版权所有：腾讯云计算（北京）有限责任公司

setFaceVLevel	设置 V 脸级别
setChinLevel	设置下巴拉伸或收缩幅度
setFaceShortLevel	设置短脸级别
setNoseSlimLevel	设置瘦鼻级别
selectMotionTmpl	设置动效贴纸
setMotionMute	设置动效静音
setFilter	设置色彩滤镜效果
setFilterConcentration	设置色彩滤镜浓度
setGreenScreenFile	设置绿幕背景视频
setReverbType	设置混响效果
setVoiceChangerType	设置变声类型
enableAudioEarMonitoring	开启（或关闭）耳返
enableAudioVolumeEvaluation	启用音量大小提示
switchCamera	切换摄像头
isCameraZoomSupported	查询当前摄像头是否支持缩放
setZoom	设置摄像头缩放倍数（焦距）
isCameraTorchSupported	查询是否支持开关闪光灯
enableTorch	开关/关闭闪光灯
isCameraFocusPositionInPreviewSupported	查询摄像头是否支持设置焦点
setFocusPosition	设置摄像头焦点坐标位置
isCameraAutoFocusFaceModeSupported	查询是否支持自动识别人脸位置
setSystemVolumeType	设置系统音量类型
checkAudioCapabilitySupport	查询是否支持音频某种能力（仅适用于Android）
startLocalAudio	设置音频质量
startRemoteView	开始显示远端视频画面

版权所有：腾讯云计算（北京）有限责任公司

stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode	设置本地图像的渲染模式
setLocalViewRotation	设置本地图像的顺时针旋转角度
setLocalViewMirror	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode	设置远端图像的渲染模式
setRemoteViewRotation	设置远端图像的顺时针旋转角度
startRemoteSubStreamView	开始显示远端用户的辅路画面
stopRemoteSubStreamView	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode	设置辅路画面的填充模式
setRemoteSubStreamViewRotation	设置辅路画面的顺时针旋转角度
setAudioQuality	设置音频质量
setPriorRemoteVideoStreamType	设定优先观看大画面还是小画面
setMicVolumeOnMixing	设置麦克风音量大小
playBGM	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration	获取背景音乐总时长（单位：毫秒）
setBGMPosition	设置背景音乐的播放进度
setBGMVolume	设置背景音乐的音量大小
setBGMPlayoutVolume	设置背景音乐的本地播放音量
setBGMPublishVolume	设置背景音乐的远端播放音量
playAudioEffect	播放音效
setAudioEffectVolume	设置音效音量
stopAudioEffect	停止播放音效

版权所有：腾讯云计算（北京）有限责任公司

stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume	设置所有音效音量
pauseAudioEffect	暂停音效
resumeAudioEffect	暂停音效
enableCustomVideoCapture	启用视频自定义采集模式
sendCustomVideoData	投送自己采集的视频数据
muteLocalVideo	暂停/恢复发布本地的视频流
muteRemoteVideoStream	暂停 / 恢复订阅远端用户的视频流
snapshotVideo	视频画面截图
startSpeedTest	开始进行网络测速（进入房间前使用）
startScreenCapture	启动屏幕分享
setVideoEncoderRotation	设置视频编码器输出的画面方向
setVideoEncoderMirror	设置编码器输出的画面镜像模式
setGSensorMode	设置重力感应的适配模式

版权所有：腾讯云计算（北京）有限责任公司

TRTCCloud

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTCCloud @ TXLiteAVSDK
Function: 腾讯云 TRTC 主功能接口
Version: 11.9

TRTCCloud

TRTCCloud

函数列表	描述
sharedInstance	创建 TRTCCloud 实例（单例模式）
destroySharedInstance	销毁 TRTCCloud 实例（单例模式）
addListener	添加 TRTC 事件回调
removeListener	移除 TRTC 事件回调
setListenerHandler	设置驱动 TRTCCloudListener 事件回调的队列
enterRoom	进入房间
exitRoom	离开房间
switchRole	切换角色
switchRole	切换角色（支持设置权限位）
switchRoom	切换房间
ConnectOtherRoom	请求跨房通话
DisconnectOtherRoom	退出跨房通话
setDefaultStreamRecvMode	设置订阅模式（需要在进入房前设置才能生效）
createSubCloud	创建子房间实例（用于多房间并发观看）

destroySubCloud	销毁子房间实例
updateOtherRoomForwardMode	更改跨房主播在本房间的上行能力
startPublishing	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig	设置云端混流的排版布局和转码参数
startPublishMediaStream	开始发布媒体流
updatePublishMediaStream	更新发布媒体流
stopPublishMediaStream	停止发布媒体流
startLocalPreview	开启本地摄像头的预览画面（移动端）
updateLocalView	更新本地摄像头的预览画面
stopLocalPreview	停止摄像头预览
muteLocalVideo	暂停/恢复发布本地的视频流
setVideoMuteImage	设置本地画面被暂停期间的替代图片
startRemoteView	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView	更新远端用户的视频渲染控件
stopRemoteView	停止订阅远端用户的视频流，并释放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源
muteRemoteVideoStream	暂停/恢复订阅远端用户的视频流
muteAllRemoteVideoStreams	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam	设置视频编码器的编码参数
setNetworkQosParam	设置网络质量控制的相关参数
setLocalRenderParams	设置本地画面的渲染参数
setRemoteRenderParams	设置远端画面的渲染模式

版权所有：腾讯云计算（北京）有限责任公司

enableEncSmallVideoStream	开启大小画面双路编码模式
setRemoteVideoStreamType	切换指定远端用户的大小画面
snapshotVideo	视频画面截图
setPerspectiveCorrectionPoints	视频画面透视校正坐标设置
setGravitySensorAdaptiveMode	设置重力感应的适配模式（11.7 及以上版本）
startLocalAudio	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布
muteLocalAudio	暂停/恢复发布本地的音频流
muteRemoteAudio	暂停/恢复播放远端的音频流
muteAllRemoteAudio	暂停/恢复播放所有远端用户的音频流
setAudioRoute	设置音频路由
setRemoteAudioVolume	设定某一个远端用户的声音播放音量
setAudioCaptureVolume	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
enableAudioVolumeEvaluation	启用音量大小提示
startAudioRecording	开始录音
stopAudioRecording	停止录音
startLocalRecording	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition	设置 3D 音效中远端用户坐标信息

版权所有：腾讯云计算（北京）有限责任公司

set3DSpatialReceivingRange	设置指定用户所发出声音的可被接收范围
getDeviceManager	获取设备管理类（TXDeviceManager）
getBeautyManager	获取美颜管理类（TXBeautyManager）
setWatermark	添加水印
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集
stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）
startScreenCapture	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
setSubStreamEncoderParam	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）
enableCustomVideoCapture	启用/关闭视频自定义采集模式
sendCustomVideoData	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture	启用音频自定义采集模式
sendCustomAudioData	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame	启用/关闭自定义音轨
mixExternalAudioFrame	向 SDK 混入自定义音轨
setMixExternalAudioVolume	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳
setLocalVideoProcessListener	设置第三方美颜的视频数据回调
setLocalVideoRenderListener	设置本地视频自定义渲染回调
setRemoteVideoRenderListener	设置远端视频自定义渲染回调
setAudioFrameListener	设置音频数据自定义回调
setCapturedAudioFrameCallbackFormat	设置本地麦克风采集出的音频帧回调格式

版权所有：腾讯云计算（北京）有限责任公司

setLocalProcessedAudioFrameCallbackFormat	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameCallbackFormat	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering	开启音频自定义播放
getCustomAudioRenderingFrame	获取可播放的音频数据
sendCustomCmdMsg	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg	使用 SEI 通道发送自定义消息给房间内所有用户
startSpeedTest	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速
getSDKVersion	获取 SDK 版本信息
setLogLevel	设置 Log 输出级别
setConsoleEnabled	启用/禁用控制台日志打印
setLogCompressEnabled	启用/禁用日志的本地压缩
setLogDirPath	设置本地日志的保存路径
setLogListener	设置日志回调
showDebugView	显示仪表盘
TRTCViewMargin	设置仪表盘的边距
callExperimentalAPI	调用实验性接口
enablePayloadPrivateEncryption	开启或关闭媒体流私有加密

sharedInstance

sharedInstance

TRTCCloud sharedInstance	(Context context)

创建 TRTCCloud 实例（单例模式）

参数	描述

context	仅适用于 Android 平台，SDK 内部会将其转化为 Android 平台的 ApplicationContext 用于调用 Android System API。 如果传入的 context 参数为空，SDK 内部会自动获取当前进程的 ApplicationContext。

注意

1. 如果您使用 delete ITRTCCloud* 会导致编译错误，请使用 destroyTRTCCloud 释放对象指针。
2. 在 Windows、Mac 和 iOS 平台上，请调用 getTRTCShareInstance() 接口。
3. 在 Android 平台上，请调用 getTRTCShareInstance(void *context) 接口。

destroySharedInstance

destroySharedInstance

销毁 TRTCCloud 实例（单例模式）

addListener

addListener

void addListener	(TRTCCloudListener listener)

添加 TRTC 事件回调

您可以通过 TRTCCloudListener 获得来自 SDK 的各类事件通知（比如：错误码，警告码，音视频状态参数等）。

removeListener

removeListener

void removeListener	(TRTCCloudListener listener)

移除 TRTC 事件回调

setListenerHandler

setListenerHandler

void setListenerHandler	(Handler listenerHandler)

设置驱动 TRTCCloudListener 事件回调的队列

如果您不指定自己的 listenerHandler 默认会采用 MainQueue 作为驱动 TRTCCloudListener 事件回调的队列。
也就是当您不设置 listenerHandler 属性时，TRTCCloudListener 中的所有回调函数都是由 MainQueue 来驱动的。

参数	描述
listenerHandler	事件回调句柄

注意

如果您指定了自己的 listenerHandler TRTCCloudListener 回调函数中操作 UI，否则会引发线程安全问题。

enterRoom

enterRoom

void enterRoom	(TRTCCloudDef.TRTCParams param
	int scene)

进入房间

TRTC 的所有用户都需要进入房间才能“发布”或“订阅”音视频流，“发布”是指将自己的音频和视频推送到云端，“订阅”是指从云端拉取房间里其他用户的音视频流。
调用该接口时，您需要指定您的应用场景 TRTCAppScene 以获取最佳的音视频传输体验，这些场景可以分成两大类：

实时通话：

包括 TRTC_APP_SCENE_VIDEOCALL 和 TRTC_APP_SCENE_AUDIOCALL 两个可选项，分别是视频通话和语音通话，该模式适合 1对1 的音视频通话，或者参会人数在 300 人以内的在线会议。

在线直播：

包括 TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM 两个可选项，分别是视频直播和语音直播，该模式适合十万人以内的在线直播场景，但需要您在接下来介绍的 TRTCParams 参数中指定 **角色(role)** 这个字段，也就是将房间中的用户区分为 **主播** (TRTCRoleAnchor) 和 **观众** (TRTCRoleAudience) 两种不同的角色。

调用该接口后，您会收到来自 TRTCCloudListener 中的 onEnterRoom(result) 回调：
如果进房成功，参数 result 会是一个正数（result > 0），表示从函数调用到进入房间所花费的时间，单位是毫秒（ms）。
如果进房失败，参数 result 会是一个负数（result < 0），表示进房失败的**错误码**。

参数	描述
param	进房参数，用于指定用户的身份、角色和安全票据等信息，详情请参见 TRTCParams 。
scene	应用场景，用于指定您的业务场景，同一个房间内的所有用户需要设定相同的 TRTCAppScene。

注意

1. 同一个房间内的所有用户需要设定相同的 scene。不同的 scene 会导致偶现的异常问题。
2. 当您指定参数 scene 为 TRTC_APP_SCENE_LIVE 或 TRTC_APP_SCENE_VOICE_CHATROOM 时，您必须通过 TRTCParams 中的 “role” 字段为当前用户设定他/她在房间中的角色。
3. 请您尽量保证 enterRoom 与 exitRoom 前后配对使用，即保证”先退出前一个房间再进入下一个房间”，否则会导致很多异常问题。

exitRoom

exitRoom

离开房间

调用该接口会让用户离开自己所在的音视频房间，并释放摄像头、麦克风、扬声器等设备资源。
等资源释放完毕之后，SDK 会通过 TRTCCloudListener 中的 onExitRoom 回调向您通知。

如果您要再次调用 enterRoom 或者切换到其他的供应商的 SDK，建议等待 onExitRoom 回调到来之后再执行之后的操作，以避免摄像头或麦克风被占用的问题。

switchRole

switchRole

void switchRole	(int role)

切换角色

调用本接口可以实现用户在 `主播` 和 `观众` 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了 **只有主播才能发布自己的音视频** 的规则。因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成 **主播**。

您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

版权所有：腾讯云计算（北京）有限责任公司

参数	描述
role	角色，默认为 **主播**。 TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。 TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成 **主播**，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTC_APP_SCENE_LIVE）和语音聊天室（TRTC_APP_SCENE_VOICE_CHATROOM）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTC_APP_SCENE_VIDEOCALL 或 TRTC_APP_SCENE_AUDIOCALL，请不要调用这个接口。

switchRole

switchRole

void switchRole	(int role
	final String privateMapKey)

切换角色（支持设置权限位）

调用本接口可以实现用户在 主播 和 观众 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了“只有主播才能发布自己的音视频”的规则。
因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成“主播”。
您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

参数	描述
privateMapKey	用于权限控制的权限票据，当您希望某个房间只能让特定的 userId 进入或者上行视频时，需要使用 privateMapKey 进行权限保护。 仅建议有高级别安全需求的客户使用，更多详情请参见 开启高级权限控制。
role	角色，默认为“主播”： TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。

	TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成“主播”，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTC_APP_SCENE_LIVE）和语音聊天室（TRTC_APP_SCENE_VOICE_CHATROOM）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTC_APP_SCENE_VIDEOCALL 或 TRTC_APP_SCENE_AUDIOCALL，请不要调用这个接口。

switchRoom

switchRoom

void switchRoom	(final TRTCCloudDef.TRTCSwitchRoomConfig config)

切换房间

使用该接口可以让用户可以快速从一个房间切换到另一个房间。
如果用户的身份是“观众”，该接口的调用效果等同于 exitRoom(当前房间) + enterRoom（新的房间）。
如果用户的身份是“主播”，该接口在切换房间的同时还会保持自己的音视频发布状态，因此在房间切换过程中，摄像头的预览和声音的采集都不会中断。

该接口适用于在线教育场景中，监课老师在多个房间中进行快速切换的场景。在该场景下使用 switchRoom 可以获得比 exitRoom+enterRoom 更好的流畅性和更少的代码量。
接口调用结果会通过 TRTCCloudListener 中的 onSwitchRoom(errCode, errMsg) 回调。

参数	描述
config	房间参数，详情请参见 TRTCSwitchRoomConfig。

注意

由于对老版本 SDK 兼容的需求，参数 config 中同时包含 roomId 与 strRoomId 两个参数，这两个参数的填写格外讲究，请注意如下事项：

1. 若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，将优先选用 roomId。
2. 所有房间需要同时使用 strRoomId 或同时使用 roomId，不可混用，否则将会出现很多预期之外的 bug。

ConnectOtherRoom

ConnectOtherRoom

void ConnectOtherRoom	(String param)

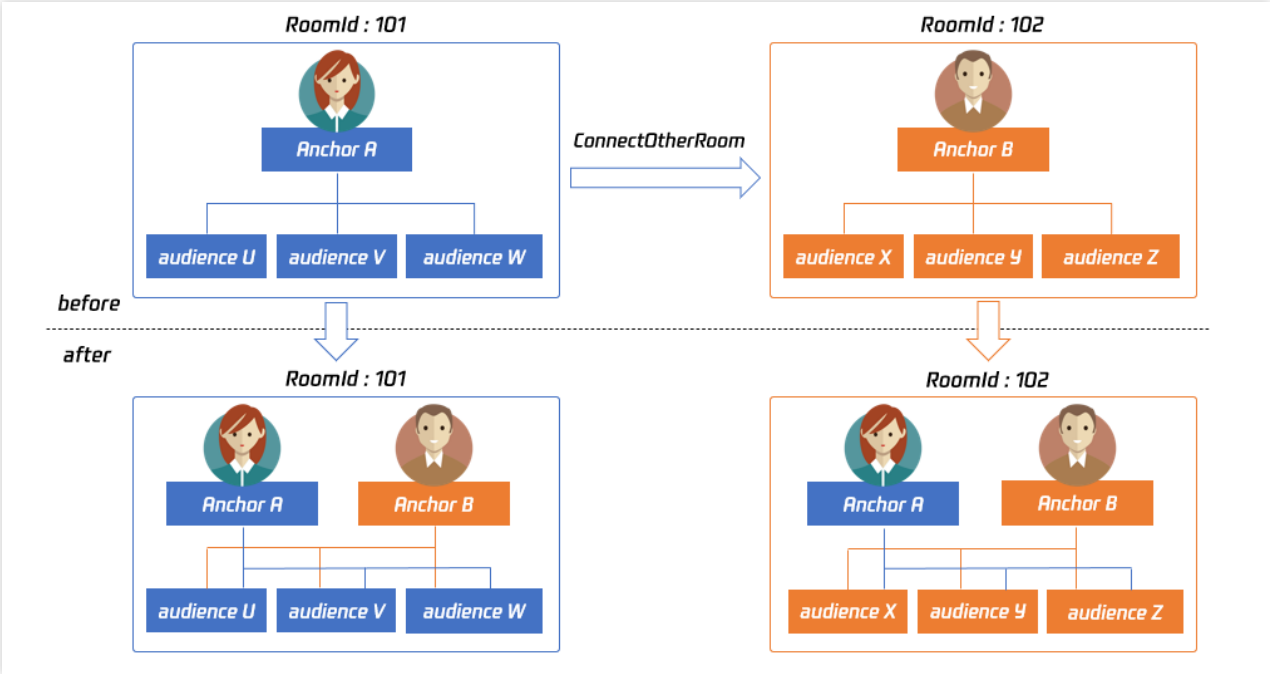
请求跨房通话

默认情况下，只有同一个房间中的用户之间可以进行音视频通话，不同的房间之间的音视频流是相互隔离的。
但您可以通过调用该接口，将另一个房间中某个主播音视频流发布到自己所在的房间中，与此同时，该接口也会将自己的音视频流发布到目标主播的房间中。

也就是说，您可以使用该接口让身处两个不同房间中的主播进行跨房间的音视频流分享，从而让每个房间中的观众都能观看到这两个主播的音视频。该功能可以用来实现主播之间的 PK 功能。

跨房通话的请求结果会通过 TRTCCloudListener 中的 onConnectOtherRoom 回调通知给您。

例如：当房间“101”中的主播 A 通过 connectOtherRoom() 跟房间“102”中的主播 B 建立跨房通话后，
房间“101”中的用户都会收到主播 B 的 onRemoteUserEnterRoom(B) 和 onUserVideoAvailable(B,true) 这两个事件回调，即房间“101”中的用户都可以订阅主播 B 的音视频。
房间“102”中的用户都会收到主播 A 的 onRemoteUserEnterRoom(A) 和 onUserVideoAvailable(A,true) 这两个事件回调，即房间“102”中的用户都可以订阅主播 A 的音视频。

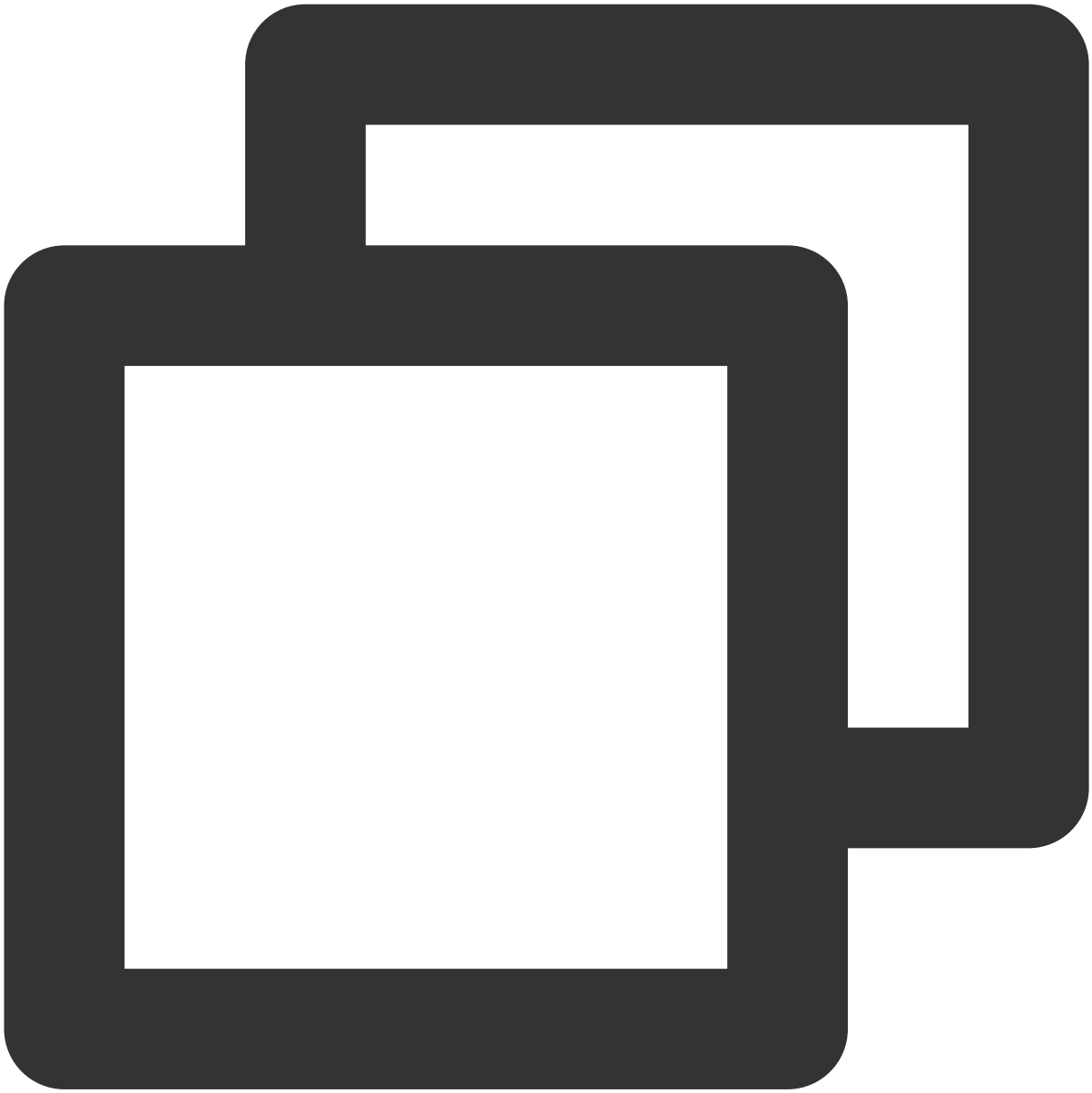


跨房通话的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数：

情况一：数字房间号

如果房间“101”中的主播 A 要跟房间“102”中的主播 B 连麦，那么主播 A 调用该接口时需要传入：{"roomId": 102, "userId": "userB"}

示例代码如下：

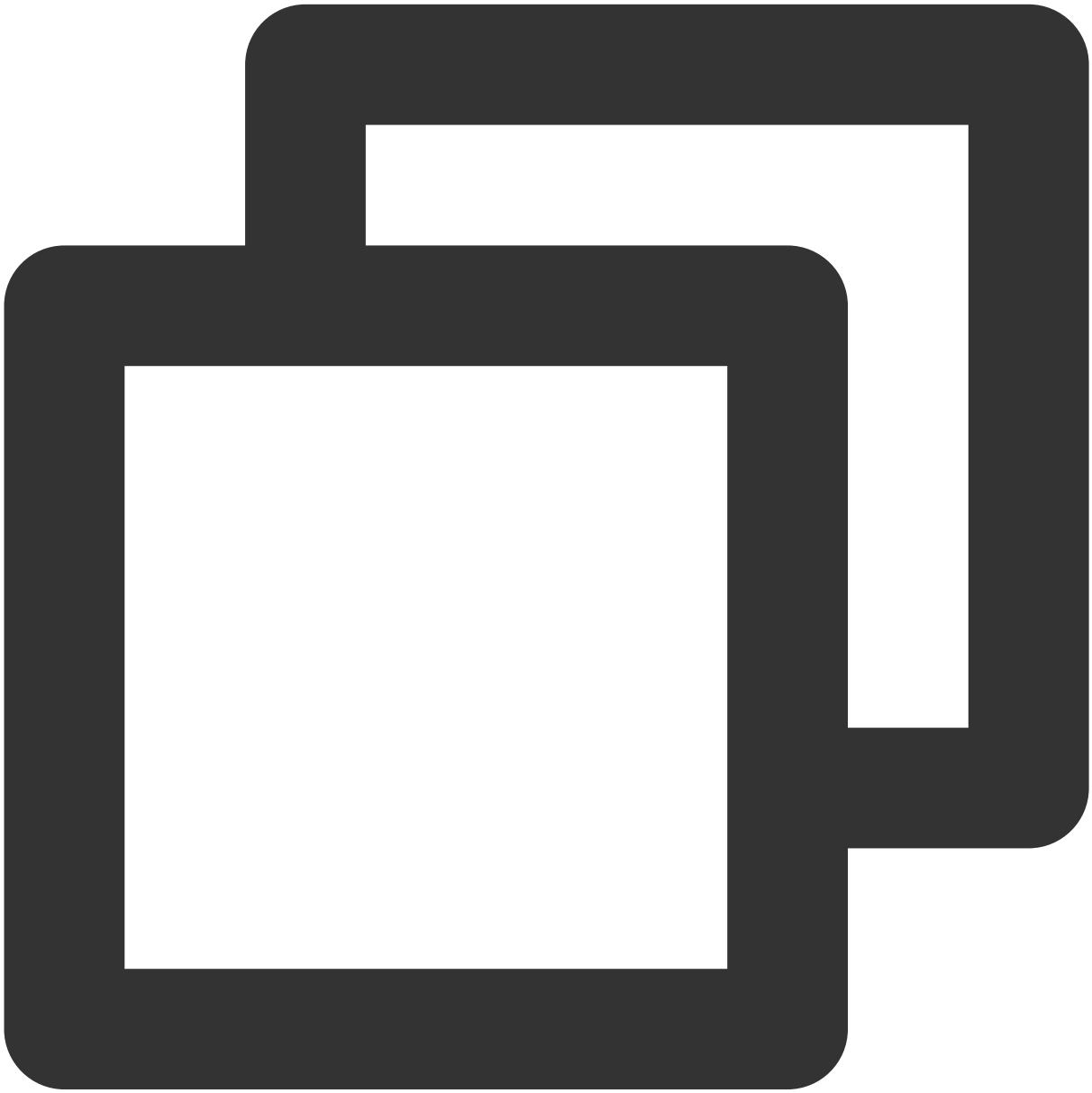


```
JSONObject jsonObj = new JSONObject();
jsonObj.put("roomId", 102);
jsonObj.put("userId", "userB");
trtc.ConnectOtherRoom(jsonObj.toString());
```

情况二：字符串房间号

如果您使用的是字符串房间号，务必请将 json 中的 “roomId” 替换成 “strRoomId”: {"strRoomId": "102", "userId": "userB"}

示例代码如下：



```
JSONObject jsonObj = new JSONObject();
jsonObj.put("strRoomId", "102");
jsonObj.put("userId", "userB");
trtc.ConnectOtherRoom(jsonObj.toString());
```

参数	描述
param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID。

DisconnectOtherRoom

DisconnectOtherRoom

退出跨房通话

退出结果会通过 TRTCCloudListener 中的 onDisconnectOtherRoom 回调通知给您。

setDefaultStreamRecvMode

setDefaultStreamRecvMode

void setDefaultStreamRecvMode	(boolean autoRecvAudio
	boolean autoRecvVideo)

设置订阅模式（需要在进入房前设置才能生效）

您可以通过该接口在“自动订阅”和“手动订阅”两种模式下进行切换：
自动订阅：默认模式，用户在进入房间后会立刻接收到该房间中的音视频流，音频会自动播放，视频会自动开始解码（依然需要您通过 startRemoteView 接口绑定渲染控件）。
手动订阅：在用户进入房间后，需要手动调用 startRemoteView 接口才能启动视频流的订阅和解码，需要手动调用 muteRemoteAudio (false) 接口才能启动声音的播放。

在绝大多数场景下，用户进入房间后都会订阅房间中所有主播的音视频流，因此 TRTC 默认采用了自动订阅模式，以求得最佳的“秒开体验”。
如果您的应用场景中每个房间同时会有很多路音视频流在发布，而每个用户只想选择性地订阅其中的 1-2 路，则推荐使用“手动订阅”模式以节省流量费用。

参数	描述
autoRecvAudio	true：自动订阅音频；false：需手动调用 muteRemoteAudio(false) 订阅音频。默认值：true。
autoRecvVideo	true：自动订阅视频；false：需手动调用 startRemoteView 订阅视频。默认值：true。

注意

1. 需要在进入房间前调用该接口进行设置才能生效。
2. 在自动订阅模式下，如果用户在进入房间后没有调用 startRemoteView 订阅视频流，SDK 会自动停止订阅视频流，以便达到节省流量的目的。

createSubCloud

createSubCloud

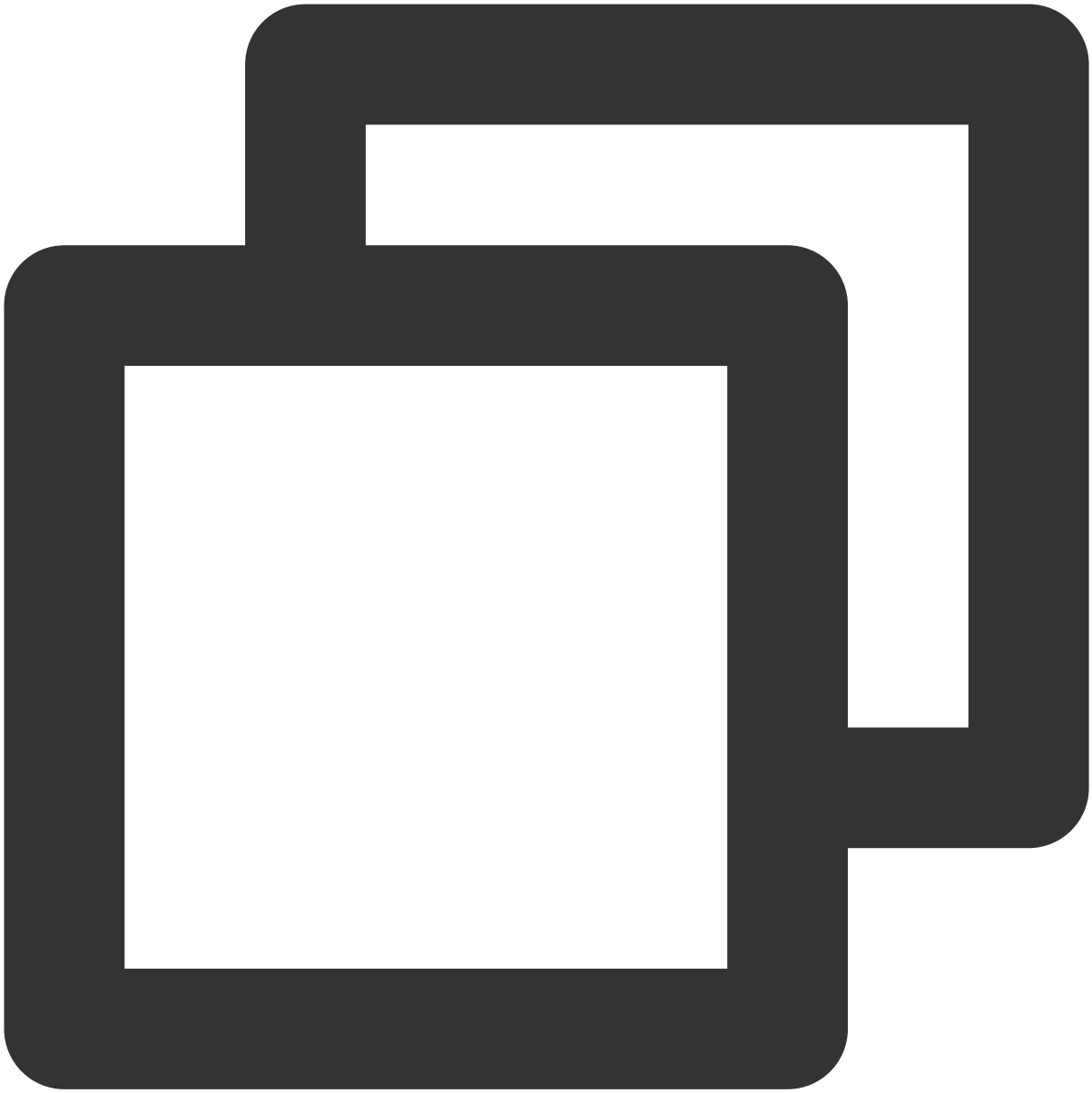
创建子房间实例（用于多房间并发观看）

TRTCCloud 一开始被设计成单例模式，限制了多房间并发观看的能力。
通过调用该接口，您可以创建出多个 TRTCCloud 实例，以便同时进入多个不同的房间观看音视频流。

但需要注意的是，您在多个 TRTCCloud 实例中发布音视频流的能力会受到一定限制。

该功能主要用于在线教育场景中一种被称为“超级小班课”的业务场景中，用于解决“每个 TRTC 的房间中最多只能有 50 人同时发布自己音视频流”的限制。

示例代码如下：



```
//In the small room that needs interaction, enter the room as an anchor and pus
TRTCCloud mainCloud = TRTCCloud.sharedInstance(mContext);
TRTCCloudDef.TRTCParams mainParams = new TRTCCloudDef.TRTCParams();
//Fill your params
mainParams.role = TRTCCloudDef.TRTCRoleAnchor;
mainCloud.enterRoom(mainParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
//...
mainCloud.startLocalPreview(true, videoView);
mainCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_DEFAULT);

//In the large room that only needs to watch, enter the room as an audience and
```

```
TRTCCloud subCloud = mainCloud.createSubCloud();
TRTCCloudDef.TRTCParams subParams = new TRTCCloudDef.TRTCParams();
//Fill your params
subParams.role = TRTCCloudDef.TRTCRoleAudience;
subCloud.enterRoom(subParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
//...
subCloud.startRemoteView(userId, TRTCCloudDef.TRTC_VIDEO_STREAM_TYPE_BIG, view)
//...
//Exit from new room and release it.
subCloud.exitRoom();
mainCloud.destroySubCloud(subCloud);
```

注意

同一个用户，可以使用同一个 userId 进入多个不同 roomId 的房间。

两台不同的终端设备不可以同时使用同一个 userId 进入同一个 roomId 的房间。

您可以分别为不同实例分别设置 TRTCCloudListener 获取各自的事件通知。

同一个用户可以在多个 TRTCCloud 实例中推流，也可以调用子实例中与本地音视频相关的接口。但需要注意：

多实例的音频需要同时为麦克风采集或自定义采集，而且与音频设备相关的接口调用会以最后一次为准；

与摄像头相关的调用会以最后一次为准：startLocalPreview。

返回值说明：

子 TRTCCloud 实例

destroySubCloud

destroySubCloud

void destroySubCloud	(final TRTCCloud subCloud)

销毁子房间实例

参数	描述
subCloud	子房间实例

updateOtherRoomForwardMode

updateOtherRoomForwardMode

void updateOtherRoomForwardMode	(String param)

更改跨房主播在本房间的上行能力

通常情况下，在调用 connectOtherRoom 接口与另一个房间的主播进行跨房通话后，本房间内的所有观众都将收到该主播发布的音视频流。
您可以通过调用该接口，限制跨房主播在本房间内的上行能力，禁止或允许跨房主播发布音频/主路视频/辅路视频，该行为会影响房间内的所有用户。
在禁用跨房主播某种上行能力后，本房间内所有用户将无法收到对应音视频流，且无法再订阅对应的音视频。

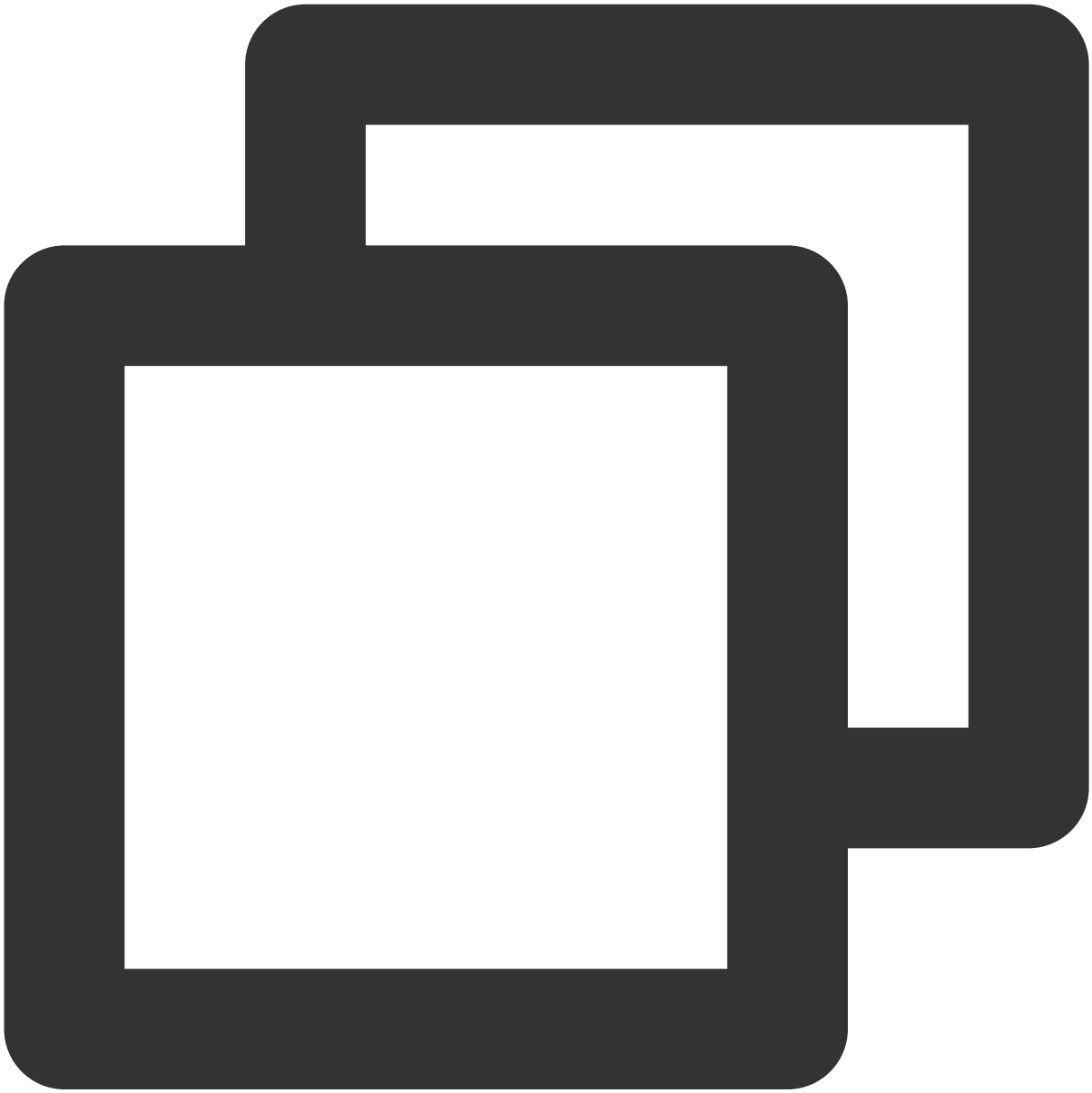
需要注意的是，该接口只能由进行跨房通话的主播调用，且通过该接口设置的限制会因为跨房通话的中断或对应主播退房重置。

该接口的调用结果会通过 TRTCCloudListener 中的 onUpdateOtherRoomForwardMode 回调通知给您。

例如：
房间“101”中有主播 A 与观众 B，房间“102”中有主播 C，主播 C 正常发布音视频流。主播 A 通过 connectOtherRoom() 跟主播 C 建立跨房通话。
此时，主播 A 与观众 B 都会收到主播 C 的 onRemoteUserEnterRoom(C) 和 onUserVideoAvailable(C,true)、onUserAudioAvailable(C,true) 这三个事件回调，可以订阅主播 C 的音视频。

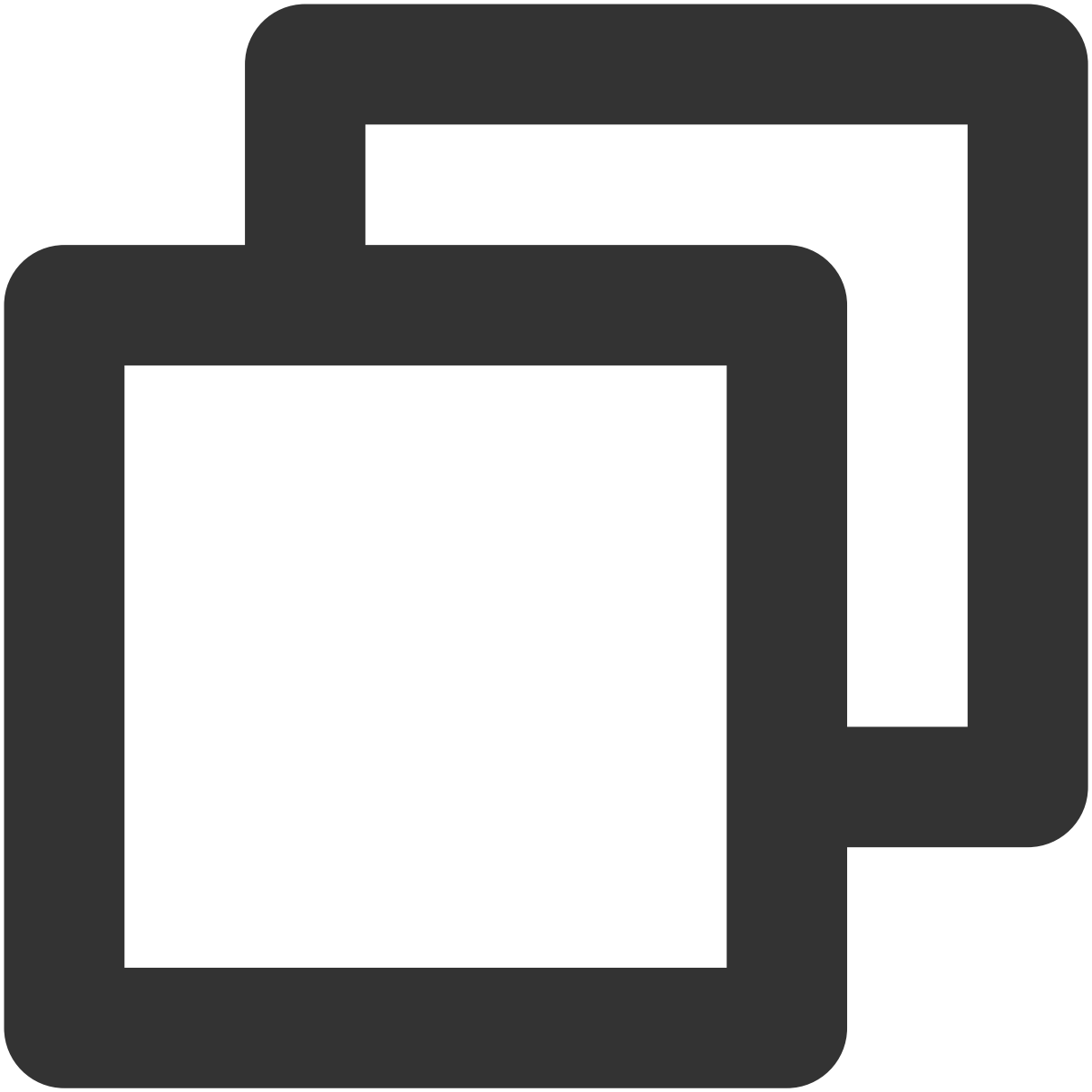
之后，主播 A 调用该接口禁用主播 C 在本房间发布音频的能力。
此后，主播 C 的音频流将无法发布至房间“101”，主播 A 与观众 B 都将收到 onUserAudioAvailable(C,false) 事件回调，且无法再通过 muteRemoteAudio(C,false) 订阅主播 C 的音频。
主播 C 的视频流将不受影响。房间“102”中的其他观众也不受影响，可以正常订阅主播 C 的音频。

该接口的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数，示例如下：
情况一：数字房间号



```
{
  "roomId":102,
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

情况二：字符串房间号



```
{
  "strRoomId":"102",
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

参数	描述

param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID，muteAudio、muteVideo、muteSubStream均为可选项，分表代表禁止或允许跨房主播发布音频/主路视频/辅路视频的能力。

startPublishing

startPublishing

void startPublishing	(final String streamId
	final int streamType)

开始向腾讯云直播 CDN 上发布音视频流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流旁路到直播 CDN 上。您可以通过参数 streamId 设定直播流的 StreamId，从而可以指定该用户的音视频流对应在直播 CDN 上的播放地址。

例如：您可以通过该接口将当前用户的直播流 ID 指定为 user_stream_001，那么该用户音视频流对应的 CDN 播放地址为：
“http://yourdomain/live/user_stream_001.flv”，其中 yourdomain 为您自己备案的播放域名，

您可以在直播控制台 配置您的播放域名，腾讯云不提供默认的播放域名。

您也可以在设置 enterRoom 的参数 TRTCParams 时指定 streamId, 而且我们更推荐您采用这种方案。

参数	描述
streamId	自定义流 ID。
streamType	仅支持 TRTC_VIDEO_STREAM_TYPE_BIG 和 TRTC_VIDEO_STREAM_TYPE_SUB。

注意

您需要提前在 实时音视频控制台 中的功能配置页面上开启“启用旁路推流”才能生效。
若您选择“指定流旁路”，则您可以通过该接口将对应音视频流推送到腾讯云 CDN 且指定为填写的流 ID。
若您选择“全局自动旁路”，则您可以通过该接口调整默认的流 ID。

stopPublishing

stopPublishing

停止向腾讯云直播 CDN 上发布音视频流

startPublishCDNStream

startPublishCDNStream

void startPublishCDNStream	(TRTCCloudDef.TRTCPublishCDNParam param)

开始向非腾讯云 CDN 上发布音视频流

该接口跟 startPublishing 功能类似，不同之处在于，startPublishing 仅支持向腾讯云的 CDN 发布，而本接口支持向非腾讯云的直播 CDN 上转推音视频流。

参数	描述
param	CDN 转推参数，详情请参见 TRTCPublishCDNParam

注意

使用 startPublishing 接口向腾讯云的直播 CDN 上发布音视频流不会收取额外费用。
使用 startPublishCDNStream 接口向非腾讯云的直播 CDN 上发布音视频流，需要收取额外的转推带宽费用。

stopPublishCDNStream

stopPublishCDNStream

停止向非腾讯云 CDN 上发布音视频流

setMixTranscodingConfig

setMixTranscodingConfig

void setMixTranscodingConfig	(TRTCCloudDef.TRTCTranscodingConfig config)

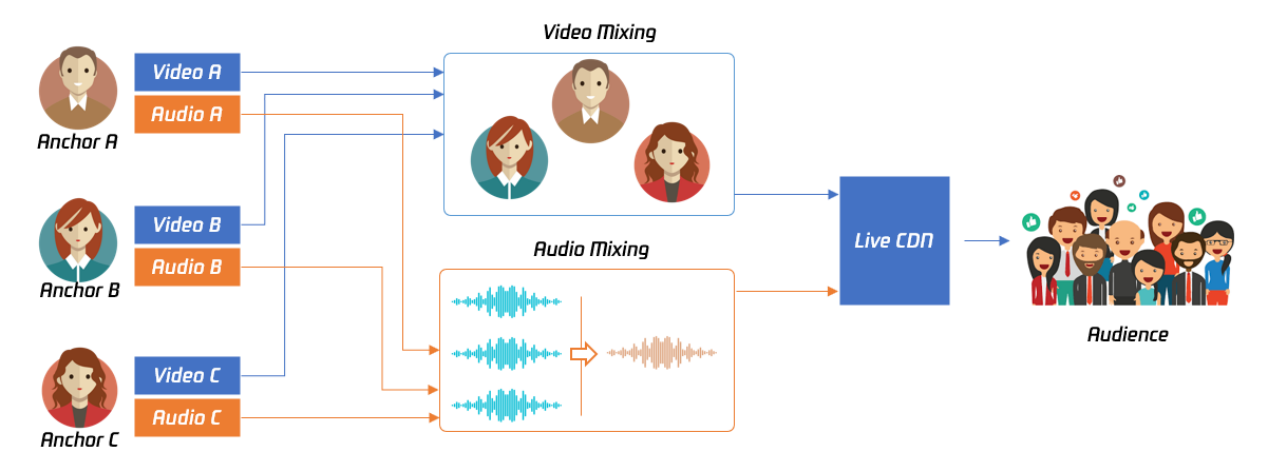
设置云端混流的排版布局和转码参数

在一个直播间中可能同时会有多个主播发布自己的音视频流，但对于直播 CDN 上的观众而言，只需要观看一条 HTTP-FLV 或 HLS 格式的视频流即可。
当您调用本接口函数时，SDK 会向腾讯云的 TRTC 混流服务器发送一条指令，混流服务器会将房间里的多路音视频流混合成一路。

版权所有：腾讯云计算（北京）有限责任公司

您可以通过 TRTCTranscodingConfig 参数来调整每一路画面的排版布局，也可以设置混合后的音视频流的各项编码参数。

参考文档：云端混流转码。



参数	描述
config	如果 config 不为空，则开启云端混流，如果 config 为空则停止云端混流。详情请参见 TRTCTranscodingConfig 。

注意

关于云端混流的注意事项：

混流转码为收费功能，调用接口将产生云端混流转码费用，详见 云端混流转码计费说明 。

调用该接口的用户，如果没设定 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到当前用户所对应的音视频流上，即 A + B => A。

调用该接口的用户，如果设定了 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到您指定的 streamId 上，即 A + B => streamId。

请注意，若您还在房间中且不再需要混流，请务必再次调用本接口并将 config 设置为空以进行取消，不及时取消混流可能会引起不必要的计费损失。

请放心，当您退房时 TRTC 会自动取消混流状态。

startPublishMediaStream

startPublishMediaStream

void startPublishMediaStream	(TRTCCloudDef.TRTCPublishTarget target

	TRTCCloudDef.TRTCStreamEncoderParam params
	TRTCCloudDef.TRTCStreamMixingConfig config)

开始发布媒体流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流转推/转码到直播 CDN 或者回推到 TRTC 房间中
您可以通过 TRTCPublishTarget 配置中的 TRTCPublishMode 指定具体的发布模式

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。

注意

1. SDK 会通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）。
2. 同一个任务（TRTCPublishMode 与 TRTCPublishCdnUrl 均相同）仅支持启动一次。若您后续需要更新或者停止该项任务，需要记录并使用返回的 taskId，通过 updatePublishMediaStream 或者 stopPublishMediaStream 来操作。
3. target 支持同时配置多个 CDN URL（最多同时 10 个）。若您的同一个转推/转码任务需要发布至多路 CDN，则仅需要在 target 中配置多个 CDN URL 即可。同一个转码任务即使有多个转推地址，对应的转码计费仍只收取一份。
4. 使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。一种推荐的方案是 URL 中使用 “sdkappid_roomid_userid_main” 作为区分标识，这种命名方式容易辨认且不会在您的多个应用中发生冲突。

updatePublishMediaStream

updatePublishMediaStream

void updatePublishMediaStream	(final String taskId
	TRTCCloudDef.TRTCPublishTarget target
	TRTCCloudDef.TRTCStreamEncoderParam params
	TRTCCloudDef.TRTCStreamMixingConfig config)

更新发布媒体流

该接口会向 TRTC 服务器发送指令，更新通过 startPublishMediaStream 启动的媒体流

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 您可以通过本接口来更新发布的 CDN URL（支持增删，最多同时 10 个），但您使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。
2. 您可以通过 taskId 来更新调整转推/转码任务。例如在 pk 业务中，您可以先通过 startPublishMediaStream 发起转推，接着在主播发起 pk 时，通过 taskId 和本接口将转推更新为转码任务。此时，CDN 播放将连续并且不会发生断流（您需要保持媒体流编码输出参数 param 一致）。
3. 同一个任务不支持纯音频、音视频、纯视频之间的切换。

stopPublishMediaStream

stopPublishMediaStream

void stopPublishMediaStream	(final String taskId)

停止发布媒体流

该接口会向 TRTC 服务器发送指令，停止通过 startPublishMediaStream 启动的媒体流

参数	描述
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 若您的业务后台并没有保存该 taskId，在您的主播异常退房重进后，如果您需要重新获取 taskId，您可以再次调用 startPublishMediaStream 启动任务。此时 TRTC 后台会返回任务启动失败，同时带给您上一次启动的 taskId

2. 若 taskId 填空字符串，将会停止该用户所有通过 startPublishMediaStream 启动的媒体流，如果您只启动了一个媒体流或者想停止所有通过您启动的媒体流，推荐使用这种方式。

startLocalPreview

startLocalPreview

void startLocalPreview	(boolean frontCamera
	TXCloudVideoView view)

开启本地摄像头的预览画面（移动端）

在 enterRoom 之前调用此函数，SDK 只会开启摄像头，并一直等到您调用 enterRoom 之后才开始推流。
在 enterRoom 之后调用此函数，SDK 会开启摄像头并自动开始视频推流。当开始渲染首帧摄像头画面时，您会收到 TRTCCloudListener 中的 onCameraDidReady 回调通知。

参数	描述
frontCamera	true：前置摄像头；false：后置摄像头。
view	承载视频画面的控件。

注意

如果希望开播前预览摄像头画面并通过 BeautyManager 调节美颜参数，您可以：
方案一：在调用 enterRoom 之前调用 startLocalPreview。
方案二：在调用 enterRoom 之后调用 startLocalPreview + muteLocalVideo(true)。

updateLocalView

updateLocalView

void updateLocalView	(TXCloudVideoView view)

更新本地摄像头的预览画面

stopLocalPreview

stopLocalPreview

停止摄像头预览

muteLocalVideo

muteLocalVideo

void muteLocalVideo	(int streamType
	boolean mute)

暂停/恢复发布本地的视频流

该接口可以暂停（或恢复）发布本地的视频画面，暂停之后，同一房间中的其他用户将无法继续看到自己画面。
该接口在指定 TRTC_VIDEO_STREAM_TYPE_BIG 时等效于 start/stopLocalPreview 这两个接口，但具有更好的响应速度。
因为 start/stopLocalPreview 需要打开和关闭摄像头，而打开和关闭摄像头都是硬件设备相关的操作，非常耗时。
相比之下，muteLocalVideo 只需要在软件层面对数据流进行暂停或者放行即可，因此效率更高，也更适合需要频繁打开关闭的场景。

当暂停/恢复发布指定 TRTC_VIDEO_STREAM_TYPE_BIG 后，同一房间中的其他用户将会收到 onUserVideoAvailable 回调通知。
当暂停/恢复发布指定 TRTC_VIDEO_STREAM_TYPE_SUB 后，同一房间中的其他用户将会收到 onUserSubStreamAvailable 回调通知。

参数	描述
mute	true：暂停；false：恢复。
streamType	要暂停/恢复的视频流类型（仅支持 TRTC_VIDEO_STREAM_TYPE_BIG 和 TRTC_VIDEO_STREAM_TYPE_SUB）。

setVideoMuteImage

setVideoMuteImage

void setVideoMuteImage	(Bitmap image
	int fps)

设置本地画面被暂停期间的替代图片

当您调用 muteLocalVideo(true) 暂停本地画面时，您可以通过调用本接口设置一张替代图片，设置后，房间中的其他用户会看到这张替代图片，而不是黑屏画面。

参数	描述
fps	设置替代图片帧率，最小值为5，最大值为10，默认5。
image	设置替代图片，空值代表在 muteLocalVideo 之后不再发送视频流数据，默认值为空。

startRemoteView

startRemoteView

void startRemoteView	(String userId
	int streamType
	TXCloudVideoView view)

订阅远端用户的视频流，并绑定视频渲染控件

调用该接口可以让 SDK 拉取指定 userid 的视频流，并渲染到参数 view 指定的渲染控件上。您可以通过 setRemoteRenderParams 设置画面的显示模式。
如果您已经知道房间中有视频流的用户的 userid，可以直接调用 startRemoteView 订阅该用户的画面。
如果您不知道房间中有哪些用户在发布视频，您可以在 enterRoom 之后等待来自 onUserVideoAvailable 的通知。

调用本接口只是启动视频流的拉取，此时画面还需要加载和缓冲，当缓冲完毕后您会收到来自 onFirstVideoFrame 的通知。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTC_VIDEO_STREAM_TYPE_BIG。 低清小画面：TRTC_VIDEO_STREAM_TYPE_SMALL（需要远端用户通过 enableEncSmallVideoStream 开启双路编码后才有效果）。 辅流画面（常用于屏幕分享）：TRTC_VIDEO_STREAM_TYPE_SUB。
userId	指定远端用户的 ID。
view	用于承载视频画面的渲染控件。

注意

注意几点规则需要您关注：

1. SDK 支持同时观看某 userid 的大画面和辅路画面，或者同时观看某 userid 的小画面和辅路画面，但不支持同时观看大画面和小画面。
2. 只有当指定的 userid 通过 enableEncSmallVideoStream 开启双路编码后，才能观看该用户的小画面。
3. 当指定的 userid 的小画面不存在时，SDK 默认切换到该用户的大画面。

updateRemoteView

updateRemoteView

void updateRemoteView	(String userId
	int streamType
	TXCloudVideoView view)

更新远端用户的视频渲染控件

该接口可用于更新远端视频画面的渲染控件，常被用于切换显示区域的交互场景中。

参数	描述
streamType	要设置预览窗口的流类型（仅支持 TRTC_VIDEO_STREAM_TYPE_BIG 和 TRTC_VIDEO_STREAM_TYPE_SUB）。
userId	指定远端用户的 ID。
view	承载视频画面的控件。

stopRemoteView

stopRemoteView

void stopRemoteView	(String userId
	int streamType)

停止订阅远端用户的视频流，并释放渲染控件

调用此接口会让 SDK 停止接收该用户的视频流，并释放该路视频流的解码和渲染资源。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTC_VIDEO_STREAM_TYPE_BIG。

	低清小画面：TRTC_VIDEO_STREAM_TYPE_SMALL。 辅流画面（常用于屏幕分享）：TRTC_VIDEO_STREAM_TYPE_SUB。
userId	指定远端用户的 ID。

stopAllRemoteView

stopAllRemoteView

停止订阅所有远端用户的视频流，并释放全部渲染资源

调用此接口会让 SDK 停止接收所有来自远端的视频流，并释放全部的解码和渲染资源。

注意

如果当前有正在显示的辅路画面（屏幕分享）也会一并被停止。

muteRemoteVideoStream

muteRemoteVideoStream

void muteRemoteVideoStream	(String userId
	int streamType
	boolean mute)

暂停/恢复订阅远端用户的视频流

该接口仅暂停/恢复接收指定用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。
streamType	要暂停/恢复的视频流类型。 高清大画面：TRTC_VIDEO_STREAM_TYPE_BIG。 低清小画面：TRTC_VIDEO_STREAM_TYPE_SMALL。 辅流画面（常用于屏幕分享）：TRTC_VIDEO_STREAM_TYPE_SUB。
userId	指定远端用户的 ID。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。

版权所有：腾讯云计算（北京）有限责任公司

调用该接口暂停接收指定用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(false) 或 muteAllRemoteVideoStreams(false) 才能恢复。

muteAllRemoteVideoStreams

muteAllRemoteVideoStreams

void muteAllRemoteVideoStreams	(boolean mute)

暂停/恢复订阅所有远端用户的视频流

该接口仅暂停/恢复接收所有用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。
调用该接口暂停接收所有用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(false) 或 muteAllRemoteVideoStreams(false) 才能恢复。

setVideoEncoderParam

setVideoEncoderParam

void setVideoEncoderParam	(TRTCCloudDef.TRTCVideoEncParam param)

设置视频编码器的编码参数

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。

参数	描述
param	用于设置视频编码器的相关参数，详情请参见 TRTCVideoEncParam。

注意

从v11.5版本开始，编码输出分辨率会按照宽8高2字节对齐，并且是向下调整，eg:输入分辨率540x960，实际编码输出分辨率536x960。

版权所有：腾讯云计算（北京）有限责任公司

setNetworkQosParam

setNetworkQosParam

void setNetworkQosParam	(TRTCCloudDef.TRTCNetworkQosParam param)

设置网络质量控制的相关参数

该设置决定在差网络环境下的质量调控策略，如“画质优先”或“流畅优先”等策略。

参数	描述
param	用于设置网络质量控制的相关参数，详情请参见 TRTCNetworkQosParam。

setLocalRenderParams

setLocalRenderParams

void setLocalRenderParams	(TRTCCloudDef.TRTCRenderParams params)

设置本地画面的渲染参数

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述
params	画面渲染参数，详情请参见 TRTCRenderParams。

setRemoteRenderParams

setRemoteRenderParams

void setRemoteRenderParams	(String userId
	int streamType
	TRTCCloudDef.TRTCRenderParams params)

设置远端画面的渲染模式

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述

params	画面渲染参数，详情请参见 TRTCRenderParams。
streamType	可以设置为主路画面（TRTC_VIDEO_STREAM_TYPE_BIG）或辅路画面（TRTC_VIDEO_STREAM_TYPE_SUB）。
userId	指定远端用户的 ID。

enableEncSmallVideoStream

enableEncSmallVideoStream

int enableEncSmallVideoStream	(boolean enable
	TRTCCloudDef.TRTCVideoEncParam smallVideoEncParam)

开启大小画面双路编码模式

开启双路编码模式后，当前用户的编码器会同时输出【高清大画面】和【低清小画面】两路视频流（但只有一路音频流）。
如此以来，房间中的其他用户就可以根据自身的网络情况或屏幕大小选择订阅【高清大画面】或是【低清小画面】。

参数	描述
enable	是否开启小画面编码，默认值：false。
smallVideoEncParam	小流的视频参数。

注意

双路编码开启后，会消耗更多的 CPU 和 网络带宽，所以 Mac、Windows 或者高性能 Pad 可以考虑开启，不建议手机端开启。

返回值说明：

0：成功；-1：当前大画面已被设置为较低画质，开启双路编码已无必要。

setRemoteVideoStreamType

setRemoteVideoStreamType

int setRemoteVideoStreamType	(String userId
	int streamType)

切换指定远端用户的大小画面

当房间中某个主播开启了双路编码之后，房间中其他用户通过 startRemoteView 订阅到的画面默认会是【高清大画面】。
您可以通过此接口选定希望订阅的画面是大画面还是小画面，该接口在 startRemoteView 之前和之后调用均可生效。

参数	描述
streamType	视频流类型，即选择看大画面还是小画面，默认为大画面。
userId	指定远端用户的 ID。

注意

此功能需要目标用户已经通过 enableEncSmallVideoStream 提前开启了双路编码模式，否则此调用无实际效果。

snapshotVideo

snapshotVideo

void snapshotVideo	(String userId
	int streamType
	int sourceType
	TRTCCloudListener.TRTCSnapshotListener listener)

视频画面截图

您可以通过本接口截取本地的视频画面，远端用户的主路画面以及远端用户的辅路（屏幕分享）画面。

参数	描述
sourceType	画面来源，可选择截取视频流画面（TRTCSnapshotSourceTypeStream）、视频渲染画面（TRTCSnapshotSourceTypeView）或 采集画面（TRTCSnapshotSourceTypeCapture），采集画面截图更清晰。
streamType	视频流类型，可选择截取主路画面（TRTC_VIDEO_STREAM_TYPE_BIG，常用于摄像头）或辅路画面（TRTC_VIDEO_STREAM_TYPE_SUB，常用于屏幕分享）。
userId	用户 ID，如指定空置表示截取本地的视频画面。

注意

Windows 平台目前仅支持截取 TRTCSnapshotSourceTypeStream 来源的视频画面。

版权所有：腾讯云计算（北京）有限责任公司

setPerspectiveCorrectionPoints

setPerspectiveCorrectionPoints

void setPerspectiveCorrectionPoints	(String userId
	PointF[] srcPoints
	PointF[] dstPoints)

视频画面透视校正坐标设置

您可以通过本接口指定渲染画面透视校正的坐标区域。

参数	描述
dstPoints	期望校正到的目标坐标区域4个顶点坐标，需按照左上、左下、右上、右下的顺序传入，所有的坐标需要根据画面宽高做 [0,1] 区间的归一化；传 null 则停止透视校正。
srcPoints	原始画面坐标区域4个顶点坐标，需按照左上、左下、右上、右下的顺序传入，所有的坐标需要根据画面宽高做 [0,1] 区间的归一化; 传 null 则停止透视校正。
userId	用户 ID，如指定空值表示校正本地的视频画面。

setGravitySensorAdaptiveMode

setGravitySensorAdaptiveMode

void setGravitySensorAdaptiveMode	(int mode)

设置重力感应的适配模式（11.7 及以上版本）

开启重力感应后，如果采集端的设备发生旋转，采集端和观众端的画面都会进行相应地渲染以确保视野中的画面始终朝上。

只在sdk内部摄像头采集场景生效，并且只在移动端生效。

1. 该接口仅对采集端起作用，如果只是观看房间中的画面，开启此接口是无效的
2. 当采集端设备发生 90 度或 270 度旋转时，采集端或者观众看到的画面可能会被裁剪以保持比例的协调

参数	描述
mode	重力感应模式，详情请参见 TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_DISABLE、TRTC_GRAVITY_SENSOR_ADAPTIVE_I 和 TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_FIT_WITH_BLACK_BORDER 默认值：TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_DISABLE。

startLocalAudio

startLocalAudio

void startLocalAudio	(int quality)

开启本地音频的采集和发布

SDK 默认不开启麦克风，当用户需要发布本地音频时，需要调用该接口开启麦克风采集，并将音频编码并发布到当前的房间中。

开启本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, true) 的通知。

参数	描述
quality	声音音质 TRTC_AUDIO_QUALITY_SPEECH，流畅：采样率：16k；单声道；音频裸码率：16kbps；适合语音通话为主的场景，比如在线会议，语音通话。 TRTC_AUDIO_QUALITY_DEFAULT，默认：采样率：48k；单声道；音频裸码率：50kbps；SDK 默认的音频质量，如无特殊需求推荐选择之。 TRTC_AUDIO_QUALITY_MUSIC，高音质：采样率：48k；双声道 + 全频带；音频裸码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

注意

该函数会检查麦克风的使用权限，如果当前 App 没有麦克风权限，SDK 会自动向用户申请麦克风使用权限。

stopLocalAudio

stopLocalAudio

停止本地音频的采集和发布

停止本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, false) 的通知。

muteLocalAudio

muteLocalAudio

void muteLocalAudio	(boolean mute)

暂停/恢复发布本地的音频流

当您暂停发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, false) 的通知。

版权所有：腾讯云计算（北京）有限责任公司

当您恢复发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, true) 的通知。

与 stopLocalAudio 的不同之处在于，muteLocalAudio(true) 并不会释放麦克风权限，而是继续发送码率极低的静音包。
这对于需要云端录制的场景非常适用，因为 MP4 等格式的视频文件，对于音频数据的连续性要求很高，使用 stopLocalAudio 会导致录制出的 MP4 文件不易播放。
因此在对录制文件的质量要求较高的场景中，建议选择 muteLocalAudio 而不建议使用 stopLocalAudio。

参数	描述
mute	true：静音；false：恢复。

muteRemoteAudio

muteRemoteAudio

void muteRemoteAudio	(String userId
	boolean mute)

暂停/恢复播放远端的音频流

当您静音某用户的远端音频时，SDK 会停止播放指定用户的声音，同时也会停止拉取该用户的音频数据。

参数	描述
mute	true：静音；false：取消静音。
userId	用于指定远端用户的 ID。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 false。

muteAllRemoteAudio

muteAllRemoteAudio

void muteAllRemoteAudio	(boolean mute)

暂停/恢复播放所有远端用户的音频流

版权所有：腾讯云计算（北京）有限责任公司

当您静音所有用户的远端音频时，SDK 会停止播放所有来自远端的音频流，同时也会停止拉取所有用户的音频数据。

参数	描述
mute	true：静音；false：取消静音。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 false。

setAudioRoute

setAudioRoute

void setAudioRoute	(int route)

设置音频路由

设置“音频路由”，即设置声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。

手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

参数	描述
route	音频路由，即声音由哪里输出（扬声器、听筒），默认值：TRTC_AUDIO_ROUTE_SPEAKER。

setRemoteAudioVolume

setRemoteAudioVolume

void setRemoteAudioVolume	(String userId
	int volume)

设定某一个远端用户的声音播放音量

您可以通过 setRemoteAudioVolume(userId, 0) 将某一个远端用户的声音静音。

参数	描述

userId	用于指定远端用户的 ID。
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setAudioCaptureVolume

setAudioCaptureVolume

void setAudioCaptureVolume	(int volume)

设定本地音频的采集音量

参数	描述
volume	音量大小，取值范围为0 - 100；默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioCaptureVolume

getAudioCaptureVolume

获取本地音频的采集音量

setAudioPlayoutVolume

setAudioPlayoutVolume

void setAudioPlayoutVolume	(int volume)

设定远端音频的播放音量

该接口会控制 SDK 最终交给系统播放的声音音量，调节效果会影响到本地音频录制文件的音量大小，但不会影响到耳返的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioPlayoutVolume

getAudioPlayoutVolume

获取远端音频的播放音量

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(boolean enable
	TRTCCloudDef.TRTCAudioVolumeEvaluateParams params)

启用音量大小提示

开启此功能后，SDK 会在 TRTCCloudListener 中的 onUserVoiceVolume 回调中反馈本地或远端用户的音频音量评估信息，包括音量大小、人声检测、音频频谱等。

参数	描述
enable	是否启用音量提示，默认为关闭状态。
params	音量评估等相关参数，请参见 TRTCAudioVolumeEvaluateParams。

startAudioRecording

startAudioRecording

int startAudioRecording	(TRTCCloudDef.TRTCAudioRecordingParams param)

开始录音

当您调用该接口后， SDK 会将本地和远端的所有音频（包括本地音频，远端音频，背景音乐和音效等）混合并录制到一个本地文件中。
该接口在进入房间前后调用均可生效，如果录制任务在退出房间前尚未通过 stopAudioRecording 停止，则退出房间后录制任务会自动被停止。
本次录制的启动、完成状态会通过本地录制相关回调进行通知。参见 TRTCCloud 相关回调。

参数	描述
param	录音参数，请参见 TRTCAudioRecordingParams。

注意

自 v11.5 版本，音频录制的状态结果由返回值统一调整为异步回调进行通知。参见 TRTCCloud 相关回调。

返回值说明：

0：成功；-1：录音已开始；-2：文件或目录创建失败；-3：后缀指定的音频格式不支持。

stopAudioRecording

stopAudioRecording

停止录音

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

startLocalRecording

startLocalRecording

void startLocalRecording	(TRTCCloudDef.TRTCLocalRecordingParams params)

开启本地媒体录制

开启后把直播过程中的音视频内容录制到本地的一个文件中。

参数	描述
params	录制参数，请参见 TRTCLocalRecordingParams。

stopLocalRecording

stopLocalRecording

停止本地媒体录制

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

setRemoteAudioParallelParams

setRemoteAudioParallelParams

void setRemoteAudioParallelParams	(TRTCCloudDef.TRTCAudioParallelParams params)

设置远端音频流智能并发播放策略

设置远端音频流智能并发播放策略，适用于上麦人数比较多的场景。

参数	描述
params	音频并发参数，请参见 TRTCAudioParallelParams。

enable3DSpatialAudioEffect

enable3DSpatialAudioEffect

void enable3DSpatialAudioEffect	(boolean enabled)

启用 3D 音效

启用 3D 音效。注意需使用流畅音质 TRTC_AUDIO_QUALITY_SPEECH 或默认音质 TRTC_AUDIO_QUALITY_DEFAULT。

参数	描述
enabled	是否启用 3D 音效，默认为关闭状态。

updateSelf3DSpatialPosition

updateSelf3DSpatialPosition

void updateSelf3DSpatialPosition	(int[] position
	float[] axisForward

	float[] axisRight
	float[] axisUp)

设置 3D 音效中自身坐标及朝向信息

更新自身在世界坐标系中的位置和朝向， SDK 会根据该方法参数计算自身和远端用户之间的相对位置，进而渲染出空间音效。注意各参数应分别传入长度为 3 的数组。

参数	描述
axisForward	自身坐标系前轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisRight	自身坐标系右轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisUp	自身坐标系上轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
position	自身在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。

注意

请适当限制调用频率，推荐两次坐标设置至少间隔 100ms。

updateRemote3DSpatialPosition

updateRemote3DSpatialPosition

void updateRemote3DSpatialPosition	(String userId
	int[] position)

设置 3D 音效中远端用户坐标信息

更新远端用户在世界坐标系中的位置，SDK 会根据自身和远端用户之间的相对位置，进而渲染出空间音效。注意参数为长度等于 3 的数组。

参数	描述
position	该远端用户在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。
userId	指定远端用户的 ID。

注意

请适当限制调用频率，推荐同一远端用户两次坐标设置至少间隔 100ms。

版权所有：腾讯云计算（北京）有限责任公司

set3DSpatialReceivingRange

set3DSpatialReceivingRange

void set3DSpatialReceivingRange	(String userId
	int range)

设置指定用户所发出声音的可被接收范围

设置该范围大小之后，该指定用户的声音将在该范围内可被听见，超出该范围将被衰减为 0。

参数	描述
range	声音最大可被接收范围。
userId	指定远端用户的 ID。

getDeviceManager

getDeviceManager

获取设备管理类（TXDeviceManager）

getBeautyManager

getBeautyManager

获取美颜管理类（TXBeautyManager）

通过美颜管理，您可以使用以下功能：
设置“磨皮”、“美白”、“红润”等美颜特效。
设置“大眼”、“瘦脸”、“V脸”、“下巴”、“短脸”、“小鼻”、“亮眼”、“白牙”、“祛眼袋”、“祛皱纹”、“祛法令纹”等修脸特效。
设置“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”等修脸特效。
设置“眼影”、“腮红”等美妆特效。
设置动态贴纸和人脸挂件等动画特效。

setWatermark

版权所有：腾讯云计算（北京）有限责任公司

setWatermark

void setWatermark	(Bitmap image
	int streamType
	float x
	float y
	float width)

添加水印

水印的位置是通过 rect 参数来指定的，rect 是一个四元组参数，其格式为 (x，y，width，height)
x：水印的坐标，取值范围为0 - 1的浮点数。
y：水印的坐标，取值范围为0 - 1的浮点数。
width：水印的宽度，取值范围为0 - 1的浮点数。
height：是不用设置的，SDK 内部会根据水印图片的宽高比自动计算一个合适的高度。

参数设置举例：
如果当前视频的编码分辨率是 540 × 960，且 rect 参数被您设置为（0.1，0.1，0.2，0.0），
那么水印的左上坐标点就是（540 × 0.1，960 × 0.1）即（54，96），水印的宽度是 540 × 0.2 = 108px，水印的高度会根据水印图片的宽高比由 SDK 自动算出。

参数	描述
image	水印图片，**必须使用透明底色的 png 格式**。
rect	水印相对于编码分辨率的归一化坐标，x，y，width，height 取值范围0 - 1。
streamType	指定给哪一路画面设置水印，详情请参见TRTCVideoStreamType。

注意
如果您要给主画面（一般为摄像头）和辅路画面（一般用作屏幕分享）同时设置水印，需要调用该接口两次，并设定不同的 streamType。

getAudioEffectManager

getAudioEffectManager

获取音效管理类（TXAudioEffectManager）

TXAudioEffectManager 是音效管理接口，您可以通过该接口实现如下功能：
背景音乐：支持在线音乐和本地音乐，支持变速、变调等特效、支持原生和伴奏并播放和循环播放。
耳机耳返：麦克风捕捉的声音实时通过耳机播放，常用于音乐直播。
混响效果：KTV、小房间、大会堂、低沉、洪亮...。
变声特效：萝莉、大叔、重金属...。
短音效：鼓掌声、欢笑声等简短的音效文件（对于小于10秒的文件，请将 isShortFile 参数设置为 true）。

startSystemAudioLoopback

startSystemAudioLoopback

开启系统声音采集

该接口会采集其他应用音频数据，并将其混入到 SDK 当前的音频数据流中，从而使房间中的其他用户也能听到主播的其他应用所播放出的声音。
在线教育场景中，老师可以使用此功能让 SDK 采集教学影片中的声音，并广播给同房间中的学生。
音乐直播场景中，主播可以使用此功能让 SDK 采集音乐播放器中的音乐，从而为自己的直播间增加背景音乐。
注意
1. 该接口只在 Android API 29 及以上的版本生效。
2. 您需要先使用该接口来开启系统声音采集，当使用屏幕分享接口开启屏幕分享时才会真正生效。
3. 您需要添加一个前台服务来确保系统声音采集不被静默，并设置 android:foregroundServiceType="mediaProjection"。
4. SDK 只采集满足捕获策略和音频用法的应用音频，目前采集的音频用法包括USAGE_MEDIA, USAGE_GAME。

stopSystemAudioLoopback

stopSystemAudioLoopback

停止系统声音采集（iOS 端暂未支持）

startScreenCapture

startScreenCapture

void startScreenCapture	(int streamType
	TRTCCloudDef.TRTCVideoEncParam encParams
	TRTCCloudDef.TRTCScreenShareParams shareParams)

启动屏幕分享

该接口支持抓取整个 Android 系统的屏幕，可以实现类似腾讯会议的全系统级的屏幕分享。

参考文档：实时屏幕分享(Android)

Android 推荐的屏幕分享视频编码参数（TRTCVideoEncParam）：
分辨率(videoResolution)：1280×720。
帧率(videoFps)：10 FPS。
码率(videoBitrate)：1200 kbps。
分辨率自适应(enableAdjustRes)：false。

参数	描述
encParams	编码参数，请参见 TRTCVideoEncParam。如果您指定 encParams 为 null，SDK 会自动使用之前设定的编码参数。
shareParams	请参见 TRTCScreenShareParams，您可以通过其中的 floatingView 参数弹出悬浮窗（也可以使用 Android 系统的 WindowManager 接口自行弹出）。
streamType	屏幕分享使用的线路，可以设置为主路（TRTC_VIDEO_STREAM_TYPE_BIG）或者辅路（TRTC_VIDEO_STREAM_TYPE_SUB）。

stopScreenCapture

stopScreenCapture

停止屏幕分享

pauseScreenCapture

pauseScreenCapture

暂停屏幕分享

注意
从v11.5版本开始，暂停屏幕采集会使用最后一帧按照1fps帧率输出。

resumeScreenCapture

resumeScreenCapture

恢复屏幕分享

setSubStreamEncoderParam

setSubStreamEncoderParam

void setSubStreamEncoderParam	(TRTCCloudDef.TRTCVideoEncParam param)

设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）

该接口可以设定远端用户所看到的屏幕分享（即辅路）的画面质量，同时也能决定云端录制出的视频文件中屏幕分享的画面质量。

请注意如下两个接口的差异：

setVideoEncoderParam 用于设置主路画面（TRTC_VIDEO_STREAM_TYPE_BIG，一般用于摄像头）的视频编码参数。

setSubStreamEncoderParam 用于设置辅路画面（TRTC_VIDEO_STREAM_TYPE_SUB，一般用于屏幕分享）的视频编码参数。

参数	描述
param	辅流编码参数，详情请参见 TRTCVideoEncParam。

enableCustomVideoCapture

enableCustomVideoCapture

void enableCustomVideoCapture	(int streamType
	boolean enable)

启用/关闭视频自定义采集模式

开启该模式后，SDK 不在运行原有的视频采集流程，即不再继续从摄像头采集数据和美颜，而是只保留视频编码和发送能力。

您需要通过 sendCustomVideoData 不断地向 SDK 塞入自己采集的视频画面。

参数	描述
enable	是否启用，默认值：false。
streamType	用于指定视频流类型，TRTC_VIDEO_STREAM_TYPE_BIG：高清大画

	面；TRTC_VIDEO_STREAM_TYPE_SUB：辅路画面。

sendCustomVideoData

sendCustomVideoData

void sendCustomVideoData	(int streamType
	TRTCCloudDef.TRTCVideoFrame frame)

向 SDK 投送自己采集的视频帧

使用此接口可以向 SDK 投送自己采集的视频帧，SDK 会将视频帧进行编码并通过自身的网络模块传输出去。
Android 平台有两种投送方案：
内存传递方案：对接起来比较简单，但是性能较差，不适合分辨率较高的场景。
显存传递方案：对接需要一定的 OpenGL 基础，但是性能较好，640 × 360 以上的分辨率请采用该方案。

参考文档：自定义采集和渲染。

参数	描述
frame	视频数据，如果是内存传递方案，请设置 data 字段；如果是显存传递方案，请设置 TRTCTexture 字段。详情请参见 TRTCVideoFrame。
streamType	用于指定视频流类型，TRTC_VIDEO_STREAM_TYPE_BIG：高清大画面；TRTC_VIDEO_STREAM_TYPE_SUB：辅路画面。

注意

1. 推荐您在采集到的一帧视频帧后，即调用 generateCustomPTS 接口获取该帧的 timestamp 数值，这样可以获得最佳的音画同步效果。
2. SDK 最终编码出的视频帧率并不是由您调用本接口的频率决定的，而是由您在 setVideoEncoderParam 中所设置的 FPS 决定的。
3. 请尽量保持本接口的调用间隔是均匀的，否则会导致编码器输出帧率不稳或者音画不同步等问题。

enableCustomAudioCapture

enableCustomAudioCapture

void enableCustomAudioCapture	(boolean enable)

启用音频自定义采集模式

开启该模式后，SDK 不在运行原有的音频采集流程，即不再继续从麦克风采集音频数据，而是只保留音频编码和发送能力。
您需要通过 sendCustomAudioData 不断地向 SDK 塞入自己采集的音频数据。

参数	描述
enable	是否启用，默认值：false。

注意

由于回声抵消（AEC）需要严格的控制声音采集和播放的时间，所以开启自定义音频采集后，AEC 能力可能会失效。

sendCustomAudioData

sendCustomAudioData

void sendCustomAudioData	(TRTCCloudDef.TRTCAudioFrame frame)

向 SDK 投送自己采集的音频数据

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：
audioFormat：音频数据格式，仅支持 TRTCAudioFrameFormatPCM。
data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。
sampleRate：采样率，支持：16000、24000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在采集到一帧音频帧之后，通过调用 generateCustomPTS 获取时间戳）。

参考文档：自定义采集和渲染。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

enableMixExternalAudioFrame

enableMixExternalAudioFrame

void enableMixExternalAudioFrame	(boolean enablePublish
	boolean enablePlayout)

启用/关闭自定义音轨

开启后，您可以通过本接口向 SDK 混入一条自定义的音轨。通过两个布尔型参数，您可以控制该音轨是否要在远端和本地播放。

参数	描述
enablePlayout	控制混入的音轨是否要在本地播放，默认值：false。
enablePublish	控制混入的音轨是否要在远端播放，默认值：false。

注意

如果您指定参数 enablePublish 和 enablePlayout 均为 false，代表完全关闭您的自定义音轨。

mixExternalAudioFrame

mixExternalAudioFrame

int mixExternalAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

向 SDK 混入自定义音轨

调用该接口之前，您需要先通过 enableMixExternalAudioFrame 开启自定义音轨，之后就可以通过本接口将自己的音轨以 PCM 格式混入到 SDK 中。

理想情况下，我们期望您的代码能够以非常均匀的速度向 SDK 提供音轨数据。但我们也非常清楚，完美的调用间隔是一个巨大的挑战。

所以 SDK 内部会开启一个音轨数据的缓冲区，该缓冲区的作用类似一个“蓄水池”，它能够暂存您传入的音轨数据，平抑由于接口调用间隔的抖动问题。

本接口的返回值代表这个音轨缓冲区的大小，单位是毫秒（ms），比如：如果该接口返回 50，则代表当前的音轨缓冲区有 50ms 的音轨数据。因此只要您在 50ms 内再次调用本接口，SDK 就能保证您混入的音轨数据是连续的。

当您调用该接口后，如果发现返回值 > 100ms，则可以等待一帧音频帧的播放时间之后再次调用；如果返回值 < 100ms，则代表缓冲区比较小，您可以再次混入一些音轨数据以确保音轨缓冲区的大小维持在“安全水位”以上。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：

data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。

sampleRate：采样率，支持：16000、24000、32000、44100、48000。

channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。

timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在获得一帧音频帧之后，通过调用 generateCustomPTS 获得时间戳）。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

返回值说明：

>= 0 缓冲的长度，单位：ms。< 0 错误（-1 未启用 mixExternalAudioFrame）

setMixExternalAudioVolume

setMixExternalAudioVolume

void setMixExternalAudioVolume	(int publishVolume
	int playoutVolume)

设置推流时混入外部音频的推流音量和播放音量

参数	描述
playoutVolume	设置的播放音量大小，范围0 - 100，-1表示不改变。
publishVolume	设置的推流音量大小，范围0 - 100，-1表示不改变。

generateCustomPTS

generateCustomPTS

生成自定义采集时的时间戳

本接口仅适用于自定义采集模式，用于解决音视频帧的采集时间（capture time）和投送时间（send time）不一致所导致的音画不同步问题。

当您通过 sendCustomVideoData 或 sendCustomAudioData 等接口进行自定义视频或音频采集时，请按照如下操作使用该接口：
1. 首先，在采集到一帧视频或音频帧时，通过调用本接口获得当时的 PTS 时间戳。
2. 之后可以将该视频或音频帧送入您使用的前处理模块（如第三方美颜组件，或第三方音效组件）。
3. 在真正调用 sendCustomVideoData 或 sendCustomAudioData 进行投送时，请将该帧在采集时记录的 PTS 时间戳赋值给 TRTCVideoFrame 或 TRTCAudioFrame 中的 timestamp 字段。

返回值说明：

时间戳（单位：ms）

setLocalVideoProcessListener

setLocalVideoProcessListener

int setLocalVideoProcessListener	(int pixelFormat
	int bufferType
	TRTCCloudListener.TRTCVideoFrameListener listener)

设置第三方美颜的视频数据回调

设置该回调之后，SDK 会把采集到的视频帧通过您设置的 listener 回调出来，用于第三方美颜组件进行二次处理，之后 SDK 会将处理后的视频帧进行编码和发送。

参数	描述
bufferType	指定回调的数据格式，目前支持： TRTC_VIDEO_BUFFER_TYPE_TEXTURE：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_Texture_2D 时。 TRTC_VIDEO_BUFFER_TYPE_BYTE_BUFFER：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_I420 时。 TRTC_VIDEO_BUFFER_TYPE_BYTE_ARRAY：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_I420 时。
listener	自定义预处理回调，详见 TRTCVideoFrameListener。
pixelFormat	指定回调的像素格式，目前支持： TRTC_VIDEO_PIXEL_FORMAT_Texture_2D: 显存纹理方案。 TRTC_VIDEO_PIXEL_FORMAT_I420：内存数据方案。

版权所有：腾讯云计算（北京）有限责任公司

返回值说明：

0：成功；< 0：错误

setLocalVideoRenderListener

setLocalVideoRenderListener

int setLocalVideoRenderListener	(int pixelFormat
	int bufferType
	TRTCCloudListener.TRTCVideoRenderListener listener)

设置本地视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
pixelFormat 指定回调的数据格式，目前支持 Texture2D、I420 和 RGBA 三种格式。
bufferType 指定 buffer 的类型，BYTE_BUFFER 适合在 jni 层使用，BYTE_ARRAY 则可用于 Java 层的直接操作。

参考文档：自定义采集和渲染。

参数	描述
bufferType	指定视频帧的数据结构： TRTC_VIDEO_BUFFER_TYPE_TEXTURE：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_Texture_2D 时。 TRTC_VIDEO_BUFFER_TYPE_BYTE_BUFFER：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_I420 和 TRTC_VIDEO_PIXEL_FORMAT_RGBA 时。 TRTC_VIDEO_BUFFER_TYPE_BYTE_ARRAY：适用于 pixelFormat 被设置为 TRTC_VIDEO_PIXEL_FORMAT_I420 和 TRTC_VIDEO_PIXEL_FORMAT_RGBA 时。
listener	自定义视频渲染回调，每一帧视频数据回调一次。
pixelFormat	指定视频帧的像素格式，如： TRTC_VIDEO_PIXEL_FORMAT_Texture_2D：OpenGL 纹理格式，比较适合 GPU 进行处理，处理效率高。 TRTC_VIDEO_PIXEL_FORMAT_I420：标准 i420 格式，比较适合 CPU 进行处理，处理效率较差。 TRTC_VIDEO_PIXEL_FORMAT_RGBA：RGBA 格式，比较适合 CPU 进行处理，处理效率较差。

返回值说明：

0：成功；< 0：错误。

setRemoteVideoRenderListener

setRemoteVideoRenderListener

int setRemoteVideoRenderListener	(String userId
	int pixelFormat
	int bufferType
	TRTCCloudListener.TRTCVideoRenderListener listener)

设置远端视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
pixelFormat 指定回调的数据格式，例如 NV12、i420 以及 32BGRA。
bufferType 指定 buffer 的类型，直接使用 PixelBuffer 效率最高；使用 NSData 相当于让 SDK 在内部做了一次内存转换，因此会有额外的性能损耗。

参考文档：自定义采集和渲染。

参数	描述
bufferType	指定视频帧的数据结构，如 TRTC_VIDEO_BUFFER_TYPE_BYTE_BUFFER，TRTC_VIDEO_BUFFER_TYPE_BYTE_ARRA
listener	自定义视频渲染回调，每一帧视频数据回调一次。
pixelFormat	指定视频帧的像素格式，目前支持 TRTC_VIDEO_PIXEL_FORMAT_I420，TRTC_VIDEO_PIXEL_FORMAT_RGBA。
userId	指定远端用户的 ID。

注意

实际使用时，需要先调用 startRemoteView(userid, null) 启动远程视频流的拉取，并将 view 设置为 null，否则不会有数据回调出来。

返回值说明：

0：成功；< 0：错误。

setAudioFrameListener

setAudioFrameListener

void setAudioFrameListener	(TRTCCloudListener.TRTCAudioFrameListener listener)

设置音频数据自定义回调

设置该回调之后，SDK 内部会把音频数据（PCM 格式）回调出来，包括：

onCapturedAudioFrame：本地麦克风采集到的音频数据回调

onLocalProcessedAudioFrame：本地采集并经过音频模块前处理后的音频数据回调

onRemoteUserAudioFrame：混音前的每一路远程用户的音频数据

onMixedPlayAudioFrame：将各路音频混合之后并最终要由系统播放出的音频数据回调

注意

设置回调为空即代表停止自定义音频回调，反之，设置回调不为空则代表启动自定义音频回调。

setCapturedAudioFrameCallbackFormat

setCapturedAudioFrameCallbackFormat

int setCapturedAudioFrameCallbackFormat	(TRTCCloudDef.TRTCAudioFrameCallbackFormat format)

设置本地麦克风采集出的音频帧回调格式

本接口用于设置 onCapturedAudioFrame 回调出来的 AudioFrame 的格式：

sampleRate：采样率，支持：16000、32000、44100、48000。

channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。

samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。

举例：48000 采样率希望回调 20ms 帧长的数据，则采样点数应该填：960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）

举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setLocalProcessedAudioFrameCallbackFormat

setLocalProcessedAudioFrameCallbackFormat

int setLocalProcessedAudioFrameCallbackFormat	(TRTCCloudDef.TRTCAudioFrameCallbackFormat format)

设置经过前处理后的本地音频帧回调格式

本接口用于设置 onLocalProcessedAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意
最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setMixedPlayAudioFrameCallbackFormat

setMixedPlayAudioFrameCallbackFormat

int setMixedPlayAudioFrameCallbackFormat	(TRTCCloudDef.TRTCAudioFrameCallbackFormat format)

设置最终要由系统播放出的音频帧回调格式

本接口用于设置 onMixedPlayAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

enableCustomAudioRendering

enableCustomAudioRendering

void enableCustomAudioRendering	(boolean enable)

开启音频自定义播放

如果您需要外接一些特定的音频设备，或者希望自己掌控音频的播放逻辑，您可以通过该接口启用音频自定义播放。
启用音频自定义播放后，SDK 将不再调用系统的音频接口播放数据，您需要通过 getCustomAudioRenderingFrame 获取 SDK 要播放的音频帧并自行播放。

参数	描述
enable	是否启用音频自定义播放，默认为关闭状态。

注意

需要您在进入房间前设置才能生效，暂不支持进入房间后再设置。

getCustomAudioRenderingFrame

getCustomAudioRenderingFrame

void getCustomAudioRenderingFrame	(final TRTCCloudDef.TRTCAudioFrame audioFrame)

获取可播放的音频数据

调用该接口之前，您需要先通过 enableCustomAudioRendering 开启音频自定义播放。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：
sampleRate：采样率，必填，支持 16000、24000、32000、44100、48000。
channel：声道数，必填，单声道请填1，双声道请填2，双声道时数据是交叉的。
data：用于获取音频数据的 buffer。需要您根据一帧音频帧的帧长度分配好 data 的内存大小。获取的 PCM 数据支持 10ms 或 20ms 两种帧长，推荐使用 20ms 的帧长。

计算公式为：采样率 x 播放时长 x 声道数量 x 2（每个样点的字节数），例如：48000采样率、单声道、且播放时长为 20ms 的一帧音频帧的 buffer 大小为 48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节。

参数	描述
audioFrame	音频数据帧。

注意

1. 参数 audioFrame 中的 sampleRate、channel 需提前设置好，同时分配好所需读取帧长的 data 空间。
2. SDK 内部会根据 sampleRate 和 channel 自动填充 data 数据。
3. 建议由系统的音频播放线程直接驱动该函数的调用，在播放完一帧音频之后，即调用该接口获取下一帧可播放的音频数据。

sendCustomCmdMsg

sendCustomCmdMsg

boolean sendCustomCmdMsg	(int cmdID
	byte[] data
	boolean reliable
	boolean ordered)

使用 UDP 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 UDP 通道，向当前房间里的其他用户广播自定义数据，以达到传输信令的目的。
房间中的其他用户可以通过 TRTCCloudListener 中的 onRecvCustomCmdMsg 回调接收消息。

参数	描述
cmdID	消息 ID，取值范围为1 - 10。

data	待发送的消息，单个消息的最大长度被限制为 1KB。
ordered	是否要求有序，即是否要求接收端的数据包顺序和发送端的数据包顺序一致（这会带来一定的接收延时）。
reliable	是否可靠发送，可靠发送可以获得更高的发送成功率，但可靠发送比不可靠发送会带来更大的接收延迟。

注意

1. 发送消息到房间内所有用户（暂时不支持 Web/小程序端），每秒最多能发送30条消息。
2. 每个包最大为 1KB，超过则很有可能会被中间路由器或者服务器丢弃。
3. 每个客户端每秒最多能发送总计 8KB 数据。
4. 请将 reliable 和 ordered 同时设置为 true 或同时设置为 false，暂不支持交叉设置。
5. 强烈建议您将不同类型的消息设定为不同的 cmdID，这样可以在要求有序的情况下减小消息时延。
6. 目前仅支持主播身份。

返回值说明：

true：消息已经发出；false：消息发送失败。

sendSEIMsg

sendSEIMsg

boolean sendSEIMsg	(byte[] data
	int repeatCount)

使用 SEI 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 SEI 通道，向当前房间里的其他用户广播自定义数据，已达到传输信令的目的。

视频帧的头部有一个叫做 SEI 的头部数据块，该接口的原理就是利用这个被称为 SEI 的头部数据块，将您要发送的自定义信令嵌入其中，使其同视频帧一并发送出去。
因此，与 sendCustomCmdMsg 相比，SEI 通道传输的信令具有更好的兼容性：信令可以伴随着视频帧一直传输到直播 CDN 上。
不过，由于视频帧头部的数据块不能太大，建议您使用该接口时，尽量将信令控制在几个字节的大小。

最常见的用法是把自定义的时间戳（timestamp）用本接口嵌入视频帧中，实现消息和画面的完美对齐（比如：教育场景下的课件和视频信号的对齐）。
房间中的其他用户可以通过 TRTCCloudListener 中的 onRecvSEIMsg 回调接收消息。

参数	描述
data	待发送的数据，最大支持 1KB（1000字节）的数据大小
repeatCount	发送数据次数

注意

本接口有以下限制：

1. 数据在接口调用完后不会被即时发送出去，而是从下一帧视频帧开始与视频帧一起发送。
2. 发送消息到房间内所有用户，每秒最多能发送 30 条消息（与 sendCustomCmdMsg 共享限制）。
3. 每个包最大为 1KB，若发送大量数据，会导致视频码率增大，可能导致视频画质下降甚至卡顿（与 sendCustomCmdMsg 共享限制）。
4. 每个客户端每秒最多能发送总计8KB数据（与 sendCustomCmdMsg 共享限制）。
5. 若指定多次发送（repeatCount > 1），则数据会被带在后续的连续 repeatCount 个视频帧中发送出去，同样会导致视频码率增大。
6. 如果 repeatCount > 1，多次发送，接收消息 onRecvSEIMsg 回调也可能会收到多次相同的消息，需要去重。

返回值说明：

true：消息已通过限制，等待后续视频帧发送；false：消息被限制发送

startSpeedTest

startSpeedTest

int startSpeedTest	(TRTCCloudDef.TRTCSpeedTestParams params)

开始进行网速测试（进入房间前使用）

参数	描述
params	测速选项

注意

1. 测速过程将产生少量的基础服务费用，详见 计费概述 > 基础服务 文档说明。
2. 请在进入房间前进行网速测试，在房间中网速测试会影响正常的音视频传输效果，而且由于干扰过多，网速测试结果也不准确。
3. 同一时间只允许一项网速测试任务运行。

返回值说明：

接口调用结果，< 0：失败

stopSpeedTest

stopSpeedTest

停止网络测速

getSDKVersion

getSDKVersion

获取 SDK 版本信息

setLogLevel

setLogLevel

void setLogLevel	(int level)

设置 Log 输出级别

参数	描述
level	参见 TRTCLogLevel，默认值：TRTCLogLevelNone

setConsoleEnabled

setConsoleEnabled

void setConsoleEnabled	(boolean enabled)

启用/禁用控制台日志打印

参数	描述
enabled	指定是否启用，默认：禁止状态。

setLogCompressEnabled

setLogCompressEnabled

void setLogCompressEnabled	(boolean enabled)

启用/禁用日志的本地压缩

开启压缩后，Log 存储体积明显减小，但需要腾讯云提供的 Python 脚本解压后才能阅读。
禁用压缩后，Log 采用明文存储，可以直接用记事本打开阅读，但占用空间较大。

参数	描述
enabled	指定是否启用，默认为启动状态

setLogDirPath

setLogDirPath

void setLogDirPath	(String path)

设置本地日志的保存路径

通过该接口您可以更改 SDK 本地日志的默认存储路径，SDK 默认的本地日志的存储位置：
Windows 平台：在 C:/Users/[系统用户名]/AppData/Roaming/liteav/log，即 %appdata%/liteav/log 下。
iOS 或 Mac 平台：在 sandbox Documents/log 下。
Android 平台：在 /app私有目录/files/log/liteav/ 下。

参数	描述
path	存储日志的路径

注意

请务必在所有其他接口之前调用，并且保证您指定的目录是存在的，并且您的应用程序拥有对该目录的读写权限。

setLogListener

setLogListener

void setLogListener	(final TRTCCloudListener.TRTCLogListener logListener)

设置日志回调

showDebugView

showDebugView

void showDebugView	(int showType)

显示仪表盘

“仪表盘”是位于视频渲染控件之上的一个半透明的调试信息浮层，用于展示音视频信息和事件信息，便于对接和调试。

参数	描述
showType	0：不显示；1：显示精简版（仅显示音视频信息）；2：显示完整版（包含音视频信息和事件信息）。

TRTCViewMargin

TRTCViewMargin

public TRTCViewMargin	(float leftMargin
	float rightMargin
	float topMargin
	float bottomMargin)

设置仪表盘的边距

用于调整仪表盘在视频渲染控件中的位置，必须在 showDebugView 之前调用才能生效。

参数	描述
margin	仪表盘内边距，注意这里是基于 parentView 的百分比，margin 的取值范围是0 - 1。
userId	用户 ID。

callExperimentalAPI

callExperimentalAPI

String callExperimentalAPI	(String jsonStr)

调用实验性接口

enablePayloadPrivateEncryption

enablePayloadPrivateEncryption

int enablePayloadPrivateEncryption	(boolean enabled
	TRTCCloudDef.TRTCPayloadPrivateEncryptionConfig config)

开启或关闭媒体流私有加密

在安全要求较高的场景下，TRTC 建议您在加入房间前，调用 enablePayloadPrivateEncryption 方法开启媒体流私有加密。
用户退出房间后，SDK 会自动关闭私有加密。如需重新开启私有加密，您需要在用户再次加入房间前调用该方法。

参数	描述
config	配置媒体流私有加密的算法和密钥，参见 TRTCPayloadPrivateEncryptionConfig。
enabled	是否开启媒体流私有加密。

注意

TRTC 已经内置对媒体流进行加密后再传输，启用媒体流私有加密后将使用您传入的密匙与初始向量进行再次加密。

返回值说明：

接口调用结果，0: 方法调用成功， -1: 传入参数无效， -2: 功能已过期。若需解锁：请前往中国大陆站点开通 TRTC 旗舰版套餐，并填写 **申请**，审核通过后方可使用。

TRTCCloudListener

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTCCloudListener @ TXLiteAVSDK

Function: 腾讯云实时音视频的事件回调接口

TRTCCloudListener

TRTCVideoRenderListener

函数列表	描述
onRenderVideoFrame	自定义视频渲染回调

TRTCVideoFrameListener

函数列表	描述
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame	用于对接第三方美颜组件的视频处理回调
onGLContextDestory	SDK 内部 OpenGL 环境被销毁的通知

TRTCAudioFrameListener

函数列表	描述
onCapturedAudioFrame	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onRemoteUserAudioFrame	混音前的每一路远程用户的音频数据

onMixedPlayAudioFrame	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）
onVoiceEarMonitorAudioFrame	耳返的音频数据

TRTCLogListener

函数列表	描述
onLog	本地 LOG 的打印回调

TRTCCloudListener

函数列表	描述
onError	错误事件回调
onWarning	警告事件回调
onEnterRoom	进入房间成功与否的事件回调
onExitRoom	离开房间的事件回调
onSwitchRole	切换角色的事件回调
onSwitchRoom	切换房间的结果回调
onConnectOtherRoom	请求跨房通话的结果回调
onDisConnectOtherRoom	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode	更改跨房主播上行能力的结果回调
onRemoteUserEnterRoom	有用户加入当前房间
onRemoteUserLeaveRoom	有用户离开当前房间
onUserVideoAvailable	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable	某远端用户发布/取消了辅路视频画面
onUserAudioAvailable	某远端用户发布/取消了自己的音频

版权所有：腾讯云计算（北京）有限责任公司

onFirstVideoFrame	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated	远端音频状态变化的事件回调
onUserVideoSizeChanged	用户视频大小发生改变回调
onNetworkQuality	网络质量的实时统计回调
onStatistics	音视频技术指标的实时统计回调
onSpeedTestResult	网速测试的结果回调
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onAudioRouteChanged	当前音频路由发生变化（仅适用于移动设备）
onUserVoiceVolume	音量大小的反馈回调
onRecvCustomCmdMsg	收到自定义消息的事件回调
onMissCustomCmdMsg	自定义消息丢失的事件回调
onRecvSEIMsg	收到 SEI 消息的回调
onStartPublishing	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig	设置云端混流的排版布局和转码参数的事件回调

版权所有：腾讯云计算（北京）有限责任公司

onStartPublishMediaStream	开始发布媒体流的事件回调
onUpdatePublishMediaStream	更新媒体流的事件回调
onStopPublishMediaStream	停止媒体流的事件回调
onCdnStreamStateChanged	RTMP/RTMPS 推流状态发生改变回调
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused	屏幕分享暂停的事件回调
onScreenCaptureResumed	屏幕分享恢复的事件回调
onScreenCaptureStopped	屏幕分享停止的事件回调
onLocalRecordBegin	本地录制任务已经开始的事件回调
onLocalRecording	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment	本地录制分片的事件回调
onLocalRecordComplete	本地录制任务已经结束的事件回调
onSnapshotComplete	本地截图完成的事件回调
onUserEnter	有主播加入当前房间（已废弃）
onUserExit	有主播离开当前房间（已废弃）
onAudioEffectFinished	音效播放已结束（已废弃）
onSpeedTest	服务器测速的结果回调（已废弃）

onRenderVideoFrame

onRenderVideoFrame

void onRenderVideoFrame	(String userId
	int streamType
	TRTCCloudDef.TRTCVideoFrame frame)

自定义视频渲染回调

当您设置了本地或者远端的视频自定义渲染回调之后，SDK 就会将原本要交给渲染控件进行渲染的视频帧通过此回调接口抛送给您，以便于您进行自定义渲染。

参数	描述
frame	待渲染的视频帧信息。
streamType	频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	视频源的 userId，如果是本地视频回调（setLocalVideoRenderDelegate），该参数可以忽略。

onGLContextCreated

onGLContextCreated

SDK 内部 OpenGL 环境已经创建的通知

onProcessVideoFrame

onProcessVideoFrame

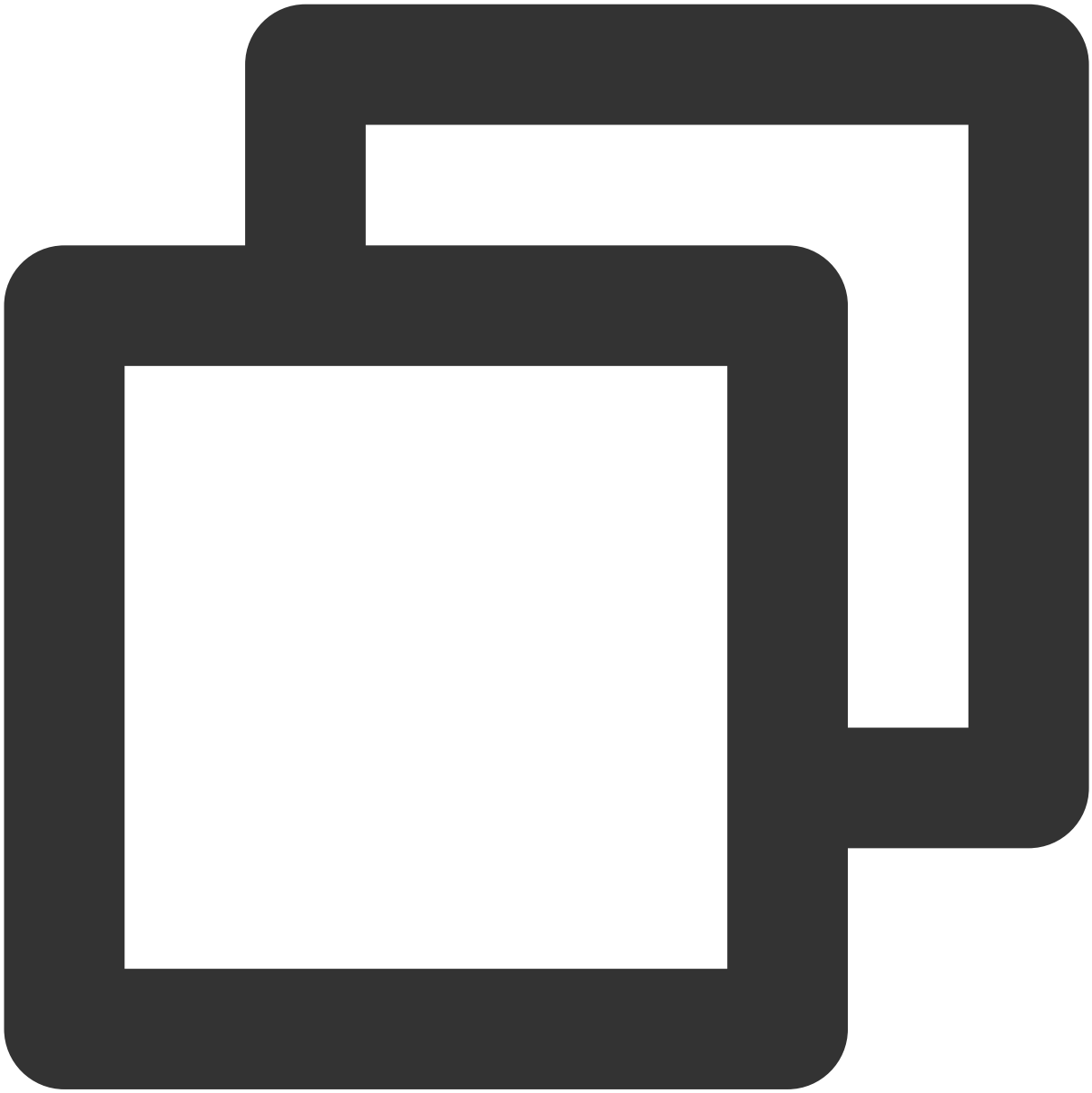
int onProcessVideoFrame	(TRTCCloudDef.TRTCVideoFrame srcFrame
	TRTCCloudDef.TRTCVideoFrame dstFrame)

用于对接第三方美颜组件的视频处理回调

如果您选购了第三方美颜组件，就需要在 TRTCCloud 中设置第三方美颜回调，之后 TRTC 就会将原本要进行预处理的视频帧通过此回调接口抛送给您。
之后您就可以将 TRTC 抛出的视频帧交给第三方美颜组件进行图像处理，由于抛出的数据是可读且可写的，因此第三方美颜的处理结果也可以同步给 TRTC 进行后续的编码和发送。

情况一：美颜组件自身会产生新的纹理。
如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函数中将 dstFrame.textureId 设置为新纹理的 ID：

版权所有：腾讯云计算（北京）有限责任公司

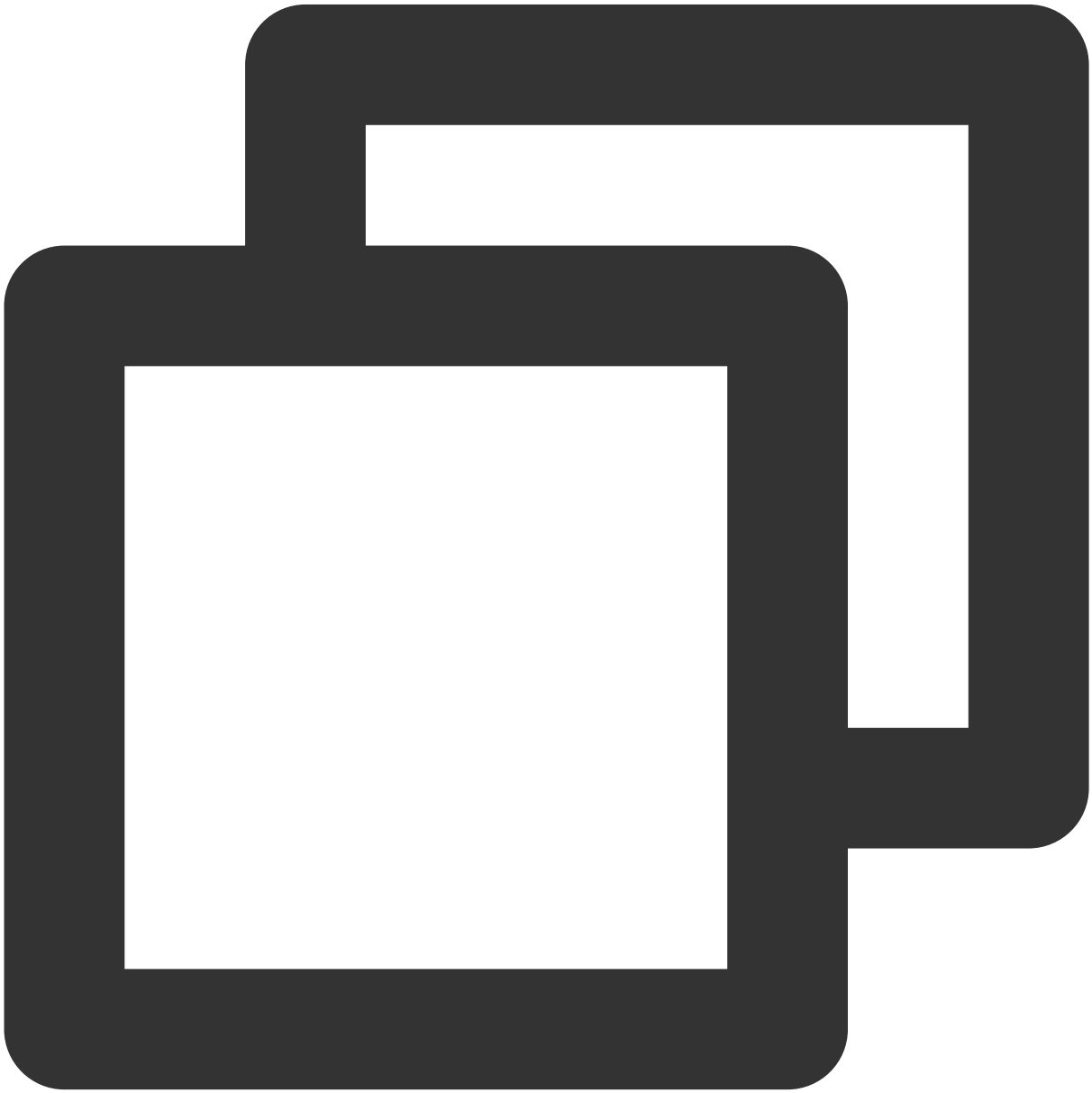


```
private final TRTCVideoFrameListener mVideoFrameListener = new TRTCVideoFrameListen
    @Override
    public void onGLContextCreated() {
        mFURenderer.onSurfaceCreated();
        mFURenderer.setUseTexAsync(true);
    }
    @Override
    public int onProcessVideoFrame(TRTCVideoFrame srcFrame, TRTCVideoFrame dstFrame
        dstFrame.texture.textureId = mFURenderer.onDrawFrameSingleInput(srcFrame.te
        return 0;
    }
```

```
    @Override
    public void onGLContextDestory() {
        mFURenderer.onSurfaceDestroyed();
    }
};
```

情况二：美颜组件需要您提供目标纹理。

如果您使用的第三方美颜模块并不生成新的纹理，而是需要您设置给该模块一个输入纹理和一个输出纹理，则可以考虑如下方案：



```
int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame, TRTCCloudDef.TRTCVide
```

```
    thirdparty_process(srcFrame.texture.textureId, srcFrame.width, srcFrame.height,
    return 0;
}
```

参数	描述
dstFrame	用于接收第三方美颜处理过的视频画面。
srcFrame	用于承载 TRTC 采集到的摄像头画面。

注意

目前仅支持 OpenGL 纹理方案（PC 仅支持 TRTCVideoBufferType_Buffer 格式）。

onGLContextDestory

onGLContextDestory

SDK 内部 OpenGL 环境被销毁的通知

onCapturedAudioFrame

onCapturedAudioFrame

void onCapturedAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

本地采集并经过音频模块前处理后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理(ANS、AEC、AGC）之后的数据，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的音频数据已经经过了前处理(ANS、AEC、AGC），但**不包含**背景音、音效、混响等前处理效果，延迟较低。

onLocalProcessedAudioFrame

onLocalProcessedAudioFrame

void onLocalProcessedAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理、音效处理和混 BGM 之后的数据，在最终进行网络编码之前，以 PCM 格式的形式通过本接口回调给您。
此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。
由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。
以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

特殊说明：
您可以通过设置接口中的 `TRTCAudioFrame.extraData` 字段，达到传输信令的目的。由于音频帧头部的数据块不能太大，建议您写入 `extraData` 时，尽量将信令控制在几个字节的大小，如果超过 100 个字节，写入的数据不会被发送。

房间内其他用户可以通过 TRTCAudioFrameListener 中的 `onRemoteUserAudioFrame` 中的 `TRTCAudioFrame.extraData` 字段回调接收数据。

参数	描述
frame	PCM 格式的音频数据帧。

注意
1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的数据已经经过了前处理(ANS、AEC、AGC）、音效和混 BGM 处理，声音的延迟相比于 onCapturedAudioFrame 要高一些。

版权所有：腾讯云计算（北京）有限责任公司

onRemoteUserAudioFrame

onRemoteUserAudioFrame

void onRemoteUserAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame
	String userId)

混音前的每一路远程用户的音频数据

当您设置完音频数据自定义回调之后，SDK 内部会把远端的每一路原始数据，在最终混音之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。
userId	用户标识。

注意

此接口回调出的音频数据是只读的，不支持修改。

onMixedPlayAudioFrame

onMixedPlayAudioFrame

void onMixedPlayAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

将各路待播放音频混合之后并在最终提交系统播放之前的数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把各路待播放的音频混合之后的音频数据，在提交系统播放之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述

frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的是对各路待播放音频数据的混合，但其中并不包含耳返的音频数据。

onMixedAllAudioFrame

onMixedAllAudioFrame

void onMixedAllAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

SDK 所有音频混合后的音频数据（包括采集到的和待播放的）

当您设置完音频数据自定义回调之后，SDK 内部会把所有采集到的和待播放的音频数据混合起来，以 PCM 格式的形式通过本接口回调给您，便于您进行自定义录制。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 此接口回调出的是SDK所有音频数据的混合数据，包括：经过 3A 前处理、特效叠加以及背景音乐混音后的本地音频，所有远端音频，但不包括耳返音频。
2. 此接口回调出的音频数据不支持修改。

onVoiceEarMonitorAudioFrame

onVoiceEarMonitorAudioFrame

void onVoiceEarMonitorAudioFrame	(TRTCCloudDef.TRTCAudioFrame frame)

耳返的音频数据

当您设置完音频数据自定义回调之后，SDK 内部会把耳返的音频数据在播放之前以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长不固定，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式 48000 采样率、单声道、16采样点位宽为例，0.02s 的音频数据字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，否则会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。

onLog

onLog

void onLog	(String log
	int level
	String module)

本地 LOG 的打印回调

如果您希望捕获 SDK 的本地日志打印行为，可以通过设置日志回调，让 SDK 将要打印的日志都通过本回调接口抛送给您。

参数	描述
level	日志等级请参见 TRTC_LOG_LEVEL。
log	日志内容。
module	保留字段，暂无具体意义，目前为固定值 TXLiteAVSDK。

onError

onError

void onError	(int errCode
	String errMsg
	Bundle extraInfo)

错误事件回调

错误事件，表示 SDK 抛出的不可恢复的错误，比如进入房间失败或设备开启失败等。

参考文档：**错误码表**

参数	描述
errCode	错误码
errMsg	错误信息
extInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题

onWarning

onWarning

void onWarning	(int warningCode
	String warningMsg
	Bundle extraInfo)

警告事件回调

警告事件，表示 SDK 抛出的提示性问题，比如视频出现卡顿或 CPU 使用率太高等。

参考文档：**错误码表**

参数	描述
extInfo	扩展信息字段，个别警告码可能会带额外的信息帮助定位问题
warningCode	警告码
warningMsg	警告信息

onEnterRoom

版权所有：腾讯云计算（北京）有限责任公司

onEnterRoom

void onEnterRoom	(long result)

进入房间成功与否的事件回调

调用 TRTCCloud 中的 enterRoom() 接口执行进房操作后，会收到来自 TRTCCloudListener 的 onEnterRoom(result) 回调：
如果加入成功，回调 result 会是一个正数（result > 0），代表进入房间所消耗的时间，单位是毫秒（ms）。
如果加入失败，回调 result 会是一个负数（result < 0），代表失败原因的错误码。
进房失败的错误码含义请参见**错误码表**。

参数	描述
result	result > 0 时为进房耗时（ms），result < 0 时为进房错误码。

注意

1. 在 Ver6.6 之前的版本，只有进房成功会抛出 onEnterRoom(result) 回调，进房失败由 onError() 回调抛出。
2. 在 Ver6.6 之后的版本：无论进房成功或失败，均会抛出 onEnterRoom(result) 回调，同时进房失败也会有 onError() 回调抛出。

onExitRoom

onExitRoom

void onExitRoom	(int reason)

离开房间的事件回调

调用 TRTCCloud 中的 exitRoom 接口会执行退出房间的相关逻辑，例如释放音视频设备资源和编解码器资源等。待 SDK 占用的所有资源释放完毕后，SDK 会抛出 onExitRoom() 回调通知到您。

如果您要再次调用 enterRoom 或者切换到其他的音视频 SDK，请等待 onExitRoom 回调到来后再执行相关操作。否则可能会遇到例如摄像头、麦克风设备被强占等各种异常问题。

参数	描述
reason	离开房间原因，0：主动调用 exitRoom 退出房间；1：被服务器踢出当前房间；2：当前房间整个被解散。

onSwitchRole

onSwitchRole

void onSwitchRole	(final int errCode
	final String errMsg)

切换角色的事件回调

调用 TRTCCloud 中的 switchRole() 接口会切换主播和观众的角色，该操作会伴随一个线路切换的过程，待 SDK 切换完成后，会抛出 onSwitchRole() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onSwitchRoom

onSwitchRoom

void onSwitchRoom	(final int errCode
	final String errMsg)

切换房间的结果回调

调用 TRTCCloud 中的 switchRoom() 接口可以让用户快速地从一个房间切换到另一个房间，待 SDK 切换完成后，会抛出 onSwitchRoom() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onConnectOtherRoom

onConnectOtherRoom

void onConnectOtherRoom	(final String userId
	final int errCode
	final String errMsg)

请求跨房通话的结果回调

调用 TRTCCloud 中的 connectOtherRoom() 接口会将两个不同房间中的主播拉通视频通话，也就是所谓的“主播 PK”功能。
调用者会收到 onConnectOtherRoom() 回调来获知跨房通话是否成功，
如果成功，两个房间中的所有用户都会收到来自另一个房间中的 PK 主播的 onUserVideoAvailable() 回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。
userId	要跨房通话的另一个房间中的主播的用户 ID。

onDisConnectOtherRoom

onDisConnectOtherRoom

void onDisConnectOtherRoom	(final int errCode
	final String errMsg)

结束跨房通话的结果回调

onUpdateOtherRoomForwardMode

onUpdateOtherRoomForwardMode

void onUpdateOtherRoomForwardMode	(final int errCode
	final String errMsg)

更改跨房主播上行能力的结果回调

onRemoteUserEnterRoom

onRemoteUserEnterRoom

void onRemoteUserEnterRoom	(String userId)

有用户加入当前房间

出于性能方面的考虑，在 TRTC 两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：
直播类场景（TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM）：该场景下的用户区分主播和观众两种角色，只有主播进入房间时才会触发该通知，观众进入房间时不会触发该通知。
通话类场景（TRTC_APP_SCENE_VIDEOCALL 和 TRTC_APP_SCENE_AUDIOCALL）：该场景下的用户没有角色的区分（可认为都是主播），任何用户进入房间都会触发该通知。

参数	描述
userId	远端用户的用户标识。

注意

1. 事件回调 onRemoteUserEnterRoom 和 onRemoteUserLeaveRoom 只适用于维护当前房间里的“用户列表”，有此事件回调不代表一定有视频画面。
2. 如果需要显示远程画面，请监听代表某个用户是否有视频画面的 onUserVideoAvailable() 事件回调。

onRemoteUserLeaveRoom

onRemoteUserLeaveRoom

void onRemoteUserLeaveRoom	(String userId
	int reason)

有用户离开当前房间

与 onRemoteUserEnterRoom 相对应，在两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：
直播类场景（TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM）：只有主播离开房间时才会触发该通知，观众离开房间不会触发该通知。
通话类场景（TRTC_APP_SCENE_VIDEOCALL 和 TRTC_APP_SCENE_AUDIOCALL）：该场景下用户没有角色的区别，任何用户的离开都会触发该通知。

参数	描述
reason	离开原因，0表示用户主动退出房间，1表示用户超时退出，2表示被踢出房间，3表示主播因切换到观众退出房间。
userId	远端用户的用户标识。

版权所有：腾讯云计算（北京）有限责任公司

onUserVideoAvailable

onUserVideoAvailable

void onUserVideoAvailable	(String userId
	boolean available)

某远端用户发布/取消了主路视频画面

主路画面 一般被用于承载摄像头画面。当您收到 onUserVideoAvailable(userId, true) 通知时，表示该路画面已经有可播放的视频帧到达。
此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

当您收到 onUserVideoAvailable(userId, false) 通知时，表示该路远程画面已经被关闭，关闭的原因可能是该用户调用了 muteLocalVideo 或 stopLocalPreview。

参数	描述
available	该用户是否发布（或取消发布）了主路视频画面，true：发布；false：取消发布。
userId	远端用户的用户标识。

onUserSubStreamAvailable

onUserSubStreamAvailable

void onUserSubStreamAvailable	(String userId
	boolean available)

某远端用户发布/取消了辅路视频画面

“辅路画面”一般被用于承载屏幕分享的画面。当您收到 onUserSubStreamAvailable(userId, true) 通知时，表示该路画面已经有可播放的视频帧到达。
此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

参数	描述
available	该用户是否发布（或取消发布）了辅路视频画面，true: 发布；false：取消发布。
userId	远端用户的用户标识。

注意

显示辅路画面使用的函数是 startRemoteView 而非 startRemoteSubStreamView，startRemoteSubStreamView 已经被废弃。

onUserAudioAvailable

onUserAudioAvailable

void onUserAudioAvailable	(String userId
	boolean available)

某远端用户发布/取消了自己的音频

当您收到 onUserAudioAvailable(userId, true) 通知时，表示该用户发布了自己的声音，此时 SDK 的表现为：
在自动订阅模式下，您无需做任何操作，SDK 会自动播放该用户的声音。
在手动订阅模式下，您可以通过 muteRemoteAudio(userid, false) 来播放该用户的声音。

参数	描述
available	该用户是否发布（或取消发布）了自己的音频，true: 发布；false：取消发布。
userId	远端用户的用户标识。

注意

SDK 默认使用自动订阅模式，您可以通过 setDefaultStreamRecvMode 设置为手动订阅，但需要在您进入房间之前调用才生效。

onFirstVideoFrame

onFirstVideoFrame

void onFirstVideoFrame	(String userId
	int streamType
	int width
	int height)

SDK 开始渲染自己本地或远端用户的首帧画面

SDK 会在渲染自己本地或远端用户的首帧画面时抛出该事件，您可以通过回调事件中的 userId 参数来判断事件来自于“本地”还是来自于“远端”。

如果 userId 为空值，代表 SDK 已经开始渲染自己本地的视频画面，不过前提是您已经调用了 startLocalPreview 或 startScreenCapture。

如果 userId 不为空，代表 SDK 已经开始渲染远端用户的视频画面，不过前提是您已经调用了 startRemoteView 订阅了该用户的视频画面。

参数	描述
height	画面的高度。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	本地或远端的用户标识，如果 userId 为空值代表自己本地的首帧画面已到来，userId 不为空则代表远端用户的首帧画面已到来。
width	画面的宽度。

注意

1. 只有当您调用了 startLocalPreview 或 startScreenCapture 之后，才会触发自己本地的首帧画面事件回调。
2. 只有当您调用了 startRemoteView 或 startRemoteSubStreamView 之后，才会触发远端用户的首帧画面事件回调。

onFirstAudioFrame

onFirstAudioFrame

void onFirstAudioFrame	(String userId)

SDK 开始播放远端用户的首帧音频

SDK 会在播放远端用户的首帧音频时抛出该事件，本地音频的首帧事件暂不抛出。

参数	描述
userId	远端用户的用户标识。

onSendFirstLocalVideoFrame

onSendFirstLocalVideoFrame

void onSendFirstLocalVideoFrame	(int streamType)

版权所有：腾讯云计算（北京）有限责任公司

自己本地的首个视频帧已被发布出去

当您成功进入房间并通过 startLocalPreview 或 startScreenCapture 开启本地视频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行视频编码，并通过自身的网络模块向云端发布自己本地的视频数据。当 SDK 成功地向云端送出自己的第一帧视频数据帧以后，就会抛出 onSendFirstLocalVideoFrame 事件回调。

参数	描述
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。

onSendFirstLocalAudioFrame

onSendFirstLocalAudioFrame

自己本地的首个音频帧已被发布出去

当您成功进入房间并通过 startLocalAudio 开启本地音频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行音频编码，并通过自身的网络模块向云端发布自己本地的音频数据。当 SDK 成功地向云端送出自己的第一帧音频数据帧以后，就会抛出 onSendFirstLocalAudioFrame 事件回调。

onRemoteVideoStatusUpdated

onRemoteVideoStatusUpdated

void onRemoteVideoStatusUpdated	(String userId
	int streamType
	int status
	int reason
	Bundle extraInfo)

远端视频状态变化的事件回调

您可以通过此事件回调获取远端每一路画面的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。

reason	视频状态改变的原因。
status	视频状态：包括 Playing、Loading 和 Stopped 三种状态。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onRemoteAudioStatusUpdated

onRemoteAudioStatusUpdated

void onRemoteAudioStatusUpdated	(String userId
	int status
	int reason
	Bundle extraInfo)

远端音频状态变化的事件回调

您可以通过此事件回调获取远端每一路音频的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。
reason	音频状态改变的原因。
status	音频状态：包括 Playing、Loading 和 Stopped 三种状态。
userId	用户标识。

onUserVideoSizeChanged

onUserVideoSizeChanged

void onUserVideoSizeChanged	(String userId
	int streamType
	int newWidth

	int newHeight)

用户视频大小发生改变回调

当您收到 onUserVideoSizeChanged(userId, streamtype, newWidth, newHeight) 通知时，表示该路画面大小发生了调整，调整的原因可能是该用户调用了 setVideoEncoderParam 或者 setSubStreamEncoderParam 重新设置了画面尺寸。

参数	描述
newHeight	视频流的高度（像素）。
newWidth	视频流的宽度（像素）。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onNetworkQuality

onNetworkQuality

void onNetworkQuality	(TRTCCloudDef.TRTCQuality localQuality
	ArrayList<TRTCCloudDef.TRTCQuality> remoteQuality)

网络质量的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 感知到的当前网络的上行和下行质量。
SDK 会使用一组内嵌的自研算法对当前网络的延迟高低、带宽大小以及稳定情况进行评估，并计算出一个评估结果：如果评估结果为 1（Excellent） 代表当前的网络情况非常好，如果评估结果为 6（Down）代表当前网络无法支撑 TRTC 的正常通话。

参数	描述
localQuality	上行网络质量。
remoteQuality	下行网络质量，代表数据流历经“远端->云端->本端”这样一条完整的传输链路后，最终在本端统计到的数据质量。因此，这里的下行网络质量代表的是远端上行链路与本端下行链路共同影响的结果。

注意

暂时无法通过该接口单独判定远端用户的上行质量。

onStatistics

onStatistics

void onStatistics	(TRTCStatistics statistics)

音视频技术指标的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 内部音频、视频以及网络相关的专业技术指标，这些信息在 TRTCStatistics 均有罗列：

视频统计信息：视频的分辨率（resolution）、帧率（FPS）和比特率（bitrate）等信息。

音频统计信息：音频的采样率（samplerate）、声道（channel）和比特率（bitrate）等信息。

网络统计信息：SDK 和云端一次往返（SDK > Cloud > SDK）的网络耗时（rtt）、丢包率（loss）、上行流量（sentBytes）和下行流量（receivedBytes）等信息。

参数	描述
statistics	统计数据，包括自己本地的统计信息和远端用户的统计信息，详情请参考 TRTCStatistics。

注意

如果您只需要获知当前网络质量的好坏，并不需要花太多时间研究本统计回调，更推荐您使用 onNetworkQuality 来解决问题。

onSpeedTestResult

onSpeedTestResult

void onSpeedTestResult	(TRTCCloudDef.TRTCSpeedTestResult result)

网速测试的结果回调

该统计回调由 startSpeedTest: 触发。

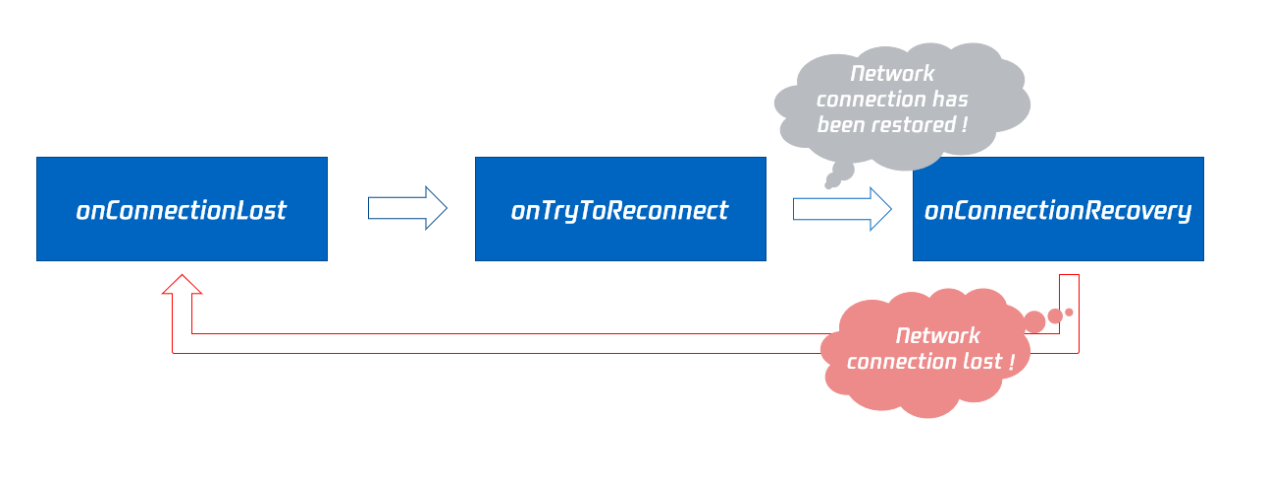
参数	描述
result	网速测试结果，包括丢包、往返延迟、上下行的带宽速率，详情请参见 TRTCSpeedTestResult。

onConnectionLost

onConnectionLost

SDK 与云端的连接已经断开

SDK 会在跟云端的连接断开时抛出此事件回调，导致断开的原因大多是网络不可用或者网络切换所致，比如用户在通话中走进电梯时就可能会遇到此事件。
在抛出此事件之后，SDK 会努力跟云端重新建立连接，重连过程中会抛出 onTryToReconnect，连接恢复后会抛出 onConnectionRecovery 。
所以，SDK 会在如下三个连接相关的事件中按如下规律切换：



onTryToReconnect

onTryToReconnect

SDK 正在尝试重新连接到云端

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出本事件，连接恢复后会抛出 onConnectionRecovery。

onConnectionRecovery

onConnectionRecovery

SDK 与云端的连接已经恢复

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出 onTryToReconnect，连接恢复后会抛出本事件回调。

onCameraDidReady

onCameraDidReady

摄像头准备就绪

当您调用 startLocalPreivew 之后，SDK 会尝试启动摄像头，如果摄像头能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问摄像头的权限，或者摄像头当前正在被其他程序独占使用中。
您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onMicDidReady

onMicDidReady

麦克风准备就绪

当您调用 startLocalAudio 之后，SDK 会尝试启动麦克风，如果麦克风能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问麦克风的权限，或者麦克风当前正在被其他程序独占使用中。
您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onAudioRouteChanged

onAudioRouteChanged

void onAudioRouteChanged	(int newRoute
	int oldRoute)

当前音频路由发生变化（仅适用于移动设备）

所谓“音频路由”，是指声音是从手机的扬声器还是从听筒中播放出来，音频路由变化也就是声音的播放位置发生了变化。
当音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
当音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。
当音频路由为有线耳机。
当音频路由为蓝牙耳机。
当音频路由为USB专业声卡设备。

参数	描述
fromRoute	变更前的音频路由。
route	音频路由，即声音由哪里输出（扬声器、听筒）。

onUserVoiceVolume

onUserVoiceVolume

void onUserVoiceVolume	(ArrayList<TRTCCloudDef.TRTCVolumeInfo> userVolumes
	int totalVolume)

音量大小的反馈回调

SDK 可以评估每一路音频的音量大小，并每隔一段时间抛出该事件回调，您可以根据音量大小在 UI 上做出相应的提示，比如 `波形图` 或 `音量槽` 。
要完成这个功能， 您需要先调用 enableAudioVolumeEvaluation 开启这个能力并设定事件抛出的时间间隔。
需要补充说明的是，无论当前房间中是否有人说话，SDK 都会按照您设定的时间间隔定时抛出此事件回调。

参数	描述
totalVolume	所有远端用户的总音量大小，取值范围 0 - 100。
userVolumes	是一个数组，用于承载所有正在说话的用户的音量大小，取值范围 0 - 100。

注意

userVolumes 为一个数组，对于数组中的每一个元素，当 userId 为空时表示本地麦克风采集的音量大小，当 userId 不为空时代表远端用户的音量大小。

onRecvCustomCmdMsg

onRecvCustomCmdMsg

void onRecvCustomCmdMsg	(String userId
	int cmdID
	int seq
	byte[] message)

收到自定义消息的事件回调

当房间中的某个用户使用 sendCustomCmdMsg 发送自定义 UDP 消息时，房间中的其他用户可以通过 onRecvCustomCmdMsg 事件回调接收到该条消息。

参数	描述
cmdID	命令 ID。
message	消息数据。
seq	消息序号。
userId	用户标识。

onMissCustomCmdMsg

onMissCustomCmdMsg

void onMissCustomCmdMsg	(String userId
	int cmdID
	int errCode
	int missed)

自定义消息丢失的事件回调

当您使用 sendCustomCmdMsg 发送自定义 UDP 消息时，即使设置了可靠传输（reliable），也无法确 100% 不丢失，只是丢消息概率极低，能满足常规可靠性要求。
在发送端设置了可靠运输（reliable）后，SDK 都会通过此回调通知过去时间段内（通常为5s）传输途中丢失的自定义消息数量统计信息。

参数	描述
cmdID	命令 ID。
errCode	错误码。
missed	丢失的消息数量。
userId	用户标识。

注意

只有在发送端设置了可靠传输（reliable），接收方才能收到消息的丢失回调。

版权所有：腾讯云计算（北京）有限责任公司

onRecvSEIMsg

onRecvSEIMsg

void onRecvSEIMsg	(String userId
	byte[] data)

收到 SEI 消息的回调

当房间中的某个用户使用 sendSEIMsg 借助视频数据帧发送 SEI 消息时，房间中的其他用户可以通过 onRecvSEIMsg 事件回调接收到该条消息。

参数	描述
message	数据。
userId	用户标识。

onStartPublishing

onStartPublishing

void onStartPublishing	(int err
	String errMsg)

开始向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 startPublishing 开始向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStopPublishing

onStopPublishing

void onStopPublishing	(int err

	String errMsg)

停止向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 stopPublishing 停止向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStartPublishCDNStream

onStartPublishCDNStream

void onStartPublishCDNStream	(int err
	String errMsg)

开始向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 startPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

注意

当您收到成功的事件回调时，只是说明您的发布指令已经同步到腾讯云后台服务器，但如果目标 CDN 厂商的服务器不接收该条视频流，依然可能导致发布失败。

onStopPublishCDNStream

onStopPublishCDNStream

void onStopPublishCDNStream	(int err
	String errMsg)

停止向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 stopPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onSetMixTranscodingConfig

onSetMixTranscodingConfig

void onSetMixTranscodingConfig	(int err
	String errMsg)

设置云端混流的排版布局和转码参数的事件回调

当您调用 setMixTranscodingConfig 调整云端混流的排版布局和转码参数时，SDK 会立刻将这一调整指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	错误码：0表示成功，其余值表示失败。
errMsg	具体的错误原因。

onStartPublishMediaStream

onStartPublishMediaStream

void onStartPublishMediaStream	(String taskId
	int code
	String message
	Bundle extraInfo)

开始发布媒体流的事件回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	当请求成功时，TRTC 后台会在回调中提供给您这项任务的 taskId，后续您可以通过该 taskId 结合 updatePublishMediaStream 和 stopPublishMediaStream 进行更新和停止。

onUpdatePublishMediaStream

onUpdatePublishMediaStream

void onUpdatePublishMediaStream	(String taskId
	int code
	String message
	Bundle extraInfo)

更新媒体流的事件回调

当您调用媒体流发布接口 (updatePublishMediaStream) 开始向 TRTC 后台服务更新媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用媒体流发布接口 (updatePublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次更新请求。

onStopPublishMediaStream

onStopPublishMediaStream

void onStopPublishMediaStream	(String taskId
	int code
	String message
	Bundle extraInfo)

停止媒体流的事件回调

当您调用停止发布媒体流 (stopPublishMediaStream) 开始向 TRTC 后台服务停止媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用停止发布媒体流 (stopPublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次停止请求。

onCdnStreamStateChanged

onCdnStreamStateChanged

void onCdnStreamStateChanged	(String cdnUrl
	int status
	int code
	String msg
	Bundle extraInfo)

RTMP/RTMPS 推流状态发生改变回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务。

若您在目标推流配置 (TRTCPublishTarget) 设置了向腾讯或者第三方 CDN 上发布音视频流的 URL 配置，则具体 RTMP 或者 RTMPS 推流状态将通过此回调同步给您。

参数	描述
cdnUrl	您调用 startPublishMediaStream 时通过目标推流配置 (TRTCPublishTarget) 传入的 url，在推流状态变更时，会通过此回调同步给您。
code	推流结果，0 表示成功，其余值表示出错。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体推流信息。
status	推流状态。 0：推流未开始或者已结束。在您调用 stopPublishMediaStream 时会返回该状态。 1：正在连接 TRTC 服务器和 CDN 服务器。若无法立刻成功，TRTC 后台服务会多次重试尝试推流，并返回该状态（5s回调一次）。如成功进行推流，则进入状态 2；如服务器出错或 60 秒内未成功推流，则进入状态 4。 2：CDN 推流正在进行。在成功推流后，会返回该状态。 3：TRTC 服务器和 CDN 服务器推流中断，正在恢复。当 CDN 出现异常，或推流短暂中断时，TRTC 后台服务会自动尝试恢复推流，并返回该状态（5s回调一次）。如成功恢复推流，则进入状态 2；如服务器出错或 60 秒内未成功恢复，则进入状态 4。 4：TRTC 服务器和 CDN 服务器推流中断，且恢复或连接超时。即此时推流失败，你可以再次调用 updatePublishMediaStream 尝试推流。 5：正在断开 TRTC 服务器和 CDN 服务器。在您调用 stopPublishMediaStream 时，TRTC 后台服务会依次同步状态 5 和状态 0。

onScreenCaptureStarted

onScreenCaptureStarted

屏幕分享开启的事件回调

当您通过 startScreenCapture 等相关接口启动屏幕分享时，SDK 便会抛出此事件回调。

onScreenCapturePaused

onScreenCapturePaused

屏幕分享暂停的事件回调

当您通过 pauseScreenCapture 暂停屏幕分享时，SDK 便会抛出此事件回调。

onScreenCaptureResumed

版权所有：腾讯云计算（北京）有限责任公司

onScreenCaptureResumed

屏幕分享恢复的事件回调

当您通过 resumeScreenCapture 恢复屏幕分享时，SDK 便会抛出此事件回调。

onScreenCaptureStopped

onScreenCaptureStopped

void onScreenCaptureStopped	(int reason)

屏幕分享停止的事件回调

当您通过 stopScreenCapture 停止屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	停止原因，0：用户主动停止；1：屏幕窗口关闭导致停止；2：表示屏幕分享的显示屏状态变更（如接口被拔出、投影模式变更等）。

onLocalRecordBegin

onLocalRecordBegin

void onLocalRecordBegin	(int errCode
	String storagePath)

本地录制任务已经开始的事件回调

当您调用 startLocalRecording 启动本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务是否已经顺利启动。

参数	描述
errCode	状态码。 0：录制任务启动成功。 -1：内部错误导致录制任务启动失败。 -2：文件后缀名有误（比如不支持的录制格式）。 -6：录制已经启动，需要先停止录制。 -7：录制文件已存在，需要先删除文件。 -8：录制目录无写入权限，请检查目录权限问题。

storagePath	录制文件存储路径。

onLocalRecording

onLocalRecording

void onLocalRecording	(long duration
	String storagePath)

本地录制任务正在进行中的进展事件回调

当您调用 startLocalRecording 成功启动本地媒体录制任务后，SDK 会定时地抛出本事件回调。
您可通过捕获该事件回调来获知录制任务的健康状况。
您可以在 startLocalRecording 时设定本事件回调的抛出间隔。

参数	描述
duration	已经录制的累计时长，单位毫秒。
storagePath	录制文件存储路径。

onLocalRecordFragment

onLocalRecordFragment

void onLocalRecordFragment	(String storagePath)

本地录制分片的事件回调

当您开启分片录制时，每完成一个分片，都会通过此接口回调。

参数	描述
storagePath	分片文件存储路径。

onLocalRecordComplete

onLocalRecordComplete

void onLocalRecordComplete	(int errCode

	String storagePath)

本地录制任务已经结束的事件回调

当您调用 stopLocalRecording 停止本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务的最终结果。

参数	描述
errCode	状态码。 0：结束录制任务成功。 -1：录制失败。 -2：切换分辨率或横竖屏导致录制结束。 -3：录制时间太短，或未采集到任何视频或音频数据，请检查录制时长，或是否已开启音、视频采集。
storagePath	录制文件存储路径。

onSnapshotComplete

onSnapshotComplete

void onSnapshotComplete	(Bitmap bmp)

本地截图完成的事件回调

参数	描述
bmp	截图结果，如果 bmp 为 null 代表本次截图操作失败。
data	截图数据，为 nullptr 表示截图失败。
format	截图数据格式，目前只支持 TRTCVideoPixelFormat_BGRA32。
height	截图画面的高度。
length	截图数据长度，对于BGRA32而言，length = width * height * 4。
type	视频流类型。
userId	用户标识，如果 userId 为空字符串，则代表截取的是本地画面。
width	截图画面的宽度。

注意

全平台 C++ 接口和 Java 接口在参数上是不一样的，C++ 接口用 7 个参数描述一个截图画面，Java 接口只用一个 Bitmap 描述一个截图画面。

onUserEnter

onUserEnter

void onUserEnter	(String userId)

有主播加入当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserEnterRoom 替代之。

onUserExit

onUserExit

void onUserExit	(String userId
	int reason)

有主播离开当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserLeaveRoom 替代之。

onAudioEffectFinished

onAudioEffectFinished

void onAudioEffectFinished	(int effectId
	int code)

音效播放已结束（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXAudioEffectManager 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onSpeedTest

onSpeedTest

void onSpeedTest	(TRTCCloudDef.TRTCSpeedTestResult currentResult
	int finishedCount
	int totalCount)

服务器测速的结果回调（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onSpeedTestResult 接口替代之。

TRTCStatistics

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 音视频统计指标（只读）

Function: TRTC SDK 会以两秒钟一次的频率向您汇报当前实时的音视频指标（帧率、码率、卡顿情况等）

TRTCStatistics

结构体类型

函数列表	描述
TRTCLocalStatistics	本地的音视频统计指标
TRTCRemoteStatistics	远端的音视频统计指标
TRTCStatistics	网络和性能的汇总统计指标

TRTCLocalStatistics

TRTCLocalStatistics

本地的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，即每秒钟新产生音频数据的多少，单位 Kbps。
audioCaptureState	【字段含义】音频设备采集状态（用于检测音频外设的健康度）。 0：采集设备状态正常；1：检测到长时间静音；2：检测到破音；3：检测到声音异常间断。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
frameRate	【字段含义】本地视频的帧率，即每秒钟会有多少视频帧，单位：FPS。
height	【字段含义】本地视频的高度，单位 px。

streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
videoBitrate	【字段含义】本地视频的码率，即每秒钟新产生视频数据的多少，单位 Kbps。
width	【字段含义】本地视频的宽度，单位 px。

TRTCRemoteStatistics

TRTCRemoteStatistics

远端的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，单位 Kbps
audioBlockRate	【字段含义】音频播放卡顿率，单位 (%) 音频播放卡顿率（audioBlockRate） = 音频播放的累计卡顿时长（audioTotalBlockTime） / 音频播放的总时长
audioPacketLoss	【字段含义】音频流的总丢包率（%）。 audioPacketLoss 代表音频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 audioPacketLoss 越小越好，丢包率为0即表示该路音频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 audioPacketLoss != 0 的情况，说明该路音频流在“云端=>观众”这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
audioTotalBlockTime	【字段含义】音频播放的累计卡顿时长，单位 ms
finalLoss	【字段含义】该路音视频流的总丢包率（%）。已废弃，不推荐使用；建议使用 audioPacketLoss、videoPacketLoss 替代。
frameRate	【字段含义】远端视频的帧率，单位：FPS。
height	【字段含义】远端视频的高度，单位 px。
jitterBufferDelay	【字段含义】播放延迟，单位 ms 为了避免网络抖动和网络包乱序导致的声音和画面卡顿，TRTC 会在播放端管理一个播放缓冲区，用于对接收到的网络数据包进行整理，该缓冲区的大小会根据当前的网络质量进行自适应调整，该缓冲区的大小折算成以毫秒为单位的时间长度，也就是 jitterBufferDelay。

版权所有：腾讯云计算（北京）有限责任公司

point2PointDelay	【字段含义】端到端延迟，单位 ms point2PointDelay 代表 “主播=>云端=>观众” 的延迟，更准确地说，它代表了“采集=>编码=>网络传输=>接收=>缓冲=>解码=>播放” 全链路的延迟。 point2PointDelay 需要本地和远端的 SDK 均为 8.5 及以上的版本才生效，若远端用户为 8.5 以前的版本，此数值会一直为0，代表无意义。
remoteNetworkRTT	【字段含义】该用户从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。 该数值越小越好：如果 remoteNetworkRTT < 50ms，意味着较低的音视频通话延迟；如果 remoteNetworkRTT > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，remoteNetworkRTT 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 remoteNetworkUpRTT 和 remoteNetworkDownRTT
remoteNetworkUplinkLoss	【字段含义】该用户从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 remoteNetworkUplinkLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 remoteNetworkUplinkLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。
streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
userId	【字段含义】用户 ID
videoBitrate	【字段含义】远端视频的码率，单位 Kbps。
videoBlockRate	【字段含义】视频播放卡顿率，单位 (%) 视频播放卡顿率（videoBlockRate） = 视频播放的累计卡顿时长（videoTotalBlockTime） / 视频播放的总时长。
videoPacketLoss	【字段含义】该路视频流的总丢包率（%）。 videoPacketLoss 代表该路视频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 videoPacketLoss 越小越好，丢包率为0即表示该路视频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 videoPacketLoss != 0 的情况，说明该路视频流在 `云端>观众` 这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
videoTotalBlockTime	【字段含义】视频播放的累计卡顿时长，单位 ms
width	【字段含义】远端视频的宽度，单位 px。

版权所有：腾讯云计算（北京）有限责任公司

TRTCStatistics

TRTCStatistics

网络和性能的汇总统计指标

枚举类型	描述
appCpu	【字段含义】当前应用的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。
downLoss	【字段含义】从云端到 SDK 的下行丢包率，单位 (%) 该数值越小越好，如果 downLoss 为 0%，则意味着下行链路的网络质量很好，从云端接收的数据包基本不发生丢失。 如果 downLoss 为 30%，则意味着云端向 SDK 传输的音视频数据包中，会有 30% 丢失在传输链路中。
gatewayRtt	【字段含义】从 SDK 到本地路由器的往返时延，单位 ms 该数值代表从 SDK 发送一个网络包到本地路由器网关，再从网关回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 【SDK>网关>SDK【 的总耗时。 该数值越小越好：如果 gatewayRtt < 50ms，意味着较低的音视频通话延迟；如果 gatewayRtt > 200ms，则意味着较高的音视频通话延迟。 当网络类型为蜂窝网时，该值无效。
localArray	【字段含义】本地的音视频统计信息 由于本地可能有三路音视频流（即高清大画面，低清小画面，以及辅流画面），因此本地的音视频统计信息是一个数组。
receiveBytes	【字段含义】总接收字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
remoteArray	【字段含义】远端的音视频统计信息 因为同时可能有多个远端用户，而且每个远端用户同时可能有多路音视频流（即高清大画面，低清小画面，以及辅流画面），因此远端的音视频统计信息是一个数组。
rtt	【字段含义】从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。 该数值越小越好：如果 rtt < 50ms，意味着较低的音视频通话延迟；如果 rtt > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，rtt 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 upRtt 和 downRtt。
sendBytes	【字段含义】总发送字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
systemCpu	【字段含义】当前系统的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。

upLoss	【字段含义】从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 upLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 upLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。

版权所有：腾讯云计算（北京）有限责任公司

TXAudioEffectManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 背景音乐、短音效和人声特效的管理类

Function: 用于对背景音乐、短音效和人声特效进行设置的管理类

TXAudioEffectManager

TXMusicPreloadObserver

函数列表	描述
onLoadProgress	背景音乐预加载进度
onLoadError	背景音乐预加载出错

TXMusicPlayObserver

函数列表	描述
onStart	背景音乐开始播放
onPlayProgress	背景音乐的播放进度
onComplete	背景音乐已经播放完毕

TXAudioEffectManager

函数列表	描述
enableVoiceEarMonitor	开启耳返
setVoiceEarMonitorVolume	设置耳返音量

setVoiceReverbType	设置人声的混响效果
setVoiceChangerType	设置人声的变声特效
setVoiceCaptureVolume	设置语音音量
setVoicePitch	设置语音音调
setMusicObserver	设置背景音乐的事件回调接口
startPlayMusic	开始播放背景音乐
stopPlayMusic	停止播放背景音乐
pausePlayMusic	暂停播放背景音乐
resumePlayMusic	恢复播放背景音乐
setAllMusicVolume	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume	设置某一首背景音乐的本地音量的大小
setMusicPitch	调整背景音乐的音调高低
setMusicSpeedRate	调整背景音乐的变速效果
getMusicCurrentPosInMS	获取背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInMS	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate	调整搓碟的变速效果
setPreloadObserver	设置预加载事件回调
preloadMusic	预加载背景音乐
getMusicTrackCount	获取背景音乐的音轨数量
setMusicTrack	指定背景音乐的播放音轨

结构体类型

函数列表	描述

版权所有：腾讯云计算（北京）有限责任公司

AudioMusicParam	背景音乐的播放控制信息

枚举类型

枚举类型	描述
TXVoiceReverbType	混响特效
TXVoiceChangerType	变声特效

onLoadProgress

onLoadProgress

void onLoadProgress	(int id
	int progress)

背景音乐预加载进度

onLoadError

onLoadError

void onLoadError	(int id
	int errorCode)

背景音乐预加载出错

参数	描述
errorCode	-4001:打开文件失败，如音频数据无效，FFMPEG 协议未找到等；-4002:解码失败，如音频文件损坏，网络音频文件服务器无法访问等；-4003:预加载数量超上限，请先调用 stopPlayMusic 释放无用的预加载；-4005:非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件；-4006:非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件；-4007:无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏；-4008:格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，

	aac，m4a，wav，ogg，mp4，mkv】，桌面端支持 【mp3，aac，m4a，wav，mp4，mkv】。

onStart

onStart

void onStart	(int id
	int errCode)

背景音乐开始播放

在背景音乐开始播放成功或者失败时调用。

参数	描述
errCode	错误码。0: 开始播放成功；-4001:打开文件失败，如音频数据无效，FFMPEG 协议未找到等；-4005:非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件；-4006:非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件；-4007:无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏；-4008:格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，aac，m4a，wav，ogg，mp4，mkv】，桌面端支持 【mp3，aac，m4a，wav，mp4，mkv】。
id	音乐 ID。

onPlayProgress

onPlayProgress

void onPlayProgress	(int id
	long curPtsMS
	long durationMS)

背景音乐的播放进度

onComplete

onComplete

void onComplete	(int id
	int errCode)

背景音乐已经播放完毕

在背景音乐播放完毕或播放错误时调用。

参数	描述
errCode	错误码。0: 播放结束；-4002: 解码失败，如音频文件损坏，网络音频文件服务器无法访问等。
id	音乐 ID。

enableVoiceEarMonitor

enableVoiceEarMonitor

void enableVoiceEarMonitor	(boolean enable)

开启耳返

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

需要您注意的是，由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。
同时也需要注意，并非所有的手机开启此特效后都能达到优秀的耳返效果，我们已经对部分耳返效果不佳的手机屏蔽了该特效。

参数	描述
enable	true：开启；false：关闭。

注意

仅在主播佩戴耳机时才能开启此特效，同时请您提示主播佩戴有线耳机。

setVoiceEarMonitorVolume

setVoiceEarMonitorVolume

void setVoiceEarMonitorVolume	(int volume)

设置耳返音量

通过该接口您可以设置耳返特效中声音的音量大小。

参数	描述
volume	音量大小，取值范围为 0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoiceReverbType

setVoiceReverbType

void setVoiceReverbType	(TXVoiceReverbType type)

设置人声的混响效果

通过该接口您可以设置人声的混响效果，具体特效请参见枚举定义 TXVoiceReverbType。

注意

设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceChangerType

setVoiceChangerType

void setVoiceChangerType	(TXVoiceChangerType type)

设置人声的变声特效

通过该接口您可以设置人声的变声特效，具体特效请参见枚举定义 TXVoiceChangeType。

注意

设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceCaptureVolume

setVoiceCaptureVolume

void setVoiceCaptureVolume	(int volume)

设置语音音量

该接口可以设置语音音量的大小，一般配合音乐音量的设置接口 setAllMusicVolume 协同使用，用于调谐语音和音乐在混音前各自的音量占比。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoicePitch

setVoicePitch

void setVoicePitch	(double pitch)

设置语音音调

该接口可以设置语音音调，用于实现变调不变速的目的。

参数	描述
pitch	音调，取值范围为-1.0f~1.0f，默认值：0.0f。

setMusicObserver

setMusicObserver

void setMusicObserver	(int id
	TXMusicPlayObserver observer)

设置背景音乐的事件回调接口

请在播放背景音乐之前使用该接口设置播放事件回调，以便感知背景音乐的播放进度。

参数	描述

musicId	音乐 ID。
observer	具体参考 ITXMusicPlayObserver 中定义接口。

注意

1. 如果该 ID 不需要使用，observer 可设置为 NULL 彻底释放。

startPlayMusic

startPlayMusic

boolean startPlayMusic	(final AudioMusicParam musicParam)

开始播放背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
musicParam	音乐参数。

注意

1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

stopPlayMusic

stopPlayMusic

void stopPlayMusic	(int id)

停止播放背景音乐

参数	描述
id	音乐 ID。

pausePlayMusic

pausePlayMusic

void pausePlayMusic	(int id)

暂停播放背景音乐

参数	描述
id	音乐 ID。

resumePlayMusic

resumePlayMusic

void resumePlayMusic	(int id)

恢复播放背景音乐

参数	描述
id	音乐 ID。

setAllMusicVolume

setAllMusicVolume

void setAllMusicVolume	(int volume)

设置所有背景音乐的本地音量和远端音量的大小

该接口可以设置所有背景音乐的本地音量和远端音量。
本地音量：即主播本地可以听到的背景音乐的音量大小。
远端音量：即观众端可以听到的背景音乐的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPublishVolume

setMusicPublishVolume

void setMusicPublishVolume	(int id
	int volume)

设置某一首背景音乐的远端音量的大小

该接口可以细粒度地控制每一首背景音乐的远端音量，也就是观众端可听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。
volume	音量大小，取值范围为0 - 100；默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPlayoutVolume

setMusicPlayoutVolume

void setMusicPlayoutVolume	(int id
	int volume)

设置某一首背景音乐的本地音量的大小

该接口可以细粒度地控制每一首背景音乐的本地音量，也就是主播本地可以听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。
volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

版权所有：腾讯云计算（北京）有限责任公司

setMusicPitch

setMusicPitch

void setMusicPitch	(int id
	float pitch)

调整背景音乐的音调高低

参数	描述
id	音乐 ID。
pitch	音调，默认值是0.0f，范围是：[-1 ~ 1] 之间的浮点数。

setMusicSpeedRate

setMusicSpeedRate

void setMusicSpeedRate	(int id
	float speedRate)

调整背景音乐的变速效果

参数	描述
id	音乐 ID。
speedRate	速度，默认值是1.0f，范围是：[0.5 ~ 2] 之间的浮点数。

getMusicCurrentPosInMS

getMusicCurrentPosInMS

long getMusicCurrentPosInMS	(int id)

获取背景音乐的播放进度（单位：毫秒）

参数	描述

id	音乐 ID。

返回值说明：

成功返回当前播放时间，单位：毫秒，失败返回 -1。

getMusicDurationInMS

getMusicDurationInMS

long getMusicDurationInMS	(String path)

获取背景音乐的总时长（单位：毫秒）

参数	描述
path	音乐文件路径。

返回值说明：

成功返回时长，失败返回 -1。

seekMusicToPosInMS

seekMusicToPosInMS

void seekMusicToPosInMS	(int id
	int pts)

设置背景音乐的播放进度（单位：毫秒）

参数	描述
id	音乐 ID。
pts	单位: 毫秒。

注意

请尽量避免过度频繁地调用该接口，因为该接口可能会再次读写音乐文件，耗时稍高。

因此，当用户拖拽音乐的播放进度条时，请在用户完成拖拽操作后再调用本接口。

因为 UI 上的进度条控件往往会以很高的频率反馈用户的拖拽进度，如不做频率限制，会导致较差的用户体验。

setMusicScratchSpeedRate

setMusicScratchSpeedRate

void setMusicScratchSpeedRate	(int id
	float scratchSpeedRate)

调整搓碟的变速效果

参数	描述
id	音乐 ID。
scratchSpeedRate	搓碟速度，默认值是1.0f，范围是：[-12.0 ~ 12.0] 之间的浮点数, 速度值正/负表示方向正/反，绝对值大小表示速度快慢。

注意

前置条件 preloadMusic 成功。

setPreloadObserver

setPreloadObserver

void setPreloadObserver	(TXMusicPreloadObserver observer)

设置预加载事件回调

请在预加载背景音乐之前使用该接口设置回调，以便感知背景音乐的预加载进度。

参数	描述
observer	具体参考 ITXMusicPreloadObserver 中定义接口。

preloadMusic

preloadMusic

boolean preloadMusic	(final AudioMusicParam preloadParam)

预加载背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
preloadParam	预加载音乐参数。

注意

1. 预先加载最多同时支持2个不同 ID 的预加载，且预加载时长不超过10分钟，使用完需 stopPlayMusic，否则内存不释放。
2. 若该ID对应的音乐正在播放中，预加载会失败，需先调用 stopPlayMusic。
3. 当 musicParam 和传入 startPlayMusic 的 musicParam 完全相同时，预加载有效。

getMusicTrackCount

getMusicTrackCount

int getMusicTrackCount	(int id)

获取背景音乐的音轨数量

参数	描述
id	音乐 ID。

setMusicTrack

setMusicTrack

void setMusicTrack	(int id
	int trackIndex)

指定背景音乐的播放音轨

参数	描述
id	音乐 ID。
index	默认播放第一个音轨。取值范围[0, 音轨总数)。

注意

音轨总数量可通过 getMusicTrackCount 接口获取。

TXVoiceReverbType

TXVoiceReverbType

混响特效

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受，目前支持如下几种混响效果：
0：关闭；1：KTV；2：小房间；3：大会堂；4：低沉；5：洪亮；6：金属声；7：磁性；8：空灵；9：录音棚；10：悠扬 11：录音棚2。

枚举	取值	描述
TXLiveVoiceReverbType_0	0	关闭特效
TXLiveVoiceReverbType_1	1	KTV
TXLiveVoiceReverbType_2	2	小房间
TXLiveVoiceReverbType_3	3	大会堂
TXLiveVoiceReverbType_4	4	低沉
TXLiveVoiceReverbType_5	5	洪亮
TXLiveVoiceReverbType_6	6	金属声
TXLiveVoiceReverbType_7	7	磁性
TXLiveVoiceReverbType_8	8	空灵
TXLiveVoiceReverbType_9	9	录音棚
TXLiveVoiceReverbType_10	10	悠扬
TXLiveVoiceReverbType_11	11	录音棚2

TXVoiceChangeType

TXVoiceChangeType

变声特效

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色，目前支持如下几种变声特效：
0：关闭；1：熊孩子；2：萝莉；3：大叔；4：重金属；5：感冒；6：外语腔；7：困兽；8：肥宅；9：强电流；10：重机械；11：空灵。

枚举	取值	描述
TXLiveVoiceChangerType_0	0	关闭
TXLiveVoiceChangerType_1	1	熊孩子
TXLiveVoiceChangerType_2	2	萝莉
TXLiveVoiceChangerType_3	3	大叔
TXLiveVoiceChangerType_4	4	重金属
TXLiveVoiceChangerType_5	5	感冒
TXLiveVoiceChangerType_6	6	外语腔
TXLiveVoiceChangerType_7	7	困兽
TXLiveVoiceChangerType_8	8	肥宅
TXLiveVoiceChangerType_9	9	强电流
TXLiveVoiceChangerType_10	10	重机械
TXLiveVoiceChangerType_11	11	空灵

TXAudioMusicParam

TXAudioMusicParam

背景音乐的播放控制信息

该信息用于在接口 startPlayMusic 中指定背景音乐的相关信息，包括播放 ID、文件路径和循环次数等：
1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

枚举类型	描述
endTimeMS	【字段含义】音乐结束播放时间点，单位毫秒，0表示播放至文件结尾。

id	【字段含义】音乐 ID。 【特殊说明】SDK 允许播放多路音乐，因此需要使用 ID 进行标记，用于控制音乐的开始、停止、音量等。
isShortFile	【字段含义】播放的是否为短音乐文件。 【推荐取值】true：需要重复播放的短音乐文件；false：正常的音乐文件。默认值：false。
loopCount	【字段含义】音乐循环播放的次数。 【推荐取值】取值范围为0 - 任意正整数，默认值：0。0 表示播放音乐一次；1 表示播放音乐两次；以此类推。
path	【字段含义】音效文件的完整路径或 URL 地址。支持的音频格式包括 MP3、AAC、M4A、WAV。
publish	【字段含义】是否将音乐传到远端。 【推荐取值】true：音乐在本地播放的同时，远端用户也能听到该音乐；false：主播只能在本地听到该音乐，远端观众听不到。默认值：false。
startTimeMS	【字段含义】音乐开始播放时间点，单位：毫秒。

TXBeautyManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 美颜与图像处理参数设置类

Function: 修改美颜、滤镜、绿幕等参数

TXBeautyManager

TXBeautyManager

函数列表	描述
setBeautyStyle	设置美颜（磨皮）算法
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别
enableSharpnessEnhancement	开启清晰度增强
setRuddyLevel	设置红润级别
setFilter	设置色彩滤镜效果
setFilterStrength	设置色彩滤镜的强度
setGreenScreenFile	设置绿幕背景视频
setEyeScaleLevel	设置大眼级别
setFaceSlimLevel	设置瘦脸级别
setFaceVLevel	设置 V 脸级别
setChinLevel	设置下巴拉伸或收缩
setFaceShortLevel	设置短脸级别
setFaceNarrowLevel	设置窄脸级别

setNoseSlimLevel	设置瘦鼻级别
setEyeLightenLevel	设置亮眼级别
setToothWhitenLevel	设置牙齿美白级别
setWrinkleRemoveLevel	设置祛皱级别
setPounchRemoveLevel	设置祛眼袋级别
setSmileLinesRemoveLevel	设置法令纹去除级别
setForeheadLevel	设置发际线调整级别
setEyeDistanceLevel	设置眼距
setEyeAngleLevel	设置眼角调整级别
setMouthShapeLevel	设置嘴型调整级别
setNoseWingLevel	设置鼻翼调整级别
setNosePositionLevel	设置鼻子位置
setLipsThicknessLevel	设置嘴唇厚度
setFaceBeautyLevel	设置脸型
setMotionTmpl	选择 AI 动效挂件
setMotionMute	是否在动效素材播放时静音

枚举类型

枚举类型	描述
TXBeautyStyle	美颜（磨皮）算法

setBeautyStyle

setBeautyStyle

void setBeautyStyle	(int beautyStyle)

设置美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案：

参数	描述
beautyStyle	美颜风格，TXBeautyStyleSmooth：光滑；TXBeautyStyleNature：自然；TXBeautyStylePitu：优图。

setBeautyLevel

setBeautyLevel

void setBeautyLevel	(float beautyLevel)

设置美颜级别

参数	描述
beautyLevel	美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显。

setWhitenessLevel

setWhitenessLevel

void setWhitenessLevel	(float whitenessLevel)

设置美白级别

参数	描述
whitenessLevel	美白级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

enableSharpnessEnhancement

enableSharpnessEnhancement

void enableSharpnessEnhancement	(boolean enable)

开启清晰度增强

版权所有：腾讯云计算（北京）有限责任公司

setRuddyLevel

setRuddyLevel

void setRuddyLevel	(float ruddyLevel)

设置红润级别

参数	描述
ruddyLevel	红润级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

setFilter

setFilter

void setFilter	(Bitmap image)

设置色彩滤镜效果

色彩滤镜，是一副包含色彩映射关系的颜色查找表图片，您可以在我们提供的官方 Demo 中找到预先准备好的几张滤镜图片。
SDK 会根据该查找表中的映射关系，对摄像头采集出的原始视频画面进行二次处理，以达到预期的滤镜效果。

参数	描述
image	包含色彩映射关系的颜色查找表图片，必须是 png 格式。

setFilterStrength

setFilterStrength

void setFilterStrength	(float strength)

设置色彩滤镜的强度

该数值越高，色彩滤镜的作用强度越明显，经过滤镜处理后的视频画面跟原画面的颜色差异越大。
我默认的滤镜浓度是 0.5，如果您觉得默认的滤镜效果不明显，可以设置为 0.5 以上的数字，最大值为 1。

参数	描述
strength	从 0 到 1，数值越大滤镜效果越明显，默认值为 0.5。

版权所有：腾讯云计算（北京）有限责任公司

setGreenScreenFile

setGreenScreenFile

int setGreenScreenFile	(String path)

设置绿幕背景视频

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。此接口所开启的绿幕功能不具备智能去除背景的能力，需要被拍摄者的背后有一块绿色的幕布来辅助产生特效。

参数	描述
path	MP4格式的视频文件路径; 设置空值表示关闭特效。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeScaleLevel

setEyeScaleLevel

int setEyeScaleLevel	(float eyeScaleLevel)

设置大眼级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeScaleLevel	大眼级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceSlimLevel

setFaceSlimLevel

int setFaceSlimLevel	(float faceSlimLevel)

设置瘦脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceSlimLevel	瘦脸级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceVLevel

setFaceVLevel

int setFaceVLevel	(float faceVLevel)

设置 V 脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceVLevel	V 脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setChinLevel

setChinLevel

int setChinLevel	(float chinLevel)

设置下巴拉伸或收缩

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
chinLevel	下巴拉伸或收缩级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示收缩，大于 0 表示拉伸。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceShortLevel

setFaceShortLevel

int setFaceShortLevel	(float faceShortLevel)

设置短脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceShortLevel	短脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceNarrowLevel

setFaceNarrowLevel

int setFaceNarrowLevel	(float faceNarrowLevel)

设置窄脸级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
level	窄脸级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNoseSlimLevel

setNoseSlimLevel

int setNoseSlimLevel	(float noseSlimLevel)

设置瘦鼻级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
noseSlimLevel	瘦鼻级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeLightenLevel

setEyeLightenLevel

int setEyeLightenLevel	(float eyeLightenLevel)

设置亮眼级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeLightenLevel	亮眼级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setToothWhitenLevel

setToothWhitenLevel

int setToothWhitenLevel	(float toothWhitenLevel)

设置牙齿美白级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
toothWhitenLevel	白牙级别，取值范围 0 - 9；0表示关闭，9 表示效果最明显。

版权所有：腾讯云计算（北京）有限责任公司

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setWrinkleRemoveLevel

setWrinkleRemoveLevel

int setWrinkleRemoveLevel	(float wrinkleRemoveLevel)

设置祛皱级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
wrinkleRemoveLevel	祛皱级别，取值范围0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setPounchRemoveLevel

setPounchRemoveLevel

int setPounchRemoveLevel	(float pounchRemoveLevel)

设置祛眼袋级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
pounchRemoveLevel	祛眼袋级别，取值范围 0 - 9；0 表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setSmileLinesRemoveLevel

setSmileLinesRemoveLevel

int setSmileLinesRemoveLevel	(float smileLinesRemoveLevel)

设置法令纹去除级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
smileLinesRemoveLevel	法令纹级别，取值范围 0 - 9；0表示关闭，9 表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setForeheadLevel

setForeheadLevel

int setForeheadLevel	(float foreheadLevel)

设置发际线调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
foreheadLevel	发际线级别，取值范围-9 - 9；0表示关闭，9表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeDistanceLevel

setEyeDistanceLevel

int setEyeDistanceLevel	(float eyeDistanceLevel)

设置眼距

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述

eyeDistanceLevel	眼距级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setEyeAngleLevel

setEyeAngleLevel

int setEyeAngleLevel	(float eyeAngleLevel)

设置眼角调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
eyeAngleLevel	眼角调整级别，取值范围-9 - 9；0表示关闭，9表示效果最明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setMouthShapeLevel

setMouthShapeLevel

int setMouthShapeLevel	(float mouthShapeLevel)

设置嘴型调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
mouthShapeLevel	嘴型级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNoseWingLevel

setNoseWingLevel

int setNoseWingLevel	(float noseWingLevel)

设置鼻翼调整级别

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
noseWingLevel	鼻翼调整级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setNosePositionLevel

setNosePositionLevel

int setNosePositionLevel	(float nosePositionLevel)

设置鼻子位置

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
nosePositionLevel	鼻子位置级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示抬高，大于 0 表示降低。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setLipsThicknessLevel

setLipsThicknessLevel

int setLipsThicknessLevel	(float lipsThicknessLevel)

设置嘴唇厚度

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
lipsThicknessLevel	嘴唇厚度级别，取值范围 -9 - 9；0 表示关闭，小于 0 表示拉伸，大于 0 表示收缩。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setFaceBeautyLevel

setFaceBeautyLevel

int setFaceBeautyLevel	(float faceBeautyLevel)

设置脸型

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
faceBeautyLevel	美型级别，取值范围 0 - 9；0 表示关闭，1 - 9 值越大，效果越明显。

返回值说明：

0：成功；-5：当前 License 对应 feature 不支持。

setMotionTmpl

setMotionTmpl

void setMotionTmpl	(String tmplPath)

选择 AI 动效挂件

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。

参数	描述
tmplPath	动效素材文件所在目录。

setMotionMute

setMotionMute

void setMotionMute	(boolean motionMute)

是否在动效素材播放时静音

该接口仅在 企业版 SDK（旧版已下线，新版本SDK如需使用高级美颜功能请参见腾讯美颜特效SDK）中生效。
有些挂件本身会有声音特效，通过此 API 可以关闭这些特效播放时所带的声音效果。

参数	描述
motionMute	true：静音；false：不静音。

TXBeautyStyle

TXBeautyStyle

美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案。

枚举	取值	描述
TXBeautyStyleSmooth	0	光滑，算法比较激进，磨皮效果比较明显，适用于秀场直播。
TXBeautyStyleNature	1	自然，算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。
TXBeautyStylePitu	2	优图，由优图实验室提供，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高。

TXDeviceManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 音视频设备管理模块

Function: 用于管理摄像头、麦克风和扬声器等音视频相关的硬件设备

TXDeviceManager

TXDeviceManager

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera	切换前置或后置摄像头（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition	设置摄像头的对焦位置（仅适用于移动端）
enableCameraTorch	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）
setAudioRoute	设置音频路由（仅适用于移动端）
setExposureCompensation	设置摄像头的曝光参数，取值范围从-1到1
setCameraCapturerParam	设置摄像头采集偏好
setSystemVolumeType	设置系统音量类型（仅适用于移动端）

结构体类型

函数列表	描述
TXCameraCaptureParam	摄像头采集参数

枚举类型

枚举类型	描述
TXSystemVolumeType	系统音量类型（仅适用于移动设备）
TXAudioRoute	音频路由（即声音的播放模式）
TXCameraCaptureMode	摄像头采集偏好

isFrontCamera

isFrontCamera

判断当前是否为前置摄像头（仅适用于移动端）

switchCamera

switchCamera

int switchCamera	(boolean frontCamera)

切换前置或后置摄像头（仅适用于移动端）

getCameraZoomMaxRatio

getCameraZoomMaxRatio

获取摄像头的最大缩放倍数（仅适用于移动端）

setCameraZoomRatio

setCameraZoomRatio

int setCameraZoomRatio	(float zoomRatio)

设置摄像头的缩放倍数（仅适用于移动端）

参数	描述
zoomRatio	取值范围1 - 5，取值为1表示最远视角（正常镜头），取值为5表示最近视角（放大镜头）。最大值推荐为5，若超过5，视频数据会变得模糊不清。

isAutoFocusEnabled

isAutoFocusEnabled

查询是否支持自动识别人脸位置（仅适用于移动端）

enableCameraAutoFocus

enableCameraAutoFocus

int enableCameraAutoFocus	(boolean enabled)

开启自动对焦功能（仅适用于移动端）

开启后，SDK 会自动检测画面中的人脸位置，并将摄像头的焦点始终对焦在人脸位置上。

setCameraFocusPosition

setCameraFocusPosition

int setCameraFocusPosition	(int x
	int y)

设置摄像头的对焦位置（仅适用于移动端）

您可以通过该接口实现如下交互：

1. 在本地摄像头的预览画面上，允许用户单击操作。
2. 在用户的单击位置显示一个矩形方框，以示摄像头会在此处对焦。
3. 随后将用户点击位置的坐标通过本接口传递给 SDK，之后 SDK 会操控摄像头按照用户期望的位置进行对焦。

参数	描述

position	对焦位置，请传入期望对焦点的坐标值

注意

使用该接口的前提是先通过 enableCameraAutoFocus 关闭自动对焦功能。

返回值说明：

0：操作成功；负数：操作失败。

enableCameraTorch

enableCameraTorch

boolean enableCameraTorch	(boolean enable)

开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）

setAudioRoute

setAudioRoute

int setAudioRoute	(TXAudioRoute route)

设置音频路由（仅适用于移动端）

手机有两个音频播放设备：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

setExposureCompensation

setExposureCompensation

int setExposureCompensation	(float value)

设置摄像头的曝光参数，取值范围从-1到1

setCameraCapturerParam

setCameraCapturerParam

void setCameraCapturerParam	(TXCameraCaptureParam params)

设置摄像头采集偏好

setSystemVolumeType

setSystemVolumeType

int setSystemVolumeType	(TXSystemVolumeType type)

设置系统音量类型（仅适用于移动端）

@deprecated v9.5 版本开始不推荐使用，建议使用 `TRTCCloud` 中的 startLocalAudio(quality) 接口替代之，通过 quality 参数来决策音质。

TXSystemVolumeType(Deprecated)

TXSystemVolumeType(Deprecated)

系统音量类型（仅适用于移动设备）

@deprecated v9.5 版本开始不推荐使用。

现代智能手机中一般都具备两套系统音量类型，即“通话音量”和“媒体音量”。
通话音量：手机专门为接打电话所设计的音量类型，自带回声抵消（AEC）功能，并且支持通过蓝牙耳机上的麦克风进行拾音，缺点是音质比较一般。
当您通过手机侧面的音量按键下调手机音量时，如果无法将其调至零（也就是无法彻底静音），说明您的手机当前处于通话音量。
媒体音量：手机专门为音乐场景所设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。
当您通过手机侧面的音量按键下调手机音量时，如果能够将手机音量调至彻底静音，说明您的手机当前处于媒体音量。

SDK 目前提供了三种系统音量类型的控制模式：自动切换模式、全程通话音量模式、全程媒体音量模式。

枚举	取值	描述
TXSystemVolumeTypeAuto	Not Defined	自动切换模式

TXSystemVolumeTypeMedia	Not Defined	全程媒体音量
TXSystemVolumeTypeVOIP	Not Defined	全程通话音量

TXAudioRoute

TXAudioRoute

音频路由（即声音的播放模式）

音频路由，即声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。
手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

枚举	取值	描述
TXAudioRouteSpeakerphone	Not Defined	Speakerphone：使用扬声器播放（即“免提”），扬声器位于手机底部，声音偏大，适合外放音乐。
TXAudioRouteEarpiece	Not Defined	Earpiece：使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

TXCameraCaptureMode

TXCameraCaptureMode

摄像头采集偏好

该枚举类型用于摄像头采集参数设置。

枚举	取值	描述
TXCameraResolutionStrategyAuto	Not Defined	自动调整采集参数。 SDK 根据实际的采集设备性能及网络情况，选择合适的摄像头输出参数，在设备性能及视频预览质量之间，维持平衡。
TXCameraResolutionStrategyPerformance	Not Defined	优先保证设备性能。 SDK 根据用户设置编码器的分辨率和帧率，选择最接近的摄像头输出参数，从而

		保证设备性能。
TXCameraResolutionStrategyHighQuality	Not Defined	优先保证视频预览质量。 SDK选择较高的摄像头输出参数，从而提高预览视频的质量。在这种情况下，会消耗更多的 CPU 及内存做视频前处理。
TXCameraCaptureManual	Not Defined	允许用户设置本地摄像头采集的视频宽高。

TXCameraCaptureParam

TXCameraCaptureParam

摄像头采集参数

该设置能决定本地预览图像画质。

枚举类型	描述
height	**字段含义：** 采集图像宽度
mode	**字段含义：** 摄像头采集偏好，请参见 TXCameraCaptureMode
width	**字段含义：** 采集图像长度

类型定义

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 关键类型定义

Function: 分辨率、质量等级等枚举和常量值的定义

类型定义

结构体类型

函数列表	描述
TRTCParams	进房参数
TRTCVideoEncParam	视频编码参数
TRTCNetworkQosParam	网络流控（Qos）参数集
TRTCRenderParams	视频画面的渲染参数
TRTCQuality	网络质量
TRTCVolumeInfo	音量大小
TRTCSpeedTestParams	测速参数
TRTCSpeedTestResult	网络测速结果
TRTCTexture	视频纹理数据
TRTCVideoFrame	视频帧信息
TRTCAudioFrame	音频帧数据
TRTCMixUser	云端混流中各路画面的描述信息
TRTCTranscodingConfig	云端混流的排版布局和转码参数
TRTCPublishCDNParam	向非腾讯云 CDN 上发布音视频流时需设置的转推参数

TRTCAudioRecordingParams	本地音频文件的录制参数
TRTCLocalRecordingParams	本地媒体文件的录制参数
TRTCAudioEffectParam	音效参数（已废弃）
TRTCSwitchRoomConfig	房间切换参数
TRTCAudioFrameCallbackFormat	音频自定义回调的格式参数
TRTCScreenShareParams	屏幕分享参数（仅适用于 Android 平台）
TRTCAudioParallelParams	远端音频流智能并发播放策略的参数
TRTCUser	媒体流发布相关配置的用户信息
TRTCPublishCdnUrl	向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置
TRTCPublishTarget	目标推流配置
TRTCVideoLayout	转码视频布局
TRTCWatermark	水印布局
TRTCStreamEncoderParam	媒体流编码输出参数
TRTCStreamMixingConfig	媒体流转码配置参数
TRTCPayloadPrivateEncryptionConfig	媒体流私有加密配置
TRTCAudioVolumeEvaluateParams	音量评估等相关参数设置

枚举类型

枚举类型	描述
TRTCVideoResolution	视频分辨率
TRTCVideoResolutionMode	视频宽高比模式
TRTCVideoStreamType	视频流类型
TRTCVideoFillMode	视频画面填充模式
TRTCVideoRotation	视频画面旋转方向
TRTCBeautyStyle	美颜（磨皮）算法

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoPixelFormat	视频像素格式
TRTCVideoBufferType	视频数据传递方式
TRTCVideoMirrorType	视频的镜像类型
TRTCSnapshotSourceType	本地视频截图的数据源
TRTCAppScene	应用场景
TRTCRoleType	角色
TRTCQosControlMode(Deprecated)	流控模式（已废弃）
TRTCVideoQosPreference	画质偏好
TRTCQuality	网络质量
TRTCAVStatusType	音视频状态类型
TRTCAVStatusChangeReason	音视频状态变化原因类型
TRTCAudioSampleRate	音频采样率
TRTCAudioQuality	声音音质
TRTCAudioRoute	音频路由（即声音的播放模式）
TRTCReverbType	声音混响模式
TRTCVoiceChangerType	变声类型
TRTCSystemVolumeType	系统音量类型（仅适用于移动设备）
TRTCAudioFrameFormat	音频帧的内容格式
TRTCAudioCapabilityType	系统支持的音频能力类型（仅适用于Android设备）
TRTCAudioFrameOperationMode	音频回调数据读写模式
TRTCLogLevel	Log 级别
TRTCGSensorMode	重力感应开关（仅适用于移动端）
TRTCTranscodingConfigMode	云端混流的排版模式
TRTCRecordType	媒体录制类型
TRTCMixInputType	混流输入类型

版权所有：腾讯云计算（北京）有限责任公司

TRTCDebugViewLevel	用于渲染控件上展示的调试信息
TRTCAudioRecordingContent	音频录制内容类型
TRTCPublishMode	媒体流发布模式
TRTCEncryptionAlgorithm	加密算法
TRTCSpeedTestScene	测速场景
TRTCGravitySensorAdaptiveMode	设置重力感应的适配模式（仅适用于移动端）

TRTCVideoResolution

TRTCVideoResolution

视频分辨率

此处仅定义横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTC_VIDEO_RESOLUTION_120_120	1	宽高比 1:1；分辨率 120x120；建议码率（VideoCall）80kbps; 建议码率（LIVE）120kbps。
TRTC_VIDEO_RESOLUTION_160_160	3	宽高比 1:1 分辨率 160x160；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTC_VIDEO_RESOLUTION_270_270	5	宽高比 1:1；分辨率 270x270；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTC_VIDEO_RESOLUTION_480_480	7	宽高比 1:1；分辨率 480x480；建议码率（VideoCall）350kbps; 建议码率（LIVE）500kbps。
TRTC_VIDEO_RESOLUTION_160_120	50	宽高比4:3；分辨率 160x120；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTC_VIDEO_RESOLUTION_240_180	52	宽高比 4:3；分辨率 240x180；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。

TRTC_VIDEO_RESOLUTION_280_210	54	宽高比 4:3；分辨率 280x210；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTC_VIDEO_RESOLUTION_320_240	56	宽高比 4:3；分辨率 320x240；建议码率（VideoCall）250kbps; 建议码率（LIVE）375kbps。
TRTC_VIDEO_RESOLUTION_400_300	58	宽高比 4:3；分辨率 400x300；建议码率（VideoCall）300kbps; 建议码率（LIVE）450kbps。
TRTC_VIDEO_RESOLUTION_480_360	60	宽高比 4:3；分辨率 480x360；建议码率（VideoCall）400kbps; 建议码率（LIVE）600kbps。
TRTC_VIDEO_RESOLUTION_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps; 建议码率（LIVE）900kbps。
TRTC_VIDEO_RESOLUTION_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps; 建议码率（LIVE）1500kbps。
TRTC_VIDEO_RESOLUTION_160_90	100	宽高比 16:9；分辨率 160x90；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。
TRTC_VIDEO_RESOLUTION_256_144	102	宽高比 16:9；分辨率 256x144；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTC_VIDEO_RESOLUTION_320_180	104	宽高比 16:9；分辨率 320x180；建议码率（VideoCall）250kbps; 建议码率（LIVE）400kbps。
TRTC_VIDEO_RESOLUTION_480_270	106	宽高比 16:9；分辨率 480x270；建议码率（VideoCall）350kbps; 建议码率（LIVE）550kbps。
TRTC_VIDEO_RESOLUTION_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps; 建议码率（LIVE）900kbps。
TRTC_VIDEO_RESOLUTION_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps; 建议码率（LIVE）1300kbps。

版权所有：腾讯云计算（北京）有限责任公司

TRTC_VIDEO_RESOLUTION_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps; 建议码率（LIVE）1800kbps。
TRTC_VIDEO_RESOLUTION_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps; 建议码率（LIVE）3000kbps。

TRTCVideoResolutionMode

TRTCVideoResolutionMode

视频宽高比模式

TRTCVideoResolution 中仅定义了横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTC_VIDEO_RESOLUTION_MODE_LANDSCAPE	0	横屏分辨率，例如：TRTCVideoResolution_640_360 + TRTCVideoResolutionModeLandscape = 640 × 360。
TRTC_VIDEO_RESOLUTION_MODE_PORTRAIT	1	竖屏分辨率，例如：TRTCVideoResolution_640_360 + TRTCVideoResolutionModePortrait = 360 × 640。

TRTCVideoStreamType

TRTCVideoStreamType

视频流类型

TRTC 内部有三种不同的视频流，分别是：
高清大画面：一般用来传输摄像头的视频数据。
低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

注意

不支持单独开启低清小画面，小画面必须依附于大画面而存在，SDK 会自动设定低清小画面的分辨率和码率。

枚举	取值	描述
TRTC_VIDEO_STREAM_TYPE_BIG	0	高清大画面，一般用来传输摄像头的视频数据。
TRTC_VIDEO_STREAM_TYPE_SMALL	1	低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
TRTC_VIDEO_STREAM_TYPE_SUB	2	辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

TRTCVideoFillMode

TRTCVideoFillMode

视频画面填充模式

如果视频显示区域的宽高比不等于视频内容的宽高比时，需要指定画面的填充模式。

枚举	取值	描述
TRTC_VIDEO_RENDER_MODE_FILL	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完整。
TRTC_VIDEO_RENDER_MODE_FIT	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边。

TRTCVideoRotation

TRTCVideoRotation

视频画面旋转方向

TRTC 提供了对本地和远程画面的旋转角度设置 API，下列的旋转角度都是指顺时针方向的。

枚举	取值	描述
TRTC_VIDEO_ROTATION_0	0	不旋转。
TRTC_VIDEO_ROTATION_90	1	顺时针旋转90度。

TRTC_VIDEO_ROTATION_180	2	顺时针旋转180度。
TRTC_VIDEO_ROTATION_270	3	顺时针旋转270度。

TRTCBeautyStyle

TRTCBeautyStyle

美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案。

枚举	取值	描述
TRTC_BEAUTY_STYLE_SMOOTH	0	光滑，算法比较激进，磨皮效果比较明显，适用于秀场直播。
TRTC_BEAUTY_STYLE_NATURE	1	自然，算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。
TRTC_BEAUTY_STYLE_PITU	2	优图，由优图实验室提供，磨皮效果介于光滑和自然之间，比光滑保留更多皮肤细节，比自然磨皮程度更高。

TRTCVideoPixelFormat

TRTCVideoPixelFormat

视频像素格式

TRTC 提供针对视频的自定义采集和自定义渲染功能：
在自定义采集功能中，您可以用下列枚举值描述您采集的视频像素格式。
在自定义渲染功能中，您可以指定您期望 SDK 回调出的视频像素格式。

枚举	取值	描述
TRTC_VIDEO_PIXEL_FORMAT_UNKNOWN	0	未定义的格式。
TRTC_VIDEO_PIXEL_FORMAT_I420	1	YUV420P(I420) 格式。
TRTC_VIDEO_PIXEL_FORMAT_Texture_2D	2	OpenGL 2D 纹理格式。
TRTC_VIDEO_PIXEL_FORMAT_TEXTURE_EXTERNAL_OES	3	OES 外部纹理格式（适用于 Android 平台）。
TRTC_VIDEO_PIXEL_FORMAT_NV21	4	NV21 格式。

版权所有：腾讯云计算（北京）有限责任公司

TRTC_VIDEO_PIXEL_FORMAT_RGBA	5	RGBA 格式。

TRTCVideoBufferType

TRTCVideoBufferType

视频数据传递方式

在自定义采集和自定义渲染功能，您需要用到下列枚举值来指定您希望以什么方式传递视频数据：
方案一：使用内存 Buffer 传递视频数据，该方案在 iOS 效率尚可，但在 Android 系统上效率较差，Windows 暂时仅支持内存 Buffer 的传递方式。
方案二：使用 Texture 纹理传递视频数据，该方案在 iOS 和 Android 系统下均有较高的效率，Windows 暂不支持，需要您有一定的 OpenGL 编程基础。

枚举	取值	描述
TRTC_VIDEO_BUFFER_TYPE_UNKNOWN	0	未定义的传递方式。
TRTC_VIDEO_BUFFER_TYPE_BYTE_BUFFER	1	使用内存 Buffer 传递视频数据，iOS：PixelBuffer；Android：用于 JNI 层的 Direct Buffer；Win：内存数据块。
TRTC_VIDEO_BUFFER_TYPE_BYTE_ARRAY	2	使用内存 Buffer 传递视频数据，iOS：经过一次额外整理后更加紧凑的 NSData 类型的内存块；Android：用于 JAVA 层的 byte[]。 该传递的方式的性能是几种方案中效率较差的一种。
TRTC_VIDEO_BUFFER_TYPE_TEXTURE	3	使用 OpengGL 纹理传递视频数据。

TRTCVideoMirrorType

TRTCVideoMirrorType

视频的镜像类型

视频的镜像是指对视频内容进行左右翻转，尤其是对本地的摄像头预览视频，开启镜像后能给主播带来熟悉的“照镜子”体验。

枚举	取值	描述
TRTC_VIDEO_MIRROR_TYPE_AUTO	0	自动模式：如果正使用前置摄像头则开启镜像，如果是后置摄像头则不开启镜像（仅适用于移动设

		备）。
TRTC_VIDEO_MIRROR_TYPE_ENABLE	1	强制开启镜像，不论当前使用的是前置摄像头还是后置摄像头。
TRTC_VIDEO_MIRROR_TYPE_DISABLE	2	强制关闭镜像，不论当前使用的是前置摄像头还是后置摄像头。

TRTCSnapshotSourceType

TRTCSnapshotSourceType

本地视频截图的数据源

SDK 支持从如下两种数据源中截取图片并保存成本地文件：
视频流：从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
渲染层：从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
采集层：从采集控件中截取采集到的视频内容，可以截取采集出来的高清截图。

枚举	取值	描述
TRTC_SNAPSHOT_SOURCE_TYPE_STREAM	0	从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
TRTC_SNAPSHOT_SOURCE_TYPE_VIEW	1	从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
TRTC_SNAPSHOT_SOURCE_TYPE_CAPTURE	2	从采集控件中截取采集到的视频内容，可以截取采集出来的高清截图。

TRTCAppScene

TRTCAppScene

应用场景

TRTC 针对常见的音视频应用场景都进行了定向优化，以满足各种垂直场景下的差异化要求，主要场景可以分为如下两类：
直播（LIVE）场景：包括 LIVE 和 VoiceChatRoom，前者是音频+视频，后者是纯音频。

直播场景下，用户被分成“主播”和“观众”两种角色，单个房间中同时最多支持10万人在线，适合于观众人数众多的直播场景。

实时（RTC）场景：包括 VideoCall 和 AudioCall，前者是音频+视频，后者是纯音频。

实时场景下，用户没有角色的差异，但单个房间中同时最多支持 300 人在线，适合于小范围实时通信的场景。

枚举	取值	描述
TRTC_APP_SCENE_VIDEOCALL	0	视频通话场景，支持720P、1080P高清画质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于[1对1视频通话]、[300人视频会议]、[在线问诊]、[教育小班课]、[远程面试]等业务场景。
TRTC_APP_SCENE_LIVE	1	视频互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于300ms；支持十万级别观众同时播放，播放延时低至1000ms。 适用于[低延时互动直播]、[大班课]、[主播PK]、[视频相亲]、[在线互动课堂]、[远程培训]、[超大型会议]等业务场景。 **注意** 此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。
TRTC_APP_SCENE_AUDIOCALL	2	语音通话场景，默认采用 SPEECH 音质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于 [1对1语音通话]、[300人语音会议]、[语音聊天]、[语音会议]、[在线狼人杀] 等业务场景。
TRTC_APP_SCENE_VOICE_CHATROOM	3	语音互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于 300ms；支持十万级别观众同时播放，播放延时低至 1000ms。 适用于 [语音俱乐部]、[在线K歌房]、[音乐直播间]、[FM电台] 等业务场景。 **注意** 此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。

TRTCRoleType

TRTCRoleType

角色

仅适用于直播类场景（即 TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM），把用户区分成两种不同的身份：

主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。

观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

枚举	取值	描述
TRTCRoleAnchor	20	主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。
TRTCRoleAudience	21	观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

TRTCQosControlMode(Deprecated)

TRTCQosControlMode(Deprecated)

流控模式（已废弃）

枚举	取值	描述
VIDEO_QOS_CONTROL_CLIENT	0	本地控制，用于 SDK 开发内部调试，客户请勿使用。
VIDEO_QOS_CONTROL_SERVER	1	云端控制，默认模式，推荐选择。

TRTCVideoQosPreference

TRTCVideoQosPreference

画质偏好

TRTC 在弱网络环境下有两种调控模式：“优先保证画面清晰”或“优先保证画面流畅”，两种模式均会优先保障声音数据的传输。

枚举	取值	描述
TRTC_VIDEO_QOS_PREFERENCE_SMOOTH	1	流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。
TRTC_VIDEO_QOS_PREFERENCE_CLEAR	2	清晰优先（默认值）：即当前网络不足以传输

		既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。

TRTCQuality

TRTCQuality

网络质量

TRTC 会每隔两秒对当前的网络质量进行评估，评估结果为六个等级：Excellent 表示最好，Down 表示最差。

枚举	取值	描述
TRTC_QUALITY_UNKNOWN	0	未定义。
TRTC_QUALITY_Excellent	1	当前网络非常好。
TRTC_QUALITY_Good	2	当前网络比较好。
TRTC_QUALITY_Poor	3	当前网络一般。
TRTC_QUALITY_Bad	4	当前网络较差。
TRTC_QUALITY_Vbad	5	当前网络很差。
TRTC_QUALITY_Down	6	当前网络不满足 TRTC 的最低要求。

TRTCAVStatusType

TRTCAVStatusType

音视频状态类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态。

枚举	取值	描述
TRTCAVStatusStopped	0	停止播放。
TRTCAVStatusPlaying	1	正在播放。
TRTCAVStatusLoading	2	正在加载。

TRTCAVStatusChangeReason

TRTCAVStatusChangeReason

音视频状态变化原因类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态变化原因。

枚举	取值	描述
TRTCAVStatusChangeReasonInternal	0	缺省值。
TRTCAVStatusChangeReasonBufferingBegin	1	网络缓冲。
TRTCAVStatusChangeReasonBufferingEnd	2	结束缓冲。
TRTCAVStatusChangeReasonLocalStarted	3	本地启动音频或视频流播放。
TRTCAVStatusChangeReasonLocalStopped	4	本地停止音频或视频流播放。
TRTCAVStatusChangeReasonRemoteStarted	5	远端音频或视频流开始（或继续）。
TRTCAVStatusChangeReasonRemoteStopped	6	远端音频或视频流停止（或中断）。

TRTCAudioSampleRate

TRTCAudioSampleRate

音频采样率

音频采样率用来衡量声音的保真程度，采样率越高保真程度越好，如果您的应用场景有音乐的存在，推荐使用 TRTCAudioSampleRate48000。

枚举	取值	描述
TRTCAudioSampleRate16000	16000	16k 采样率。
TRTCAudioSampleRate32000	32000	32k 采样率。
TRTCAudioSampleRate44100	44100	44.1k 采样率。
TRTCAudioSampleRate48000	48000	48k采样率

版权所有：腾讯云计算（北京）有限责任公司

TRTCAudioQuality

TRTCAudioQuality

声音音质

TRTC 提供了三种精心校调好的模式，用来满足各种垂直场景下对音质的差异化追求：
人声模式（Speech）：适用于以人声沟通为主的应用场景，该模式下音频传输的抗性较强，TRTC 会通过各种人声处理技术保障在弱网络环境下的流畅度最佳。
音乐模式（Music）：适用于对声乐要求很苛刻的场景，该模式下音频传输的数据量很大，TRTC 会通过各项技术确保音乐信号在各频段均能获得高保真的细节还原度。
默认模式（Default）：介于 Speech 和 Music 之间的档位，对音乐的还原度比人声模式要好，但传输数据量比音乐模式要低很多，对各种场景均有不错的适应性。

枚举	取值	描述
TRTC_AUDIO_QUALITY_SPEECH	1	人声模式：采样率：16k；单声道；编码码率：16kbps；具备几个模式中最强的网络抗性，适合语音通话为主的场景，比如在线会议，语音通话等。
TRTC_AUDIO_QUALITY_DEFAULT	2	默认模式：采样率：48k；单声道；编码码率：50kbps；介于 Speech 和 Music 之间的档位，SDK 默认档位，推荐选择。
TRTC_AUDIO_QUALITY_MUSIC	3	音乐模式：采样率：48k；全频带立体声；编码码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

TRTCAudioRoute

TRTCAudioRoute

音频路由（即声音的播放模式）

音频路由，即声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。
手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。
音频路由为有线耳机。
音频路由为蓝牙耳机。
音频路由为USB专业声卡设备。

枚举	取值	描述

TRTC_AUDIO_ROUTE_SPEAKER	0	Speakerphone：使用扬声器播放（即“免提”），扬声器位于手机底部，声音偏大，适合外放音乐。
TRTC_AUDIO_ROUTE_EARPIECE	1	Earpiece：使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。
TRTC_AUDIO_ROUTE_WIRED_HEADSET	2	WiredHeadset：使用有线耳机播放。
TRTC_AUDIO_ROUTE_BLUETOOTH_HEADSET	3	BluetoothHeadset：使用蓝牙耳机播放。
TRTC_AUDIO_ROUTE_SOUND_CARD	4	SoundCard：使用 USB 声卡播放。

TRTCReverbType

TRTCReverbType

声音混响模式

该枚举值应用于设定直播场景中的混响模式，常用于秀场直播中。

枚举	取值	描述
TRTC_REVERB_TYPE_0	0	关闭混响。
TRTC_REVERB_TYPE_1	1	KTV。
TRTC_REVERB_TYPE_2	2	小房间。
TRTC_REVERB_TYPE_3	3	大会堂。
TRTC_REVERB_TYPE_4	4	低沉。
TRTC_REVERB_TYPE_5	5	洪亮。
TRTC_REVERB_TYPE_6	6	金属声。
TRTC_REVERB_TYPE_7	7	磁性。

TRTCVoiceChangerType

TRTCVoiceChangerType

变声类型

该枚举值应用于设定直播场景中的变声模式，常用于秀场直播中。

枚举	取值	描述
TRTC_VOICE_CHANGER_TYPE_0	0	关闭变声。
TRTC_VOICE_CHANGER_TYPE_1	1	熊孩子。
TRTC_VOICE_CHANGER_TYPE_2	2	萝莉。
TRTC_VOICE_CHANGER_TYPE_3	3	大叔。
TRTC_VOICE_CHANGER_TYPE_4	4	重金属。
TRTC_VOICE_CHANGER_TYPE_5	5	感冒。
TRTC_VOICE_CHANGER_TYPE_6	6	外国人。
TRTC_VOICE_CHANGER_TYPE_7	7	困兽。
TRTC_VOICE_CHANGER_TYPE_8	8	死肥仔。
TRTC_VOICE_CHANGER_TYPE_9	9	强电流。
TRTC_VOICE_CHANGER_TYPE_10	10	重机械。
TRTC_VOICE_CHANGER_TYPE_11	11	空灵。

TRTCSystemVolumeType

TRTCSystemVolumeType

系统音量类型（仅适用于移动设备）

现代智能手机中一般都具备两套系统音量类型，即“通话音量”和“媒体音量”。

通话音量：手机专门为接打电话所设计的音量类型，自带回声抵消（AEC）功能，并且支持通过蓝牙耳机上的麦克风进行拾音，缺点是音质比较一般。当您通过手机侧面的音量按键下调手机音量时，如果无法将其调至零（也就是无法彻底静音），说明您的手机当前处于通话音量。

媒体音量：手机专门为音乐场景所设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。当您通过手机侧面的音量按键下调手机音量时，如果能够将手机音量调至彻底静音，说明您的手机当前处于媒体音量。

SDK 目前提供了三种系统音量类型的控制模式：自动切换模式、全程通话音量模式、全程媒体音量模式。

枚举	取值	描述

TRTCSystemVolumeTypeAuto	0	自动切换模式： 也被称为“麦上通话，麦下媒体”，即主播上麦时使用通话音量，观众不上麦则使用媒体音量，适合在线直播场景。 如果您在 enterRoom 时选择的场景为 TRTC_APP_SCENE_LIVE 或 TRTC_APP_SCENE_VOICE_CHATROOM，SDK 会自动使用该模式。
TRTCSystemVolumeTypeMedia	1	全程媒体音量： 通话全程使用媒体音量，并不是非常常用的音量类型，适用于对音质要求比较苛刻的音乐场景中。 如果您的用户大都使用外接设备（比如外接声卡）为主，可以使用该模式，否则请慎用。
TRTCSystemVolumeTypeVOIP	2	全程通话音量： 该方案的优势在于用户在上下麦时音频模块无需切换工作模式，可以做到无缝上下麦，适合于用户需要频繁上下麦的应用场景。 如果您在 enterRoom 时选择的场景为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，SDK 会自动使用该模式。

TRTCAudioFrameFormat

TRTCAudioFrameFormat

音频帧的内容格式

枚举	取值	描述
TRTC_AUDIO_FRAME_FORMAT_PCM	1	PCM 格式的音频数据。

TRTCAudioCapabilityType

TRTCAudioCapabilityType

系统支持的音频能力类型（仅适用于Android设备）

SDK 目前提供了两种系统音频能力类型是否支持的查询：低延时合唱能力、低延时耳返能力。

枚举	取值	描述
TRTCAudioCapabilityLowLatencyChorus	1	低延时合唱能力。

TRTCAudioCapabilityLowLatencyEarMonitor	2	低延时耳返能力。

TRTCAudioFrameOperationMode

TRTCAudioFrameOperationMode

音频回调数据读写模式

TRTC 提供了两种音频回调数据的操作模式。
读写模式（ReadWrite）：可以获取并修改回调的音频数据，默认模式。
只读模式（ReadOnly）：仅从回调中获取音频数据。

枚举	取值	描述
TRTC_AUDIO_FRAME_OPERATION_MODE_READWRITE	0	读写模式：可以获取并修改回调的音频数据。
TRTC_AUDIO_FRAME_OPERATION_MODE_READONLY	1	只读模式：仅从回调中获取音频数据。

TRTCLogLevel

TRTCLogLevel

Log 级别

不同的日志等级定义了不同的详实程度和日志数量，推荐一般情况下将日志等级设置为：TRTCLogLevelInfo。

枚举	取值	描述
TRTC_LOG_LEVEL_VERBOSE	0	输出所有级别的 Log。
TRTC_LOG_LEVEL_DEBUG	1	输出 DEBUG，INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTC_LOG_LEVEL_INFO	2	输出 INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTC_LOG_LEVEL_WARN	3	输出WARNING，ERROR 和 FATAL 级别的 Log。
TRTC_LOG_LEVEL_ERROR	4	输出ERROR 和 FATAL 级别的 Log。
TRTC_LOG_LEVEL_FATAL	5	仅输出 FATAL 级别的 Log。
TRTC_LOG_LEVEL_NULL	6	不输出任何 SDK Log。

版权所有：腾讯云计算（北京）有限责任公司

TRTCGSensorMode

TRTCGSensorMode

重力感应开关（仅适用于移动端）

枚举	取值	描述
TRTC_GSENSOR_MODE_DISABLE	0	不适配重力感应，该模式是桌面平台上的默认值，该模式下，当前用户发布出去的视频画面不受重力感应方向变化的影响。
TRTC_GSENSOR_MODE_UIAUTOLAYOUT	1	适配重力感应，该模式是移动平台上的默认值，该模式下，当前用户发布出去的视频画面会跟随设备的重力感应方向进行相应的调整，同时本地预览画面保持方向不变。 SDK 目前支持的一种适配模式是：当手机或 Pad 上下颠倒时，为了保证远端用户看到的画面方向正常，SDK 会自动将发布出去的画面上下旋转180度。如果您的 APP 的界面层开启了重力感应自适应，推荐使用 UIFixLayout 模式。
TRTC_GSENSOR_MODE_UIFIXLAYOUT	2	适配重力感应 该模式下，当前用户发布出去的视频画面会跟随设备的重力感应方向进行相应的调整，同时本地预览画面也会进行相应的旋转适配。 目前支持的一种特性是：当手机或 Pad 上下颠倒时，为了保证远端用户看到的画面方向正常，SDK 会自动将发布出去的画面上下旋转180度。 如果您的 APP 的界面层不支持重力感应自适应，并且希望 SDK 的视频画面能够适配重力感应方向，推荐使用 UIFixLayout 模式。 @deprecated 从v11.5版本开始，不再支持 TRTCGSensorMode_UIFixLayout，只支持上面两种模式。

TRTCTranscodingConfigMode

TRTCTranscodingConfigMode

云端混流的排版模式

TRTC 的云端混流服务能够将房间中的多路音视频流混合成一路，因此您需要指定画面的排版方案，我们提供了如下几种排版模式。

枚举	取值	描述
TRTC_TranscodingConfigMode_Unknown	0	未定义。
TRTC_TranscodingConfigMode_Manual	1	全手动排版模式。 该模式下，您需要指定每一路画面的精确排版位置。该模式的自由度最高，但易用性也最差： 您需要填写 TRTCTranscodingConfig 中的所有参数，包括每一路画面（TRTCMixUser）的位置坐标。 您需要监听 TRTCCloudListener 中的 onUserVideoAvailable() 和 onUserAudioAvailable() 事件回调，并根据当前房间中各个麦上用户的音视频状态不断地调整 mixUsers 参数。
TRTC_TranscodingConfigMode_Template_PureAudio	2	纯音频模式。 该模式适用于语音通话（AudioCall）和语音聊天室（VoiceChatRoom）等纯音频的应用场景。 您只需要在进入房间后，通过 setMixTranscodingConfig 接口设置一次，之后 SDK 就会自动把房间内所有上麦用户的声音混流到当前用户的直播流上。 您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，只需设置 audioSampleRate、audioBitrate 和 audioChannels 等参数即可。
TRTC_TranscodingConfigMode_Template_PresetLayout	3	预排版模式。 最受欢迎的排版模式，因为该模式支持您通过占位符提前对各路画面的位置进行设定，之后 SDK 会自动根据房间中画面的路数动态进行适配调整。 此模式下，您依然需要设置 mixUsers 参数，但可以将 userId 设置为“占位符”，可选的占位符有： "$PLACE_HOLDER_REMOTE$" ：指代远程用户的画面，可以设置多个。 "$PLACE_HOLDER_LOCAL_MAIN$" ：指代本地摄像头画面，只允许设置一个。

		"$PLACE_HOLDER_LOCAL_SUB$"：指代本地屏幕分享画面，只允许设置一个。 此模式下，您不需要监听 TRTCCloudListener 中的 onUserVideoAvailable 和 onUserAudioAvailable 回调进行实时调整，只需要在进房成功后调用一次 setMixTranscodingConfig 即可，之后 SDK 会自动将真实的 userId 补位到您设置的占位符上。
TRTC_TranscodingConfigMode_Template_ScreenSharing	4	屏幕分享模式。 适用于在线教育场景等以屏幕分享为主的应用场景，仅支持 Windows 和 Mac 两个平台的 SDK。 该模式下，SDK 会先根据您通过 videoWidth 和 videoHeight 参数设置的目标分辨率构建一张画布， 当老师未开启屏幕分享时，SDK 会将老师的摄像头画面等比例拉伸绘制到该画布上； 当老师开启屏幕分享之后，SDK 会将屏幕分享画面绘制到同样的画布上。 此种排版模式的目的是为了确保混流模块的输出分辨率一致，避免课程回放和网页观看的花屏问题（网页播放器不支持可变分辨率）。 同时，连麦学生的声音也会被默认混合到老师的音视频流中。 由于教学模式下的视频内容以屏幕分享为主，因此同时传输摄像头画面和屏幕分享画面是非常浪费带宽的。 推荐的做法是直接将摄像头画面通过 setLocalVideoRenderCallback 接口自定义绘制到当前屏幕上。 在该模式下，您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，SDK 不会混合学生的画面，以免干扰屏幕分享的效果。 您可以将 TRTCTranscodingConfig 中的 width x height 设为 0px × 0px，SDK 会自动根据用户当前屏幕的宽高比计算出一个合适的分辨率： 如果老师当前屏幕宽度 <= 1920px，SDK 会使用老师当前屏幕的实际分辨

		率。 如果老师当前屏幕宽度 > 1920px，SDK 会根据当前屏幕宽高比，选择 1920x1080(16:9)、1920x1200(16:10)、1920x1440(4:3) 三种分辨率中的一种。

TRTCRecordType

TRTCRecordType

媒体录制类型

该枚举类型用于本地媒体录制接口 startLocalRecording，用于指定是录制音视频文件还是纯音频文件。

枚举	取值	描述
TRTC_RECORD_TYPE_AUDIO	0	仅录制音频。
TRTC_RECORD_TYPE_VIDEO	1	仅录制视频。
TRTC_RECORD_TYPE_BOTH	2	同时录制音频和视频。

TRTCMixInputType

TRTCMixInputType

混流输入类型

枚举	取值	描述
TRTC_MixInputType_Undefined	0	默认值，考虑到针对老版本的兼容性，如果您指定了 inputType 为 Undefined，SDK 会根据另一个参数 pureAudio 的数值决定混流输入类型。
TRTC_MixInputType_AudioVideo	1	混入音频和视频。
TRTC_MixInputType_PureVideo	2	只混入视频。
TRTC_MixInputType_PureAudio	3	只混入音频。
TRTC_MixInputType_Watermark	4	混入水印，此时您无需指定 userId 字段，但需要指定 image 字段，推荐使用 png 格式的图片。

版权所有：腾讯云计算（北京）有限责任公司

TRTCDebugViewLevel

TRTCDebugViewLevel

用于渲染控件上展示的调试信息

枚举	取值	描述
TRTC_DEBUG_VIEW_LEVEL_GONE	0	不在渲染控件上展示调试信息。
TRTC_DEBUG_VIEW_LEVEL_STATUS	1	在渲染控件上展示音视频统计信息。
TRTC_DEBUG_VIEW_LEVEL_ALL	2	在渲染控件上展示音视频统计信息和关键历史事件。

TRTCAudioRecordingContent

TRTCAudioRecordingContent

音频录制内容类型

该枚举类型用于音频录制接口 startAudioRecording，用于指定录制音频的内容。

枚举	取值	描述
TRTC_AudioRecordingContent_All	0	录制本地和远端所有音频。
TRTC_AudioRecordingContent_Local	1	仅录制本地音频。
TRTC_AudioRecordingContent_Remote	2	仅录制远端音频。

TRTCPublishMode

TRTCPublishMode

媒体流发布模式

该枚举类型用于媒体流发布接口 startPublishMediaStream TRTC 的媒体流发布服务能够将房间中的多路音视频流混合成一路发布至 CDN 或者回推到房间内，也可以将您当前的这路音视频发布到腾讯或者第三方 CDN 因此您需要指定对应媒体流的发布模式，我们提供了如下几种模式。

枚举	取值	描述
TRTC_PublishMode_Unknown	0	未定义。

TRTC_PublishBigStream_ToCdn	1	您可以通过设置该参数将您房间内的主路流（TRTC_VIDEO_STREAM_TYPE_BIG）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTC_PublishSubStream_ToCdn	2	您可以通过设置该参数将您房间内的辅路流（TRTC_VIDEO_STREAM_TYPE_SUB）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTC_PublishMixStream_ToCdn	3	您可以通过设置该参数，配合编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTC_PublishMixStream_ToRoom	4	您可以通过设置该参数，配合媒体流编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您指定的房间中。 通过 TRTCPublishTarget 中的 TRTCUser 进行指定回推房间的机器人信息。

TRTCEncryptionAlgorithm

TRTCEncryptionAlgorithm

加密算法

该枚举类型用于媒体流私有加密算法选择。

枚举	取值	描述
TRTC_EncryptionAlgorithm_Aes_128_Gcm	0	AES GCM 128。
TRTC_EncryptionAlgorithm_Aes_256_Gcm	1	AES GCM 256。

TRTCSpeedTestScene

TRTCSpeedTestScene

测速场景

该枚举类型用于测速场景选择。

枚举	取值	描述

TRTC_SpeedTestScene_Delay_Testing	1	延迟测试。
TRTC_SpeedTestScene_Delay_Bandwidth_Testing	2	延迟与带宽测试。
TRTC_SpeedTestScene_Online_Chorus_Testing	3	在线合唱测试。

TRTCGravitySensorAdaptiveMode

TRTCGravitySensorAdaptiveMode

设置重力感应的适配模式（仅适用于移动端）

枚举	取值	描述
TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_DISABLE	0	关闭重力感应，根据当前采集分辨率与设置的编码分辨率决策，如果两者不一致，则通过旋转90度，保证最大画幅。
TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_FILL_BY_CENTER_CROP	1	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用居中裁剪模式。
TRTC_GRAVITY_SENSOR_ADAPTIVE_MODE_FIT_WITH_BLACK_BORDER	2	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用叠加黑边模式。

TRTCParams

TRTCParams

进房参数

作为 TRTC SDK 的进房参数，只有该参数填写正确，才能顺利进入 roomId 或者 strRoomId 所指定的音视频房间。
由于历史原因，TRTC 支持数字和字符串两种类型的房间号，分别是 roomId 和 strRoomId。
请注意：不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 “123” 在 TRTC 看来是两个完全不同的房间。

枚举类型	描述
businessInfo	【字段含义】业务数据字段（选填），部分高级特性才需要用到此字段。 【推荐取值】请不要自行设置该字段。
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
role	【字段含义】直播场景下的角色，仅适用于直播场景（TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM），通话场景下指定该参数是无效的。 【推荐取值】默认值：主播（TRTCRoleAnchor）。
roomId	【字段含义】数字房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为 0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。
sdkAppId	【字段含义】应用标识（必填），腾讯云基于 sdkAppId 完成计费统计。 【推荐取值】在 **实时音视频控制台** 创建应用后可以在账号信息页面中得到该 ID。
strRoomId	【字段含义】字符串房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。 【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`、`=`、`.`、`>`、`?`、

	`@` 、 `[` 、 `]` 、 `^` 、 `_` 、 `{` 、 `}` 、 `|` 、 `~` 、 `,` 。
streamId	【字段含义】用于指定在腾讯云直播平台上的 streamId（选填），设置之后，您可以在腾讯云直播 CDN 上通过标准拉流方案（FLV 或 HLS）播放该用户的音视频流。 【推荐取值】限制长度为64字节，可以不填写，一种推荐的方案是使用 `sdkappid_roomid_userid_main` 作为 streamid，这种命名方式容易辨认且不会在您的多个应用中发生冲突。 【特殊说明】要使用腾讯云直播 CDN，您需要先在 控制台 中的功能配置页开启“启动自动旁路直播”开关。 【参考文档】**CDN 旁路直播**。
userDefineRecordId	【字段含义】云端录制开关（选填），用于指定是否要在云端将该用户的音视频流录制下来。 【参考文档】云端录制。 【推荐取值】限制长度为64字节，只允许包含大小写英文字母（a-zA-Z）、数字（0-9）及下划线和连词符。 方案一：手动录制方案： 1. 在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `手动录制` 。 3. 设置手动录制后，在一个 TRTC 房间中只有设置了 userDefineRecordId 参数的用户才会在云端录制出视频文件，不指定该参数的用户不会产生录制行为。 4. 云端会以 “userDefineRecordId_起始时间_结束时间” 的格式命名录制下来的文件。 方案二：自动录制方案： 1. 需要在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `自动录制` 。 3. 设置自动录制后，在一个 TRTC 房间中的任何一个有音视频上行的用户，均会在云端录制出视频文件。 4. 文件会以 `userDefineRecordId_起始时间_结束时间` 的格式命名，如果不指定 userDefineRecordId，则文件会以 `streamId_起始时间_结束时间` 命名。
userId	【字段含义】用户标识（必填），当前用户的 userId，相当于用户名，使用 UTF-8 编码。 【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。
userSig	【字段含义】用户签名（必填），当前 userId 对应的验证签名，相当于使用云服务的登录密码。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCVideoEncParam

TRTCVideoEncParam

视频编码参数

该设置决定远端用户看到的画面质量，同时也决定了云端录制出的视频文件的画面质量。

枚举类型	描述
enableAdjustRes	【字段含义】是否允许动态调整分辨率（开启后会对云端录制产生影响）。 【推荐取值】该功能适用于不需要云端录制的场景，开启后 SDK 会根据当前网络情况，智能选择出一个合适的分辨率，避免出现“大分辨率+小码率”的低效编码模式。 【特别说明】默认值：关闭。如有云端录制的需求，请不要开启此功能，因为如果视频分辨率发生变化后，云端录制出的 MP4 在普通的播放器上无法正常播放。
minVideoBitrate	【字段含义】最低视频码率，SDK 会在网络不佳的情况下主动降低视频码率以保持流畅度，最低会降至 minVideoBitrate 所设定的数值。 【特别说明】 默认值：0，此时最低码率由 SDK 会根据您指定的分辨率，自动计算出合适的数值。 【推荐取值】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求 弱网络下允许卡顿但要保持清晰 的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求 弱网络下允许模糊但要保持流畅 的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
videoBitrate	【字段含义】目标视频码率，SDK 会按照目标码率进行编码，只有在弱网络环境下才会主动降低视频码率。 【推荐取值】请参见本 TRTCVideoResolution 在各档位注释的最佳码率，也可以在此基础上适当调高。比如：TRTCVideoResolution_1280_720 对应 1200kbps 的目标码率，您也可以设置为 1500kbps 用来获得更好的观感清晰度。 【特别说明】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求“弱网络下允许卡顿但要保持清晰”的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求“弱网络下允许模糊但要保持流畅”的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
videoFps	【字段含义】视频采集帧率。 【推荐取值】15fps或20fps。5fps以下，卡顿感明显。10fps以下，会有轻微卡顿感。20fps以上，会浪费带宽（电影的帧率为24fps）。

版权所有：腾讯云计算（北京）有限责任公司

	【特别说明】部分 Android 手机的前置摄像头并不支持15fps以上的采集帧率，部分主打美颜功能的 Android 手机的前置摄像头的采集帧率可能低于10fps。
videoResolution	【字段含义】视频分辨率。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如：640 × 360 + Portrait = 360 × 640。 【推荐取值】 手机视频通话：建议选择 360 × 640 及以下分辨率，resMode 选择 Portrait，即竖屏分辨率。 手机在线直播：建议选择 540 × 960，resMode 选择 Portrait，即竖屏分辨率。 桌面平台（Win + Mac）：建议选择 640 × 360 及以上分辨率，resMode 选择 Landscape，即横屏分辨率。
videoResolutionMode	【字段含义】分辨率模式（横屏分辨率 or 竖屏分辨率）。 【推荐取值】手机平台（iOS、Android）建议选择 Portrait，桌面平台（Windows、Mac）建议选择 Landscape。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如：640 × 360 + Portrait = 360 × 640。

TRTCNetworkQosParam

TRTCNetworkQosParam

网络流控（Qos）参数集

网络流控相关参数，该设置决定 SDK 在弱网络环境下的调控策略（例如：“清晰优先”或“流畅优先”）。

枚举类型	描述
controlMode	【字段含义】流控模式（已废弃）。 【推荐取值】云端控制。 【特别说明】请设置为云端控制模式（TRTCQosControlModeServer）。
preference	【字段含义】清晰优先还是流畅优先。 【推荐取值】清晰优先。 【特别说明】该参数主要影响 TRTC 在较差网络环境下的音视频表现： 流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。参见 TRTC_VIDEO_QOS_PREFERENCE_SMOOTH 清晰优先（默认值）：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。参见 TRTC_VIDEO_QOS_PREFERENCE_CLEAR

TRTCRenderParams

TRTCRenderParams

视频画面的渲染参数

您可以通过设置此参数来控制画面的旋转角度、填充模式和左右镜像模式。

枚举类型	描述
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTC_VIDEO_RENDER_MODE_FILL。
mirrorType	【字段含义】画面镜像模式。 【推荐取值】默认值：TRTC_VIDEO_MIRROR_TYPE_AUTO。
rotation	【字段含义】图像的顺时针旋转角度。 【推荐取值】支持 90、180 以及 270 旋转角度，默认值：TRTC_VIDEO_ROTATION_0。

TRTCQualityInfo

TRTCQualityInfo

网络质量

表征网络质量的好坏，您可以通过该数值在用户界面上展示每个用户的网络质量。

枚举类型	描述
quality	网络质量
userId	用户 ID

TRTCVolumeInfo

TRTCVolumeInfo

音量大小

表征语音音量的评估值，您可以通过该数值在用户界面上展示每个用户的音量大小。

枚举类型	描述
pitch	本地用户的人声频率（单位：Hz），取值范围[0 - 4000]，对于远端用户，该值始终为0。

spectrumData	音频频谱数据是将音频数据在频率域中的分布，划分为 256 个频率段，使用 spectrumData 记录各个频率段的能量值，每个能量值的取值范围为 [-300, 0]，单位为 dBFS。 **注意** 本地频谱使用编码前的音频数据计算，会受到本地采集音量、BGM等影响；远端频谱使用接收到的音频数据计算，本地调整远端播放音量等操作不会对其产生影响。
userId	说话者的 userId, 如果 userId 为空则代表是当前用户自己。
vad	是否检测到人声，0：非人声 1：人声。
volume	说话者的音量大小, 取值范围[0 - 100]。

TRTCSpeedTestParams

TRTCSpeedTestParams

测速参数

您可以在用户进入房间前通过 startSpeedTest 接口测试网速（注意：请不要在通话中调用）。

枚举类型	描述
expectedDownBandwidth	预期的下行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
expectedUpBandwidth	预期的上行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
scene	测速场景。
sdkAppId	应用标识，请参见 TRTCParams 中的相关说明。
userId	用户标识，请参见 TRTCParams 中的相关说明。
userSig	用户签名，请参见 TRTCParams 中的相关说明。

TRTCSpeedTestResult

TRTCSpeedTestResult

网络测速结果

您可以在用户进入房间前通过 startSpeedTest 接口进行测速（注意：请不要在通话中调用）。

枚举类型	描述
availableDownBandwidth	下行带宽（kbps，-1：无效值）。
availableUpBandwidth	上行带宽（kbps，-1：无效值）。
downJitter	下行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。
downLostRate	下行丢包率，取值范围是 [0 - 1.0]，例如 0.2 表示每从服务器收取 10 个数据包可能会在中途丢失 2 个。
errMsg	带宽测试错误信息。
ip	服务器 IP 地址。
quality	内部通过评估算法测算出的网络质量，更多信息请参见 TRTCQuality。
rtt	延迟（毫秒），指当前设备到 TRTC 服务器的一次网络往返时间，该值越小越好，正常数值范围是10ms - 100ms。
success	测试是否成功。
upJitter	上行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。
upLostRate	上行丢包率，取值范围是 [0 - 1.0]，例如 0.3 表示每向服务器发送 10 个数据包可能会在中途丢失 3 个。

TRTCTexture

TRTCTexture

视频纹理数据

枚举类型	描述
eglContext10	【字段含义】使用 (javax.microedition.khronos.egl.*) 定义的 OpenGL 上下文环境。
eglContext14	【字段含义】使用 (android.opengl.*) 定义的 OpenGL 上下文环境。

textureId	【字段含义】视频纹理 ID。

TRTCVideoFrame

TRTCVideoFrame

视频帧信息

TRTCVideoFrame 用来描述一帧视频画面的裸数据，也就是编码前或者解码后的视频画面数据。

枚举类型	描述
buffer	【字段含义】bufferType 为 TRTCCloudDef#TRTC_VIDEO_BUFFER_TYPE_BYTE_BUFFER 时的视频数据，承载用于 JNI 层的 Direct Buffer。
bufferType	【字段含义】视频数据结构类型。
data	【字段含义】bufferType 为 TRTCCloudDef#TRTC_VIDEO_BUFFER_TYPE_BYTE_ARRAY 时的视频数据，承载用于 java 层的字节数组。
height	【字段含义】视频高度。
pixelFormat	【字段含义】视频的像素格式。
rotation	【字段含义】视频像素的顺时针旋转角度。
texture	【字段含义】bufferType 为 TRTCCloudDef#TRTC_VIDEO_PIXEL_FORMAT_Texture_2D 时的视频数据，承载用于 OpenGL 渲染的纹理数据。
timestamp	【字段含义】视频帧的时间戳，单位毫秒。 【推荐取值】自定义视频采集时可以设置为0。若该参数为0，SDK 会自定填充 timestamp 字段，但请“均匀”地控制 sendCustomVideoData 的调用间隔。
width	【字段含义】视频宽度。

TRTCAudioFrame

TRTCAudioFrame

音频帧数据

枚举类型	描述
channel	【字段含义】声道数。
data	【字段含义】音频数据。
extraData	【字段含义】音频额外数据，远端用户通过 `onLocalProcessedAudioFrame` 写入的数据会通过该字段回调。
sampleRate	【字段含义】采样率。
timestamp	【字段含义】时间戳，单位ms。

TRTCMixUser

TRTCMixUser

云端混流中各路画面的描述信息

TRTCMixUser 用于指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
height	【字段含义】指定该路画面的高度（单位：像素）。
image	【字段含义】占位图或水印图 占位图是指当对应 userId 混流内容为纯音频时，混合后的画面中显示的是占位图片。 水印图是指一张贴在混合后画面中的半透明图片，这张图片会一直覆盖于混合后的画面上。 当指定 inputType 为 TRTCMixInputTypePureAudio 时，image 为占位图，此时需要您指定 userId。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，image 为水印图，此时不需要您指定 userId。 【推荐取值】默认值：空值，即不设置占位图或者水印图。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 “**控制台** => 应用管理 => 功能配置 => 素材管理” 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的“图片ID”，然后将“图片ID”转换成字符串类型并设置给 image 字段即可（比如假设“图片ID” 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。 image 仅在 inputType 为 TRTCMixInputTypePureAudio 或者 TRTCMixInputTypeWatermark 时才生效。

inputType	【字段含义】指定该路流的混合内容（只混音频、只混视频、混合音视频、混入水印）。 【默认取值】默认值：TRTCMixInputTypeUndefined。 【特别说明】 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 YES 时，等效于设置 inputType 为 TRTCMixInputTypePureAudio。 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 NO 时，等效于设置 inputType 为 TRTCMixInputTypeAudioVideo。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，您可以不指定 userId 字段，但需要指定 image 字段。
pureAudio	【字段含义】指定该路流是不是只混合声音。 【推荐取值】默认值：false。 【特别说明】已废弃，推荐使用8.5版本开始新引入的字段：inputType。
renderMode	【字段含义】该画面在输出时的显示模式。 【推荐取值】默认值：视频流默认为0。0为裁剪，1为缩放，2为缩放并显示黑底。 【特别说明】水印图和占位图暂时不支持设置 renderMode，默认强制拉伸处理。
roomId	【字段含义】该路音视频流所在的房间号（设置为空值代表当前用户所在的房间号）。
soundLevel	【字段含义】该路音频参与混音时的音量等级（取值范围：0 - 100）。 【默认取值】默认值：100。
streamType	【字段含义】指定该路画面是主路画面（TRTC_VIDEO_STREAM_TYPE_BIG）还是辅路画面（TRTC_VIDEO_STREAM_TYPE_SUB）。
userId	【字段含义】用户 ID
width	【字段含义】指定该路画面的宽度（单位：像素）。
x	【字段含义】指定该路画面的 x 坐标（单位：像素）。
y	【字段含义】指定该路画面的 y 坐标（单位：像素）。
zOrder	【字段含义】指定该路画面的层级（取值范围：1 - 15，不可重复）。

TRTCTranscodingConfig

TRTCTranscodingConfig

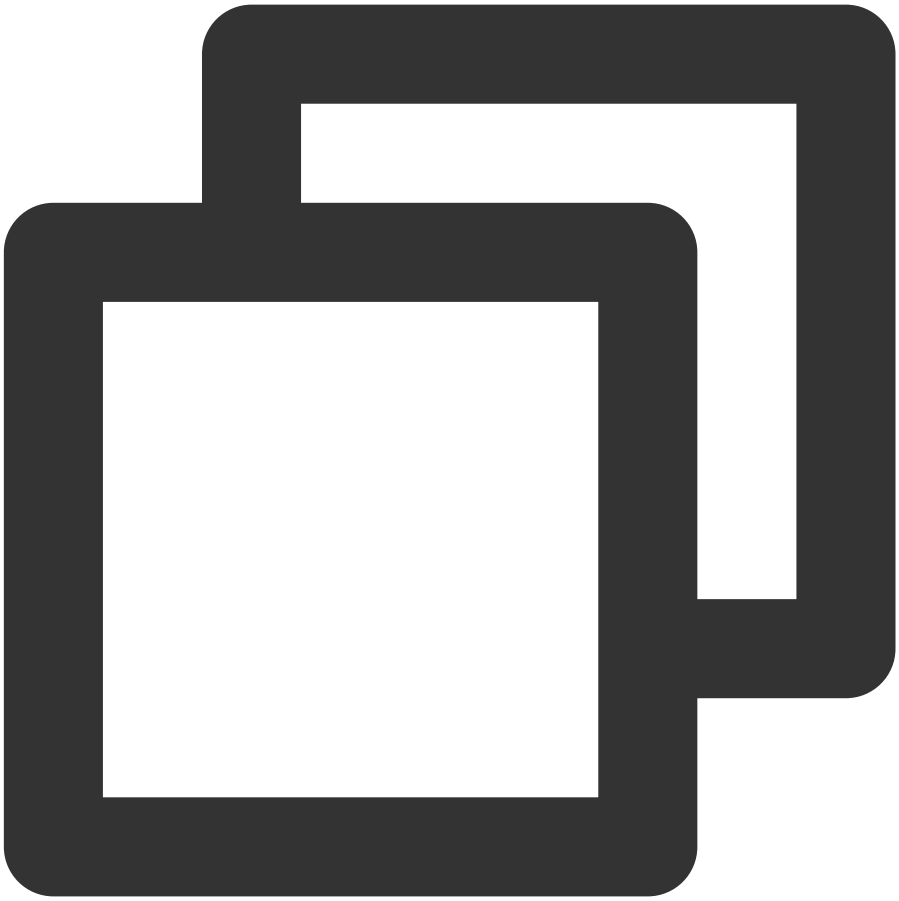
云端混流的排版布局和转码参数

用于指定混流时各路画面的排版位置信息和云端转码的编码参数。

枚举类型	描述

appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
audioBitrate	【字段含义】指定云端转码的目标音频码率。 【推荐取值】默认值：64kbps，取值范围是 [32，192]。
audioChannels	【字段含义】指定云端转码的音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioCodec	【字段含义】指定云端转码的输出流音频编码类型。 【推荐取值】默认值：0，代表LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。 【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。 【特别说明】HE-AAC 和 HE-AACv2 取值仅在输出流为您额外设置的 streamId 上时才生效。
audioSampleRate	【字段含义】指定云端转码的目标音频采样率。 【推荐取值】默认值：48000Hz。支持12000HZ、16000HZ、22050HZ、24000HZ、32000HZ、44100HZ、48000HZ。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为(97,158,241)。
backgroundImage	【字段含义】指定混合画面的背景图片。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 “**控制台** => 应用管理 => 功能配置 => 素材管理” 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的“图片ID”，然后将“图片ID”转换成字符串类型并设置给 image 字段即可（比如假设“图片ID” 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
mixUsers	【字段含义】指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。

	【推荐取值】该字段是一个 TRTCMixUser 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
mode	【字段含义】排版模式。 【推荐取值】请根据您的业务场景要求自行选择，预排版模式是适用性较好的一种模式。
streamId	【字段含义】输出到 CDN 上的直播流 ID。 【推荐取值】默认值：空值，即房间里的多路音视频流最终会混合到接口调用者的那一路音视频流上。 如不设置该参数，SDK 会执行默认逻辑，即房间里的多路音视频流会混合到该接口调用者的那一路音视频流上，也就是 A + B => A。 如您设置该参数，SDK 会将房间里的多路音视频流混合到您指定的直播流上，也就是 A + B => C（C 代表您指定的 streamId）。
videoBitrate	【字段含义】指定云端转码的目标视频码率（kbps）。 【推荐取值】如果填0，TRTC 会根据 videoWidth 和 videoHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoFramerate	【字段含义】指定云端转码的目标视频帧率（FPS）。 【推荐取值】默认值：15fps，取值范围是 (0,30]。
videoGOP	【字段含义】指定云端转码的目标视频关键帧间隔（GOP）。 【推荐取值】默认值：2，单位为秒，取值范围是 [1,8]。
videoHeight	【字段含义】指定云端转码的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：



```
{
  "payLoadContent":"xxx",
  "payloadType":5,
  "payloadUuid":"1234567890abcdef1234567890abcdef",
  "interval":1000,
  "followIdr":false
}
```

当前支持的字段及含义：
payloadContent: 必填。透传 sei 的 payload 内容，不能为空；
payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）；
payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字；
interval: 选填，默认 1000。sei 的发送间隔，单位毫秒；
followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。

videoWidth	【字段含义】指定云端转码的目标分辨率（宽度）。

	【推荐取值】单位：像素值，推荐值：360，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。

TRTCPublishCDNParam

TRTCPublishCDNParam

向非腾讯云 CDN 上发布音视频流时需设置的转推参数

TRTC 的后台服务支持通过标准 RTMP 协议，将其中的音视频流发布到第三方直播 CDN 服务商。
如果您使用腾讯云直播 CDN 服务，可无需关注此参数，直接使用 startPublish 接口即可。

枚举类型	描述
appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
streamId	【字段含义】需要转推的 streamId。 【推荐取值】默认值：空值。如果不填写，则默认转推调用者的旁路流。
url	【字段含义】指定该路音视频流在第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。 【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCAudioRecordingParams

TRTCAudioRecordingParams

本地音频文件的录制参数

该参数用于在音频录制接口 startAudioRecording 中指定录制参数。

枚举类型	描述
filePath	【字段含义】录音文件的保存路径（必填）。

	【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录音文件的格式，目前支持的格式有 PCM、WAV 和 AAC。 例如：假如您指定路径为 "mypath/record/audio.aac"，代表您希望 SDK 生成一个 AAC 格式的音频录制文件。 请您指定一个有读写权限的合法路径，否则录音文件无法生成。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值10000。默认值为0，表示不分片。
recordingContent	【字段含义】音频录制内容类型。 【特别说明】默认录制所有本地和远端音频。

TRTCLocalRecordingParams

TRTCLocalRecordingParams

本地媒体文件的录制参数

该参数用于在本地媒体文件的录制接口 startLocalRecording 中指定录制相关参数。

接口 startLocalRecording 是接口 startAudioRecording 的能力加强版本，前者可以录制视频文件，后者只能录制音频文件。

枚举类型	描述
filePath	【字段含义】录制的文件地址（必填），请确保路径有读写权限且合法，否则录制文件无法生成。 【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录制出的文件格式，目前支持的格式暂时只有 MP4。 例如：假如您指定路径为 "mypath/record/test.mp4"，代表您希望 SDK 生成一个 MP4 格式的本地视频文件。 请您指定一个有读写权限的合法路径，否则录制文件无法生成。
interval	【字段含义】interval 录制信息更新频率，单位毫秒，有效范围：1000-10000。默认值为-1，表示不回调。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值10000。默认值为0，表示不分片。
recordType	【字段含义】媒体录制类型，默认值：TRTCRecordTypeBoth，即同时录制音频和视频。

TRTCSwitchRoomConfig

TRTCSwitchRoomConfig

房间切换参数

该参数用于切换房间接口switchRoom，可以让用户从一个房间快速切换到另一个房间。

枚举类型	描述
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
roomId	【字段含义】数字房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
strRoomId	【字段含义】字符串房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
userSig	【字段含义】用户签名 [选填]，当前 userId 对应的验证签名，相当于登录密码。 如果您在切换房间时不指定新计算出的 userSig，SDK 会继续使用您在进入房间时（enterRoom）时所指定的 userSig。 这就需要您必须保证旧的 userSig 在切换房间的那一刻仍在签名允许的效期内，否则会导致房间切换失败。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCAudioFrameDelegateFormat

TRTCAudioFrameDelegateFormat

音频自定义回调的格式参数

该参数用于在音频自定义回调相关的接口中，设置 SDK 回调出来的音频数据的相关格式（包括采样率、声道数等）。

枚举类型	描述
channel	【字段含义】声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
mode	【字段含义】回调数据读写模式

	【推荐取值】TRTCAudioFrameOperationModeReadOnly：仅从回调中获取音频数据。可设定的模式有 TRTCAudioFrameOperationModeReadOnly，TRTCAudioFrameOperationModeReadWrite。
sampleRate	【字段含义】采样率。 【推荐取值】默认值：48000Hz。支持 16000, 32000, 44100, 48000。
samplesPerCall	【字段含义】采样点数。 【推荐取值】取值必须是 sampleRate/100 的整数倍。

TRTCScreenShareParams

TRTCScreenShareParams

屏幕分享参数（仅适用于 Android 平台）

该参数用于在屏幕分享接口 startScreenCapture 中指定屏幕分享时的悬浮窗等相关信息。

枚举类型	描述
enableForegroundService	@deprecated 从 v11.8 版本开始，为了适配 targetSdkVersion 34 及以上版本，屏幕分享会默认启动内置前台服务。该值设置无效
floatingView	【字段含义】可以通过此参数设置一个悬浮view。 【推荐取值】从 Android 7.0 系统开始，对于没有做过任何保活措施的后台应用，Android 系统会很快将其杀掉以减少系统能耗。 但当您的 App 处于屏幕分享状态时，不可避免地要被切到系统后台运行，此时如果能弹出悬浮窗可以避免被系统强杀掉。 同时，在手机屏幕上显示悬浮窗，也有利于告知用户当前正在做屏幕分享，以便提醒用户避免泄漏个人隐私。 【特殊说明】您也可以使用 Android 系统的 WindowsManager 接口达到同样的效果。
mediaProjection	【字段含义】可以通过此参数将你自己创建的 MediaProjection 传入给 SDK 使用。 【推荐取值】正常情况下，该参数保持为 null 即可。

TRTCAudioParallelParams

TRTCAudioParallelParams

远端音频流智能并发播放策略的参数

该参数用于设置远端音频流智能并发播放策略。

枚举类型	描述
includeUsers	【字段含义】指定用户必定能并发播放。 【特殊说明】指定必定并发播放的用户 ID 列表。这些用户不参与智能选择。 includeUsers 的数量必须小于 maxCount，否则本次并发播放设置失效。 includeUsers 仅在 maxCount > 0 时有效。当 includeUsers 生效时，参与智能并发选择的最大播放数 = maxCount - 有效 includeUsers 的数量。
maxCount	【字段含义】最大并发播放数。默认值：0。 如果 maxCount > 0，且实际人数 > maxCount，会实时智能选出 maxCount 路数据进行播放，这会极大的降低性能消耗。 如果 maxCount = 0，SDK 不限制并发播放数，在上麦人数比较多的房间可能会引发性能问题。

TRTCUser

TRTCUser

媒体流发布相关配置的用户信息

您可以通过设置该参数，配合媒体流目标发布参数 (TRTCPublishTarget) 和混流转码参数 (TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您填写的目标发布地址中

枚举类型	描述
intRoomId	【字段含义】数字房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 strRoomId，则 intRoomId 需要填写为0。若两者都填，SDK 将优先选用 intRoomId。
strRoomId	【字段含义】字符串房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 roomId，则 strRoomId 无需填写。若两者都填，SDK 将优先选用 intRoomId。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、 ! 、 # 、 $ 、 % 、 & 、 (、) 、 + 、 - 、 : 、 ; 、 < 、 = 、 . 、 > 、 ? 、 @ 、 [、] 、 ^ 、 _ 、 { 、 } 、 | 、 ~ 、 , 。
userId	【字段含义】用户标识，当前用户的 userId，相当于用户名，使用 UTF-8 编码。 【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。

TRTCPublishCdnUrl

TRTCPublishCdnUrl

向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置

该配置用于媒体流发布接口 startPublishMediaStream 中的目标推流配置 (TRTCPublishTarget)

枚举类型	描述
isInternalLine	【字段含义】指定该路音视频流是否发布至腾讯云。 【推荐取值】默认值：true。 【特别说明】若您的目标直播服务商为腾讯，请将此参数设置为 true，此时后台计费系统不会对此计算转推服务费。
rtmpUrl	【字段含义】指定该路音视频流在腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。 【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCPublishTarget

TRTCPublishTarget

目标推流配置

该配置用于媒体流发布接口 startPublishMediaStream。

枚举类型	描述
cdnUrlList	【字段含义】发布至腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【特别说明】若您的 mode 选择为 TRTCPublishMixStreamToRoom，此时您不需要设置该参数。
mixStreamIdentity	【字段含义】回推房间机器人信息。 【特别说明】仅当您的 mode 选择为 TRTCPublishMixStreamToRoom 时，您需要设置该参数。 【特别说明】设置后，该路转码音视频数据将回推到您指定的房间中。建议设置为特殊的 userId，以避免难以区分回推机器人和您通过 TRTC SDK 进房的主播。 【特别说明】参与混流的用户不支持订阅该转码流。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为手动时，您需要自行管理您想要拉取的音视频流（通常当您拉取回推房间的转码流

	时，您应该取消订阅参与转码的对应音视频单流）。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为自动时，不参与转码的用户将自动收到后台下发的转码流并不再继续接收参与转码的音视频单流。除非您明确进行取消订阅（muteRemoteVideoStream 和 muteRemoteAudio），否则转码流数据将持续下发。
mode	【字段含义】媒体流发布模式。 【推荐取值】请根据您的业务场景要求自行选择，TRTC 支持转推、转码和回推到 RTC 房间的模式。 【特别说明】若您的业务场景需要多个发布模式，您可以通过多次调用媒体流发布接口 (startPublishMediaStream) 并分别设置不同的 TRTCPublishTarget。 【特别说明】同一个 mode 请对应一个媒体流发布接口 (startPublishMediaStream)，并在后续需要调整时，使用 updatePublishCDNStream 进行更新。

TRTCVideoLayout

TRTCVideoLayout

转码视频布局

该配置用于媒体流发布接口（startPublishMediaStream）中的转码配置（TRTCStreamMixingConfig）。
用于指定转码流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为 (97,158,241)。
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTCVideoFillMode_Fill。
fixedVideoStreamType	【字段含义】指定该路画面是主路画面（TRTC_VIDEO_STREAM_TYPE_BIG）还是辅路画面（TRTC_VIDEO_STREAM_TYPE_SUB）。
fixedVideoUser	【字段含义】参与转码的用户信息。 【特别说明】用户信息 (TRTCUser) 支持不填写（即 userId、intRoomId 和 strRoomId 均不填写）。此时当发起混流的房间中有主播上行音视频数据时，TRTC 后台服务器将自动将对应主播音视频填充到您指定的布局中。
height	【字段含义】指定该路画面的高度（单位：像素）。
placeHolderImage	【字段含义】占位图 URL，即当指定用户暂时仅上行音频时，腾讯云的后台

	服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置占位图。 【特别说明】此时需要您指定用户信息 fixedVideoUser 中的 userId。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为占位图。
width	【字段含义】指定该路画面的宽度（单位：像素）。
x	【字段含义】指定该路画面的 x 坐标（单位：像素）。
y	【字段含义】指定该路画面的 y 坐标（单位：像素）。
zOrder	【字段含义】指定该路画面的层级（取值范围：0 - 15，不可重复）。

TRTCWatermark

TRTCWatermark

水印布局

该配置用于媒体流发布接口 (startPublishMediaStream) 中的转码配置 (TRTCStreamMixingConfig)

枚举类型	描述
height	【字段含义】指定该路水印画面的高度（单位：像素）。
watermarkUrl	【字段含义】水印 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。
width	【字段含义】指定该路水印画面的宽度（单位：像素）。
x	【字段含义】指定该路水印画面的 x 坐标（单位：像素）。
y	【字段含义】指定该路水印画面的 y 坐标（单位：像素）。
zOrder	【字段含义】指定该路水印画面的层级（取值范围：0 - 15，不可重复）。

TRTCStreamEncoderParam

版权所有：腾讯云计算（北京）有限责任公司

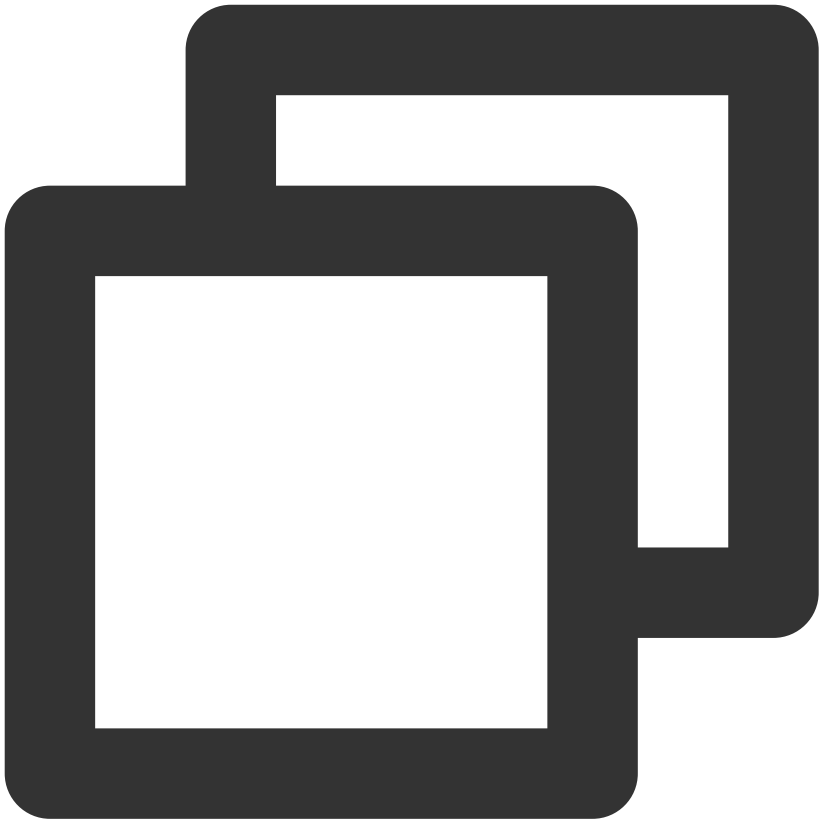
TRTCStreamEncoderParam

媒体流编码输出参数

【字段含义】该配置用于媒体流发布接口（startPublishMediaStream）。
【特别说明】当您的发布目标（TRTCPublishTarget）中的 mode 配置为 TRTCPublish_MixStream_ToCdn 或者 TRTCPublish_MixStream_ToRoom 时，该参数为必填。
【特别说明】当您使用转推服务（mode 为 TRTCPublish_BigStream_ToCdn 或者 TRTCPublish_SubStream_ToCdn）时，为了更好的转推稳定性以及更好的 CDN 播放兼容性，也建议您填写该配置的具体参数。

枚举类型	描述
audioEncodedChannelNum	【字段含义】指定媒体发布流的目标音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioEncodedCodecType	【字段含义】指定媒体发布流的目标音频编码类型。 【推荐取值】默认值：0，代表 LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。 【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。
audioEncodedKbps	【字段含义】指定媒体发布流的目标音频码率（kbps）。 【推荐取值】默认值：50kbps，取值范围是 [32，192]。
audioEncodedSampleRate	【字段含义】指定媒体发布流的目标音频采样率。 【推荐取值】默认值：48000Hz。取值为 [48000, 44100, 32000, 24000, 16000, 8000]，单位是 Hz。
videoEncodedCodecType	【字段含义】指定媒体发布流的目标视频编码类型。 【推荐取值】默认值：0，代表 H264。可设定的数值只有两个数字：0 - H264，1 - H265。
videoEncodedFPS	【字段含义】指定媒体发布流的目标视频帧率（FPS）。 【推荐取值】推荐值：20fps，取值范围是 (0,30]。
videoEncodedGOP	【字段含义】指定媒体发布流的目标视频关键帧间隔（GOP）。 【推荐取值】推荐值：3，单位为秒，取值范围是 [1,5]。
videoEncodedHeight	【字段含义】指定媒体发布流的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoEncodedKbps	【字段含义】指定媒体发布流的目标视频码率（kbps）。

版权所有：腾讯云计算（北京）有限责任公司

	【推荐取值】如果填 0，TRTC 会根据 videoEncodedWidth 和 videoEncodedHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoEncodedWidth	【字段含义】指定媒体发布流的目标分辨率（宽度）。 【推荐取值】单位：像素值，推荐值：368，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：  `{ "payLoadContent":"xxx", "payloadType":5, "payloadUuid":"1234567890abcdef1234567890abcdef", "interval":1000, "followIdr":false }` 当前支持的字段及含义： payloadContent: 必填。透传 sei 的 payload 内容，不能为空； payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）；

	payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字； interval: 选填，默认 1000。sei 的发送间隔，单位毫秒； followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。

TRTCStreamMixingConfig

TRTCStreamMixingConfig

媒体流转码配置参数

该配置用于媒体流发布接口（startPublishMediaStream）。

用于指定转码时各路画面的排版位置信息和输入的音频信息。

枚举类型	描述
audioMixUserList	【字段含义】指定转码流中的每一路输入音频的信息。 【推荐取值】该字段是一个 TRTCUser 类型的数组，数组中的每一个元素都用来代表每一路输入音频的信息。 【特别说明】用户信息支持不填写（即 audioMixUserList 为空）。此时若设置了 TRTCStreamEncoderParam 中音频相关编码输出参数，TRTC 后台服务器将自动将所有主播的音频混合输出（当前仅支持最高 16 路音视频输入）。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为 （97、158、241）。
backgroundImage	【字段含义】指定混合画面的背景图 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为背景图。
videoLayoutList	【字段含义】指定混合画面中的每一路视频画面的位置、大小、图层以及流类型等信息。 【推荐取值】该字段是一个 TRTCVideoLayout 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
watermarkList	【字段含义】指定混合画面中的每一路水印画面的位置、大小、图层等信息。 【推荐取值】该字段是一个 TRTCWatermark 类型的数组，数组中的每一个元素都用来代表每一路水印的信息。

版权所有：腾讯云计算（北京）有限责任公司

TRTCPayloadPrivateEncryptionConfig

TRTCPayloadPrivateEncryptionConfig

媒体流私有加密配置

该配置用于设置媒体流私有加密的算法和密钥。

枚举类型	描述
encryptionAlgorithm	【字段含义】加密算法，默认为 TRTCEncryptionAlgorithmAes128Gcm。
encryptionKey	【字段含义】加密用密钥，字符串类型。 【推荐取值】若加密算法为 TRTCEncryptionAlgorithmAes128Gcm，密匙长度需为 16 字节，若加密算法为 TRTCEncryptionAlgorithmAes256Gcm，密匙长度需为 32 字节。
encryptionSalt	【字段含义】盐，加密用初始向量。 【推荐取值】需确保填入该参数的数组不为空、不全为 0 且数据长度为 32 字节。

TRTCAudioVolumeEvaluateParams

TRTCAudioVolumeEvaluateParams

音量评估等相关参数设置

该设置用于开启人声检测、声音频谱计算。

枚举类型	描述
enablePitchCalculation	【字段含义】是否开启本地人声频率计算
enableSpectrumCalculation	【字段含义】是否开启声音频谱计算。
enableVadDetection	【字段含义】是否开启本地人声检测。 【请您注意】在 startLocalAudio 之前调用才可以生效。
interval	【字段含义】设置 onUserVoiceVolume 回调的触发间隔，单位为毫秒，最小间隔为 100ms，如果小于等于 0 则会关闭回调。 【推荐取值】推荐值：300，单位为毫秒。 【特别说明】interval 大于 0 时，音量提示将默认开启，无需另外设置。

版权所有：腾讯云计算（北京）有限责任公司

废弃接口

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2022 Tencent. All rights reserved.

Module: DeprecatedTRTCCloud @ TXLiteAVSDK

Function: TRTC 废弃接口

废弃接口

DeprecatedTRTCCloud

函数列表	描述
setListener	设置 TRTC 事件回调
setBeautyStyle	设置美颜、美白以及红润效果级别
setEyeScaleLevel	设置大眼级别
setFaceSlimLevel	设置瘦脸级别
setFaceVLevel	设置 V 脸级别
setChinLevel	设置下巴拉伸或收缩幅度
setFaceShortLevel	设置短脸级别
setNoseSlimLevel	设置瘦鼻级别
selectMotionTmpl	设置动效贴纸
setMotionMute	设置动效静音
setFilter	设置色彩滤镜效果
setFilterConcentration	设置色彩滤镜浓度
setGreenScreenFile	设置绿幕背景视频
setReverbType	设置混响效果

setVoiceChangerType	设置变声类型
enableAudioEarMonitoring	开启（或关闭）耳返
enableAudioVolumeEvaluation	启用音量大小提示
enableAudioVolumeEvaluation	启用音量大小提示
switchCamera	切换摄像头
isCameraZoomSupported	查询当前摄像头是否支持缩放
setZoom	设置摄像头缩放倍数（焦距）
isCameraTorchSupported	查询是否支持开关闪光灯
enableTorch	开关/关闭闪光灯
isCameraFocusPositionInPreviewSupported	查询摄像头是否支持设置焦点
setFocusPosition	设置摄像头焦点坐标位置
isCameraAutoFocusFaceModeSupported	查询是否支持自动识别人脸位置
setSystemVolumeType	设置系统音量类型
checkAudioCapabilitySupport	查询是否支持音频某种能力（仅适用于Android）
startLocalAudio	设置音频质量
startRemoteView	开始显示远端视频画面
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode	设置本地图像的渲染模式
setLocalViewRotation	设置本地图像的顺时针旋转角度
setLocalViewMirror	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode	设置远端图像的渲染模式
setRemoteViewRotation	设置远端图像的顺时针旋转角度
startRemoteSubStreamView	开始显示远端用户的辅路画面
stopRemoteSubStreamView	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode	设置辅路画面的填充模式

版权所有：腾讯云计算（北京）有限责任公司

setRemoteSubStreamViewRotation	设置辅路画面的顺时针旋转角度
setAudioQuality	设置音频质量
setPriorRemoteVideoStreamType	设定优先观看大画面还是小画面
setMicVolumeOnMixing	设置麦克风音量大小
playBGM	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration	获取背景音乐总时长（单位：毫秒）
setBGMPosition	设置背景音乐的播放进度
setBGMVolume	设置背景音乐的音量大小
setBGMPlayoutVolume	设置背景音乐的本地播放音量
setBGMPublishVolume	设置背景音乐的远端播放音量
playAudioEffect	播放音效
setAudioEffectVolume	设置音效音量
stopAudioEffect	停止播放音效
stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume	设置所有音效音量
pauseAudioEffect	暂停音效
resumeAudioEffect	暂停音效
enableCustomVideoCapture	启用视频自定义采集模式
sendCustomVideoData	投送自己采集的视频数据
muteLocalVideo	暂停/恢复发布本地的视频流
muteRemoteVideoStream	暂停 / 恢复订阅远端用户的视频流
snapshotVideo	视频画面截图

版权所有：腾讯云计算（北京）有限责任公司

startSpeedTest	开始进行网络测速（进入房间前使用）
startScreenCapture	启动屏幕分享
setVideoEncoderRotation	设置视频编码器输出的画面方向
setVideoEncoderMirror	设置编码器输出的画面镜像模式
setGSensorMode	设置重力感应的适配模式

setListener

setListener

void setListener	(TRTCCloudListener listener)

设置 TRTC 事件回调

@deprecated v11.4 版本开始不推荐使用，建议使用 addListener 替代之。

setBeautyStyle

setBeautyStyle

void setBeautyStyle	(int beautyStyle
	int beautyLevel
	int whitenessLevel
	int ruddinessLevel)

设置美颜、美白以及红润效果级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setEyeScaleLevel

setEyeScaleLevel

void setEyeScaleLevel	(int eyeScaleLevel)

设置大眼级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFaceSlimLevel

setFaceSlimLevel

void setFaceSlimLevel	(int faceScaleLevel)

设置瘦脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFaceVLevel

setFaceVLevel

void setFaceVLevel	(int faceVLevel)

设置 V 脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setChinLevel

setChinLevel

void setChinLevel	(int chinLevel)

设置下巴拉伸或收缩幅度

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFaceShortLevel

setFaceShortLevel

void setFaceShortLevel	(int faceShortlevel)

版权所有：腾讯云计算（北京）有限责任公司

设置短脸级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setNoseSlimLevel

setNoseSlimLevel

void setNoseSlimLevel	(int noseSlimLevel)

设置瘦鼻级别

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

selectMotionTmpl

selectMotionTmpl

void selectMotionTmpl	(String motionPath)

设置动效贴纸

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setMotionMute

setMotionMute

void setMotionMute	(boolean motionMute)

设置动效静音

@deprecated v6.9 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFilter

setFilter

void setFilter	(Bitmap image)

版权所有：腾讯云计算（北京）有限责任公司

设置色彩滤镜效果

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setFilterConcentration

setFilterConcentration

void setFilterConcentration	(float concentration)

设置色彩滤镜浓度

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setGreenScreenFile

setGreenScreenFile

boolean setGreenScreenFile	(String file)

设置绿幕背景视频

@deprecated v7.2 版本开始不推荐使用，建议使用 getBeautyManager 替代之。

setReverbType

setReverbType

void setReverbType	(int reverbType)

设置混响效果

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceReverbType 替代之。

setVoiceChangerType

setVoiceChangerType

boolean setVoiceChangerType	(int voiceChangerType)

设置变声类型

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceChangerType 替代之。

enableAudioEarMonitoring

enableAudioEarMonitoring

void enableAudioEarMonitoring	(boolean enable)

开启（或关闭）耳返

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setVoiceEarMonitor 替代之。

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(int interval)

启用音量大小提示

@deprecated v10.1 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(int interval
	boolean enable_vad)

启用音量大小提示

@deprecated v11.2 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

switchCamera

switchCamera

切换摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 switchCamera 接口替代之。

isCameraZoomSupported

isCameraZoomSupported

查询当前摄像头是否支持缩放

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isCameraZoomSupported 接口替代之。

setZoom

setZoom

void setZoom	(int distance)

设置摄像头缩放倍数（焦距）

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCameraZoomRatio 接口替代之。

isCameraTorchSupported

isCameraTorchSupported

查询是否支持开关闪光灯

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isCameraTorchSupported 接口替代之。

enableTorch

enableTorch

boolean enableTorch	(boolean enable)

开关/关闭闪光灯

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 enableCameraTorch 接口替代之。

isCameraFocusPositionInPreviewSupported

版权所有：腾讯云计算（北京）有限责任公司

isCameraFocusPositionInPreviewSupported

查询摄像头是否支持设置焦点

@deprecated v8.0 版本开始不推荐使用。

setFocusPosition

setFocusPosition

void setFocusPosition	(int x
	int y)

设置摄像头焦点坐标位置

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCameraFocusPosition 接口替代之。

isCameraAutoFocusFaceModeSupported

isCameraAutoFocusFaceModeSupported

查询是否支持自动识别人脸位置

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 isAutoFocusEnabled 接口替代之。

setSystemVolumeType

setSystemVolumeType

void setSystemVolumeType	(int type)

设置系统音量类型

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio(quality) 替代之，通过 quality 参数来决策音质。

checkAudioCapabilitySupport

checkAudioCapabilitySupport

int checkAudioCapabilitySupport	(int capabilityType)

查询是否支持音频某种能力（仅适用于Android）

@deprecated v10.1 版本开始不推荐使用

TRTCAudioCapabilityLowLatencyChorus，低延时合唱能力。

TRTCAudioCapabilityLowLatencyEarMonitor，低延时耳返能力。

参数	描述
capabilityType	音频能力类型.

返回值说明：

0：不支持此能力；1：支持此能力。

startLocalAudio

startLocalAudio

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

startRemoteView

startRemoteView

void startRemoteView	(String userId
	TXCloudVideoView view)

开始显示远端视频画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteView

stopRemoteView

void stopRemoteView	(String userId)

停止显示远端视频画面，同时不再拉取该远端用户的视频数据流

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setLocalViewFillMode

setLocalViewFillMode

void setLocalViewFillMode	(int mode)

设置本地图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setLocalViewRotation

setLocalViewRotation

void setLocalViewRotation	(int rotation)

设置本地图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setLocalViewMirror

setLocalViewMirror

void setLocalViewMirror	(int mirrorType)

设置本地摄像头预览画面的镜像模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setRemoteViewFillMode

setRemoteViewFillMode

void setRemoteViewFillMode	(String userId
	int mode)

设置远端图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteViewRotation

setRemoteViewRotation

void setRemoteViewRotation	(String userId
	int rotation)

设置远端图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

startRemoteSubStreamView

startRemoteSubStreamView

void startRemoteSubStreamView	(String userId
	TXCloudVideoView view)

开始显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteSubStreamView

stopRemoteSubStreamView

void stopRemoteSubStreamView	(String userId)

停止显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setRemoteSubStreamViewFillMode

setRemoteSubStreamViewFillMode

void setRemoteSubStreamViewFillMode	(String userId
	int mode)

设置辅路画面的填充模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteSubStreamViewRotation

setRemoteSubStreamViewRotation

void setRemoteSubStreamViewRotation	(final String userId
	final int rotation)

设置辅路画面的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams:streamType:params: 替代之。

setAudioQuality

setAudioQuality

void setAudioQuality	(int quality)

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

setPriorRemoteVideoStreamType

setPriorRemoteVideoStreamType

int setPriorRemoteVideoStreamType	(int streamType)

设定优先观看大画面还是小画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView:streamType:view: 替代之。

setMicVolumeOnMixing

setMicVolumeOnMixing

void setMicVolumeOnMixing	(int volume)

版权所有：腾讯云计算（北京）有限责任公司

设置麦克风音量大小

playBGM

playBGM

void playBGM	(String path
	TRTCCloud.BGMNotify notify)

启动播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

stopBGM

stopBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

pauseBGM

pauseBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

resumeBGM

resumeBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

getBGMDuration

getBGMDuration

int getBGMDuration	(String path)

获取背景音乐总时长（单位：毫秒）

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 getMusicDurationInMS 替代之。

setBGMPosition

setBGMPosition

int setBGMPosition	(int pos)

设置背景音乐的播放进度

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 seekMusicToPosInMS 替代之。

setBGMVolume

setBGMVolume

void setBGMVolume	(int volume)

设置背景音乐的音量大小

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicVolume 替代之。

setBGMPlayoutVolume

setBGMPlayoutVolume

void setBGMPlayoutVolume	(int volume)

设置背景音乐的本地播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPlayoutVolume 替代之。

setBGMPublishVolume

setBGMPublishVolume

void setBGMPublishVolume	(int volume)

设置背景音乐的远端播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setBGMPublishVolume 替代之。

playAudioEffect

playAudioEffect

void playAudioEffect	(TRTCCloudDef.TRTCAudioEffectParam effect)

播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 startPlayMusic 替代之。

setAudioEffectVolume

setAudioEffectVolume

void setAudioEffectVolume	(int effectId
	int volume)

设置音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

stopAudioEffect

stopAudioEffect

void stopAudioEffect	(int effectId)

停止播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

版权所有：腾讯云计算（北京）有限责任公司

stopAllAudioEffects

stopAllAudioEffects

停止所有音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

setAllAudioEffectsVolume

setAllAudioEffectsVolume

void setAllAudioEffectsVolume	(int volume)

设置所有音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

pauseAudioEffect

pauseAudioEffect

void pauseAudioEffect	(int effectId)

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 pauseAudioEffect 替代之。

resumeAudioEffect

resumeAudioEffect

void resumeAudioEffect	(int effectId)

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 resumePlayMusic 替代之。

enableCustomVideoCapture

enableCustomVideoCapture

void enableCustomVideoCapture	(boolean enable)

启用视频自定义采集模式

@deprecated v8.5 版本开始不推荐使用，建议使用 enableCustomVideoCapture(streamType, enable) 接口替代之。

sendCustomVideoData

sendCustomVideoData

void sendCustomVideoData	(TRTCCloudDef.TRTCVideoFrame frame)

投送自己采集的视频数据

@deprecated v8.5 版本开始不推荐使用，建议使用 sendCustomVideoData(streamType, TRTCVideoFrame) 接口替代之。

muteLocalVideo

muteLocalVideo

void muteLocalVideo	(boolean mute)

暂停/恢复发布本地的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteLocalVideo(streamType, mute) 接口替代之。

muteRemoteVideoStream

muteRemoteVideoStream

void muteRemoteVideoStream	(String userId
	boolean mute)

暂停 / 恢复订阅远端用户的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteRemoteVideoStream(userId, streamType, mute) 接口替代之。

snapshotVideo

snapshotVideo

void snapshotVideo	(String userId
	int streamType
	TRTCCloudListener.TRTCSnapshotListener listener)

视频画面截图

@deprecated v11.0 版本开始不推荐使用，建议使用 snapshotVideo(userId, streamType, sourceType, listener) 接口替代之。

startSpeedTest

startSpeedTest

void startSpeedTest	(int sdkAppId
	String userId
	String userSig)

开始进行网络测速（进入房间前使用）

@deprecated v9.2 版本开始不推荐使用，建议使用 startSpeedTest(params) 接口替代之。

startScreenCapture

startScreenCapture

void startScreenCapture	(TRTCCloudDef.TRTCVideoEncParam encParams
	TRTCCloudDef.TRTCScreenShareParams shareParams)

启动屏幕分享

@deprecated v7.2 版本开始不推荐使用，建议使用 startScreenCapture 替代之。

setVideoEncoderRotation

setVideoEncoderRotation

void setVideoEncoderRotation	(int rotation)

设置视频编码器输出的画面方向

@deprecated v11.7 版本开始不推荐使用。

setVideoEncoderMirror

setVideoEncoderMirror

void setVideoEncoderMirror	(boolean mirror)

设置编码器输出的画面镜像模式

@deprecated v11.7 版本开始不推荐使用。

setGSensorMode

setGSensorMode

void setGSensorMode	(int mode)

设置重力感应的适配模式

@deprecated v11.7 版本开始不推荐使用。建议使用 setGravitySensorAdaptiveMode 接口替代之。

错误码表

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 错误码表

Function: 用于通知客户 TRTC 在使用过程中出现的警告和错误

错误码表

枚举类型

枚举类型	描述
TXLiteAVError	错误码
TXLiteAVWarning	警告码

TXLiteAVError

TXLiteAVError

错误码

枚举	取值	描述
ERR_NULL	0	无错误
ERR_FAILED	-1	暂未归类的通用错误
ERR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法
ERR_REFUSED	-3	API 调用被拒绝
ERR_NOT_SUPPORTED	-4	当前 API 不支持调用
ERR_INVALID_LICENSE	-5	license 不合法，调用失败

ERR_REQUEST_SERVER_TIMEOUT	-6	请求服务器超时
ERR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求
ERR_DISCONNECTED	-8	断开连接
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_SCREEN_CAPTURE_START_FAIL	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了，如果在 Windows 或 Mac 系统的设备出现，请检查录屏接口的参数是否符合要求
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	录屏失败，在 Android 平台，需要5.0以上的系统，在 iOS 平台，需要11.0以上的系统
ERR_SCREEN_CAPTURE_STOPPED	-7001	录屏被系统中止
ERR_SCREEN_SHARE_NOT_AUTHORIZED	-102015	没有权限上行辅路
ERR_SCREEN_SHRAE_OCCUPIED_BY_OTHER	-102016	其他用户正在上行辅路
ERR_VIDEO_ENCODE_FAIL	-1303	视频帧编码失败，例如 iOS 设备切换到其他应用时，硬编码器可能被系统释放，再切换回来时，硬编码器重启前，可能会抛出
ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率

版权所有：腾讯云计算（北京）有限责任公司

ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	自定视频采集：设置的 pixel format 不支持
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	自定视频采集：设置的 buffer type 不支持
ERR_NO_AVAILABLE_HEVC_DECODERS	-2304	找不到可用的 HEVC 解码器
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败，例如在 Windows 或 Mac 设备，扬声器的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败
ERR_AUDIO_PLUGIN_START_FAIL	-1330	开启系统声音录制失败，例如音频驱动插件不可用
ERR_AUDIO_PLUGIN_INSTALL_NOT_AUTHORIZED	-1331	安装音频驱动插件未授权
ERR_AUDIO_PLUGIN_INSTALL_FAILED	-1332	安装音频驱动插件失败
ERR_AUDIO_PLUGIN_INSTALLED_BUT_NEED_RESTART	-1333	安装虚拟声卡插件成功，但首次安装后功能暂时不可用，此为 Mac 系统限制，请

版权所有：腾讯云计算（北京）有限责任公司

		在收到此错误码后提示用户重启当前 APP
ERR_AUDIO_ENCODE_FAIL	-1304	音频帧编码失败，例如传入自定义音频数据，SDK 无法处理
ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率
ERR_TRTC_ENTER_ROOM_FAILED	-3301	进入房间失败，请查看 onError 中的 -3301 对应的 msg 提示确认失败原因
ERR_TRTC_REQUEST_IP_TIMEOUT	-3307	请求 IP 和 sig 超时，请检查网络是否正常，或网络防火墙是否放行 UDP。 可尝试访问下列 IP： 162.14.22.165:8000 162.14.6.105:8000 和域名： default-query.trtc.tencent-cloud.com:8000
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	请求进房超时，请检查是否断网或者是否开启vpn，您也可以切换4G进行测试确认
ERR_TRTC_ROOM_PARAM_NULL	-3316	进房参数为空，请检查：enterRoom:appScene: 接口调用是否传入有效的 param
ERR_TRTC_INVALID_SDK_APPID	-3317	进房参数 sdkAppId 错误，请检查 TRTCParams.sdkAppId 是否为空
ERR_TRTC_INVALID_ROOM_ID	-3318	进房参数 roomId 错误，请检查 TRTCParams.roomId 或 TRTCParams.strRoomId 是否为空，注意 roomId 和 strRoomId 不可混用
ERR_TRTC_INVALID_USER_ID	-3319	进房参数 userId 不正确，请检查 TRTCParams.userId 是否为空
ERR_TRTC_INVALID_USER_SIG	-3320	进房参数 userSig 不正确，请检查 TRTCParams.userSig 是否为空

版权所有：腾讯云计算（北京）有限责任公司

ERR_TRTC_ENTER_ROOM_REFUSED	-3340	进房请求被拒绝，请检查是否连续调用 enterRoom 进入相同 Id 的房间
ERR_TRTC_INVALID_PRIVATE_MAPKEY	-100006	您开启了高级权限控制，但参数 TRTCParams.privateMapKey 校验失败， 您可参考 高级权限控制 进行检查
ERR_TRTC_SERVICE_SUSPENDED	-100013	服务不可用。请检查：套餐包剩余分钟数是否大于0，腾讯云账号是否欠费。 您可参考 **套餐包管理** 进行查看与配置
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig 校验失败，请检查参数 TRTCParams.userSig 是否填写正确，或是否已经过期。 您可参考 **UserSig 生成与校验** 进行校验
ERR_TRTC_PUSH_THIRD_PARTY_CLOUD_TIMEOUT	-3321	旁路转推请求超时
ERR_TRTC_MIX_TRANSCODING_TIMEOUT	-3322	云端混流请求超时
ERR_TRTC_PUSH_THIRD_PARTY_CLOUD_FAILED	-3323	旁路转推回包异常
ERR_TRTC_MIX_TRANSCODING_FAILED	-3324	云端混流回包异常
ERR_TRTC_START_PUBLISHING_TIMEOUT	-3333	开始向腾讯云的直播 CDN 推流信令超时
ERR_TRTC_START_PUBLISHING_FAILED	-3334	开始向腾讯云的直播 CDN 推流信令异常
ERR_TRTC_STOP_PUBLISHING_TIMEOUT	-3335	停止向腾讯云的直播 CDN 推流信令超时
ERR_TRTC_STOP_PUBLISHING_FAILED	-3336	停止向腾讯云的直播 CDN 推流信令异常
ERR_TRTC_CONNECT_OTHER_ROOM_TIMEOUT	-3326	请求连麦超时
ERR_TRTC_DISCONNECT_OTHER_ROOM_TIMEOUT	-3327	请求退出连麦超时

版权所有：腾讯云计算（北京）有限责任公司

ERR_TRTC_CONNECT_OTHER_ROOM_INVALID_PARAMETER	-3328	无效参数
ERR_TRTC_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	当前是观众角色，不能请求或断开跨房连麦，需要先 `switchRole` 到主播
ERR_BGM_OPEN_FAILED	-4001	打开文件失败，如音频数据无效，FFMPEG 协议未找到等
ERR_BGM_DECODE_FAILED	-4002	音频文件解码失败
ERR_BGM_OVER_LIMIT	-4003	数量超过限定值，如同时预加载两首背景音乐
ERR_BGM_INVALID_OPERATION	-4004	无效操作，如开始播放后再调用预加载操作
ERR_BGM_INVALID_PATH	-4005	非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件
ERR_BGM_INVALID_URL	-4006	非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件
ERR_BGM_NO_AUDIO_STREAM	-4007	无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏
ERR_BGM_FORMAT_NOT_SUPPORTED	-4008	格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，aac，m4a，wav，ogg，mp4，mkv】，桌面端支持【mp3，aac，m4a，wav，mp4，mkv】

TXLiteAVWarning

TXLiteAVWarning

警告码

枚举	取值	描述
WARNING_HW_ENCODER_START_FAIL	1103	硬编码启动出现问题，自动切换到软编码
WARNING_CURRENT_ENCODE_TYPE_CHANGED	1104	表示编码器发生改变，可以通过 onWarning 函数的扩展信息中的相关字段来获取当前的编码格式和类型。 type 字段值为 0 代表 H.264 编码，1 代表 H.265 编码 hardware 字段值为 0 代表软件编码，1 代表硬件编码 stream 字段值为 0 代表大流，1 代表小流，2 代表辅流
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	当前 CPU 使用率太高，无法满足软件编码需求，自动切换到硬件编码
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	摄像头采集帧率不足，部分自带美颜算法的 Android 手机上会出现
WARNING_SW_ENCODER_START_FAIL	1109	软编码启动失败
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	摄像头采集分辨率被降低，以满足当前帧率和性能最优解。
WARNING_CAMERA_DEVICE_EMPTY	1111	没有检测到可用的摄像头设备
WARNING_CAMERA_NOT_AUTHORIZED	1112	用户未授权当前应用使用摄像头
WARNING_OUT_OF_MEMORY	1113	内存不足，部分功能

版权所有：腾讯云计算（北京）有限责任公司

		可能不正常。
WARNING_CAMERA_IS_OCCUPIED	1114	摄像头被占用.
WARNING_CAMERA_DEVICE_ERROR	1115	摄像头设备异常.
WARNING_CAMERA_DISCONNECTED	1116	摄像头无法连接.
WARNING_CAMERA_START_FAILED	1117	摄像头启动失败.
WARNING_CAMERA_SERVER_DIED	1118	系统异常.
WARNING_SCREEN_CAPTURE_NOT_AUTHORIZED	1206	用户未授权当前应用使用屏幕录制
WARNING_CURRENT_DECODE_TYPE_CHANGED	2008	表示解码器发生改变，可以通过 onWarning 函数的扩展信息中的 type 字段来获取当前的解码格式。 其中 1 代表 265 解码，0 代表 264 解码。注意 Windows 端不支持此错误码的扩展信息。
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	当前视频帧解码失败
WARNING_HW_DECODER_START_FAIL	2106	硬解启动失败，采用软解码
WARNING_VIDEO_DECODER_HW_TO_SW	2108	当前流硬解第一个 I 帧失败，SDK 自动切软解
WARNING_SW_DECODER_START_FAIL	2109	软解码器启动失败
WARNING_VIDEO_RENDER_FAIL	2110	视频渲染失败
WARNING_VIRTUAL_BACKGROUND_DEVICE_UNSURPORTED	8001	虚拟背景设备不支持
WARNING_VIRTUAL_BACKGROUND_NOT_AUTHORIZED	8002	虚拟背景未授权
WARNING_VIRTUAL_BACKGROUND_INVALID_PARAMETER	8003	虚拟背景参数异常
WARNING_VIRTUAL_BACKGROUND_PERFORMANCE_INSUFFICIENT	8004	虚拟背景性能不足
WARNING_MICROPHONE_DEVICE_EMPTY	1201	没有检测到可用的麦

版权所有：腾讯云计算（北京）有限责任公司

		克风设备
WARNING_SPEAKER_DEVICE_EMPTY	1202	没有检测到可用的扬声器设备
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	用户未授权当前应用使用麦克风
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	音频采集设备不可用（例如被占用或者PC判定无效设备）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	音频播放设备不可用（例如被占用或者PC判定无效设备）
WARNING_BLUETOOTH_DEVICE_CONNECT_FAIL	1207	蓝牙设备连接失败（例如其他应用通过设置通话音量占用音频通道）
WARNING_MICROPHONE_IS_OCCUPIED	1208	音频采集设备被占用
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	当前音频帧解码失败
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	音频录制写入文件失败
WARNING_MICROPHONE_HOWLING_DETECTED	7002	录制音频时监测到啸叫
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	当前是观众角色，不支持发布音视频，需要先切换成主播角色
WARNING_UPSTREAM_AUDIO_AND_VIDEO_OUT_OF_SYNC	6006	音视频发送时间戳异常，可能引起音画不同步问题

版权所有：腾讯云计算（北京）有限责任公司

全平台（C++）
API 概览

最近更新时间：2024-06-06 15:26:14

API OVERVIEW

创建实例和事件回调

函数列表	描述
getTRTCShareInstance	创建 TRTCCloud 实例（单例模式）
destroyTRTCShareInstance	销毁 TRTCCloud 实例（单例模式）
addCallback	添加 TRTC 事件回调
removeCallback	移除 TRTC 事件回调

房间相关接口函数

函数列表	描述
enterRoom	进入房间
exitRoom	离开房间
switchRole	切换角色
switchRoom	切换房间
connectOtherRoom	请求跨房通话
disconnectOtherRoom	退出跨房通话
setDefaultStreamRecvMode	设置订阅模式（需要在进入房前设置才能生效）
createSubCloud	创建子房间实例（用于多房间并发观看）
destroySubCloud	销毁子房间实例
updateOtherRoomForwardMode	更改跨房主播在本房间的上行能力

版权所有：腾讯云计算（北京）有限责任公司

CDN 相关接口函数

函数列表	描述
startPublishing	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig	设置云端混流的排版布局和转码参数
startPublishMediaStream	开始发布媒体流
updatePublishMediaStream	更新发布媒体流
stopPublishMediaStream	停止发布媒体流

视频相关接口函数

函数列表	描述
startLocalPreview	开启本地摄像头的预览画面（移动端）
updateLocalView	更新本地摄像头的预览画面
stopLocalPreview	停止摄像头预览
muteLocalVideo	暂停/恢复发布本地的视频流
setVideoMuteImage	设置本地画面被暂停期间的替代图片
startRemoteView	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView	更新远端用户的视频渲染控件
stopRemoteView	停止订阅远端用户的视频流，并释放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源
muteRemoteVideoStream	暂停/恢复订阅远端用户的视频流

版权所有：腾讯云计算（北京）有限责任公司

muteAllRemoteVideoStreams	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam	设置视频编码器的编码参数
setNetworkQosParam	设置网络质量控制的相关参数
setLocalRenderParams	设置本地画面的渲染参数
setRemoteRenderParams	设置远端画面的渲染模式
enableSmallVideoStream	开启大小画面双路编码模式
setRemoteVideoStreamType	切换指定远端用户的大小画面
snapshotVideo	视频画面截图
setGravitySensorAdaptiveMode	设置重力感应的适配模式（11.7 及以上版本）

音频相关接口函数

函数列表	描述
startLocalAudio	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布
muteLocalAudio	暂停/恢复发布本地的音频流
muteRemoteAudio	暂停/恢复播放远端的音频流
muteAllRemoteAudio	暂停/恢复播放所有远端用户的音频流
setRemoteAudioVolume	设定某一个远端用户的声音播放音量
setAudioCaptureVolume	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
enableAudioVolumeEvaluation	启用音量大小提示
startAudioRecording	开始录音
stopAudioRecording	停止录音

版权所有：腾讯云计算（北京）有限责任公司

startLocalRecording	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition	设置 3D 音效中远端用户坐标信息
set3DSpatialReceivingRange	设置指定用户所发出声音的可被接收范围

设备管理相关接口

函数列表	描述
getDeviceManager	获取设备管理类（TXDeviceManager）

美颜特效和图像水印

函数列表	描述
setBeautyStyle	设置美颜、美白、红润等特效
setWaterMark	添加水印

背景音乐和声音特效

函数列表	描述
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集（iOS 端暂未支持）
stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）
setSystemAudioLoopbackVolume	设置系统声音的采集音量

版权所有：腾讯云计算（北京）有限责任公司

屏幕分享相关接口

函数列表	描述
startScreenCapture	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持桌面系统）
selectScreenCaptureTarget	选取要分享的屏幕或窗口（该接口仅支持桌面系统）
setSubStreamEncoderParam	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）
setSubStreamMixVolume	设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）
addExcludedShareWindow	将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）
removeExcludedShareWindow	将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
removeAllExcludedShareWindow	将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
addIncludedShareWindow	将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）
removeIncludedShareWindow	将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
removeAllIncludedShareWindow	将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

自定义采集和自定义渲染

函数列表	描述
enableCustomVideoCapture	启用/关闭视频自定义采集模式
sendCustomVideoData	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture	启用音频自定义采集模式
sendCustomAudioData	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame	启用/关闭自定义音轨

版权所有：腾讯云计算（北京）有限责任公司

mixExternalAudioFrame	向 SDK 混入自定义音轨
setMixExternalAudioVolume	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳
enableLocalVideoCustomProcess	.1 开启视频第三方美颜
setLocalVideoCustomProcessCallback	.2 设置第三方美颜的视频数据回调
setLocalVideoRenderCallback	设置本地视频自定义渲染回调
setRemoteVideoRenderCallback	设置远端视频自定义渲染回调
setAudioFrameCallback	设置音频数据自定义回调
setCapturedAudioFrameCallbackFormat	设置本地麦克风采集出的音频帧回调格式
setLocalProcessedAudioFrameCallbackFormat	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameCallbackFormat	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering	开启音频自定义播放
getCustomAudioRenderingFrame	获取可播放的音频数据

自定义消息发送接口

函数列表	描述
sendCustomCmdMsg	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg	使用 SEI 通道发送自定义消息给房间内所有用户

网络测试接口

函数列表	描述
startSpeedTest	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速

版权所有：腾讯云计算（北京）有限责任公司

调试相关接口

函数列表	描述
getSDKVersion	获取 SDK 版本信息
setLogLevel	设置 Log 输出级别
setConsoleEnabled	启用/禁用控制台日志打印
setLogCompressEnabled	启用/禁用日志的本地压缩
setLogDirPath	设置本地日志的保存路径
setLogCallback	设置日志回调
showDebugView	显示仪表盘
callExperimentalAPI	调用实验性接口

加密接口

函数列表	描述
enablePayloadPrivateEncryption	开启或关闭媒体流私有加密

错误和警告事件

函数列表	描述
onError	错误事件回调
onWarning	警告事件回调

房间相关事件回调

函数列表	描述
onEnterRoom	进入房间成功与否的事件回调

版权所有：腾讯云计算（北京）有限责任公司

onExitRoom	离开房间的事件回调
onSwitchRole	切换角色的事件回调
onSwitchRoom	切换房间的结果回调
onConnectOtherRoom	请求跨房通话的结果回调
onDisconnectOtherRoom	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode	更改跨房主播上行能力的结果回调

用户相关事件回调

函数列表	描述
onRemoteUserEnterRoom	有用户加入当前房间
onRemoteUserLeaveRoom	有用户离开当前房间
onUserVideoAvailable	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable	某远端用户发布/取消了辅路视频画面
onUserAudioAvailable	某远端用户发布/取消了自己的音频
onFirstVideoFrame	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated	远端音频状态变化的事件回调
onUserVideoSizeChanged	用户视频大小发生改变回调

网络和技术指标统计回调

函数列表	描述

onNetworkQuality	网络质量的实时统计回调
onStatistics	音视频技术指标的实时统计回调
onSpeedTestResult	网速测试的结果回调

与云端连接情况的事件回调

函数列表	描述
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复

硬件设备相关事件回调

函数列表	描述
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onUserVoiceVolume	音量大小的反馈回调
onDeviceChange	本地设备的通断状态发生变化（仅适用于桌面系统）
onAudioDeviceCaptureVolumeChanged	当前麦克风的系统采集音量发生变化
onAudioDevicePlayoutVolumeChanged	当前系统的播放音量发生变化
onSystemAudioLoopbackError	系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）
onTestMicVolume	测试麦克风时的音量回调
onTestSpeakerVolume	测试扬声器时的音量回调

自定义消息的接收事件回调

函数列表	描述
onRecvCustomCmdMsg	收到自定义消息的事件回调
onMissCustomCmdMsg	自定义消息丢失的事件回调
onRecvSEIMsg	收到 SEI 消息的回调

CDN 相关事件回调

函数列表	描述
onStartPublishing	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig	设置云端混流的排版布局和转码参数的事件回调
onStartPublishMediaStream	开始发布媒体流的事件回调
onUpdatePublishMediaStream	更新媒体流的事件回调
onStopPublishMediaStream	停止媒体流的事件回调
onCdnStreamStateChanged	RTMP/RTMPS 推流状态发生改变回调

屏幕分享相关事件回调

函数列表	描述
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused	屏幕分享暂停的事件回调
onScreenCaptureResumed	屏幕分享恢复的事件回调
onScreenCaptureStoped	屏幕分享停止的事件回调
onScreenCaptureCovered	屏幕分享的目标窗口被遮挡的事件回调（仅适用于 Windows 操作系统）

版权所有：腾讯云计算（北京）有限责任公司

本地录制和本地截图的事件回调

函数列表	描述
onLocalRecordBegin	本地录制任务已经开始的事件回调
onLocalRecording	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment	本地录制分片的事件回调
onLocalRecordComplete	本地录制任务已经结束的事件回调
onSnapshotComplete	本地截图完成的事件回调

废弃的事件回调

函数列表	描述
onUserEnter	有主播加入当前房间（已废弃）
onUserExit	有主播离开当前房间（已废弃）
onAudioEffectFinished	音效播放已结束（已废弃）
onPlayBGMBegin	开始播放背景音乐（已废弃）
onPlayBGMProgress	背景音乐的播放进度回调（已废弃）
onPlayBGMComplete	背景音乐播放已经结束（已废弃）
onSpeedTest	服务器测速的结果回调（已废弃）

视频数据自定义回调

函数列表	描述
onRenderVideoFrame	自定义视频渲染回调
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame	用于对接第三方美颜组件的视频处理回调
onGLContextDestroy	SDK 内部 OpenGL 环境被销毁的通知

版权所有：腾讯云计算（北京）有限责任公司

音频数据自定义回调

函数列表	描述
onCapturedAudioFrame	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onPlayAudioFrame	混音前的每一路远程用户的音频数据
onMixedPlayAudioFrame	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）

更多事件回调接口

函数列表	描述
onLog	本地 LOG 的打印回调

背景音乐预加载事件回调

函数列表	描述
onLoadProgress	背景音乐预加载进度
onLoadError	背景音乐预加载出错

背景音乐的播放事件回调

函数列表	描述
onStart	背景音乐开始播放
onPlayProgress	背景音乐的播放进度

版权所有：腾讯云计算（北京）有限责任公司

onComplete	背景音乐已经播放完毕

人声相关的特效接口

函数列表	描述
enableVoiceEarMonitor	开启耳返
setVoiceEarMonitorVolume	设置耳返音量
setVoiceReverbType	设置人声的混响效果
setVoiceChangerType	设置人声的变声特效
setVoiceCaptureVolume	设置语音音量
setVoicePitch	设置语音音调

背景音乐的相关接口

函数列表	描述
setMusicObserver	设置背景音乐的事件回调接口
startPlayMusic	开始播放背景音乐
stopPlayMusic	停止播放背景音乐
pausePlayMusic	暂停播放背景音乐
resumePlayMusic	恢复播放背景音乐
setAllMusicVolume	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume	设置某一首背景音乐的本地音量的大小
setMusicPitch	调整背景音乐的音调高低
setMusicSpeedRate	调整背景音乐的变速效果
getMusicCurrentPosInMS	获取背景音乐的播放进度（单位：毫秒）

版权所有：腾讯云计算（北京）有限责任公司

getMusicDurationInMS	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInTime	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate	调整搓碟的变速效果
setPreloadObserver	设置预加载事件回调
preloadMusic	预加载背景音乐
getMusicTrackCount	获取背景音乐的音轨数量
setMusicTrack	指定背景音乐的播放音轨

设备操作接口

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera	切换前置或后置摄像头（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition	设置摄像头的对焦位置（仅适用于移动端）
enableCameraTorch	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）
setAudioRoute	设置音频路由（仅适用于移动端）
getDevicesList	获取设备列表（仅适用于桌面端）
setCurrentDevice	设置当前要使用的设备（仅适用于桌面端）
getCurrentDevice	获取当前正在使用的设备（仅适用于桌面端）
setCurrentDeviceVolume	设置当前设备的音量（仅适用于桌面端）
getCurrentDeviceVolume	获取当前设备的音量（仅适用于桌面端）
setCurrentDeviceMute	设置当前设备的静音状态（仅适用于桌面端）

版权所有：腾讯云计算（北京）有限责任公司

getCurrentDeviceMute	获取当前设备的静音状态（仅适用于桌面端）
enableFollowingDefaultAudioDevice	设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）
startCameraDeviceTest	开始摄像头测试（仅适用于桌面端）
stopCameraDeviceTest	结束摄像头测试（仅适用于桌面端）
startMicDeviceTest	开始麦克风测试（仅适用于桌面端）
stopMicDeviceTest	结束麦克风测试（仅适用于桌面端）
startSpeakerDeviceTest	开始扬声器测试（仅适用于桌面端）
stopSpeakerDeviceTest	结束扬声器测试（仅适用于桌面端）
setApplicationPlayVolume	设置 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）
getApplicationPlayVolume	获取 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）
setApplicationMuteState	设置 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）
getApplicationMuteState	获取 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）
setCameraCapturerParam	设置摄像头采集偏好
setDeviceObserver	设置 onDeviceChanged 事件回调

弃用接口

函数列表	描述
setSystemVolumeType	设置系统音量类型（仅适用于移动端）

废弃接口

函数列表	描述

enableAudioVolumeEvaluation	启用音量大小提示
startLocalAudio	设置音频质量
startRemoteView	开始显示远端视频画面
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode	设置本地图像的渲染模式
setLocalViewRotation	设置本地图像的顺时针旋转角度
setLocalViewMirror	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode	设置远端图像的渲染模式
setRemoteViewRotation	设置远端图像的顺时针旋转角度
startRemoteSubStreamView	开始显示远端用户的辅路画面
stopRemoteSubStreamView	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode	设置辅路画面的填充模式
setRemoteSubStreamViewRotation	设置辅路画面的顺时针旋转角度
setAudioQuality	设置音频质量
setPriorRemoteVideoStreamType	设定优先观看大画面还是小画面
setMicVolumeOnMixing	设置麦克风音量大小
playBGM	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration	获取背景音乐总时长（单位：毫秒）
setBGMPosition	设置背景音乐的播放进度
setBGMVolume	设置背景音乐的音量大小
setBGMPlayoutVolume	设置背景音乐的本地播放音量
setBGMPublishVolume	设置背景音乐的远端播放音量

版权所有：腾讯云计算（北京）有限责任公司

playAudioEffect	播放音效
setAudioEffectVolume	设置音效音量
stopAudioEffect	停止播放音效
stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume	设置所有音效音量
pauseAudioEffect	暂停音效
resumeAudioEffect	暂停音效
enableCustomVideoCapture	启用视频自定义采集模式
sendCustomVideoData	投送自己采集的视频数据
muteLocalVideo	暂停/恢复发布本地的视频流
muteRemoteVideoStream	暂停 / 恢复订阅远端用户的视频流
startSpeedTest	开始进行网络测速（进入房间前使用）
startScreenCapture	启动屏幕分享
setLocalVideoProcessCallback	设置第三方美颜的视频数据回调
getCameraDevicesList	获取摄像头设备列表
setCurrentCameraDevice	选定当前要使用的摄像头
getCurrentCameraDevice	获取当前使用的摄像头
getMicDevicesList	获取麦克风设备列表
getCurrentMicDevice	获取当前的麦克风设备
setCurrentMicDevice	选定当前使用的麦克风
getCurrentMicDeviceVolume	获取当前麦克风的设备音量
setCurrentMicDeviceVolume	设置当前麦克风的设备音量
setCurrentMicDeviceMute	设置系统当前麦克风设备的静音状态
getCurrentMicDeviceMute	获取系统当前麦克风设备是否静音
getSpeakerDevicesList	获取扬声器设备列表

版权所有：腾讯云计算（北京）有限责任公司

getCurrentSpeakerDevice	获取当前的扬声器设备
setCurrentSpeakerDevice	设置要使用的扬声器
getCurrentSpeakerVolume	获取当前扬声器的设备音量
setCurrentSpeakerVolume	设置当前扬声器的设备音量
getCurrentSpeakerDeviceMute	获取系统当前扬声器设备是否静音
setCurrentSpeakerDeviceMute	设置系统当前扬声器设备的静音状态
startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止进行摄像头测试
startMicDeviceTest	开始进行麦克风测试
stopMicDeviceTest	开始进行麦克风测试
startSpeakerDeviceTest	开始进行扬声器测试
stopSpeakerDeviceTest	停止进行扬声器测试
selectScreenCaptureTarget	开始应用内的屏幕分享（iOS）
setVideoEncoderRotation	设置视频编码器输出的画面方向
setVideoEncoderMirror	设置编码器输出的画面镜像模式

版权所有：腾讯云计算（北京）有限责任公司

ITRTCCloud

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTCCloud @ TXLiteAVSDK
Function: 腾讯云 TRTC 主功能接口
Version: 11.9

ITRTCCloud

ITRTCCloud

函数列表	描述
getTRTCShareInstance	创建 TRTCCloud 实例（单例模式）
destroyTRTCShareInstance	销毁 TRTCCloud 实例（单例模式）
addCallback	添加 TRTC 事件回调
removeCallback	移除 TRTC 事件回调
enterRoom	进入房间
exitRoom	离开房间
switchRole	切换角色
switchRole	切换角色（支持设置权限位）
switchRoom	切换房间
connectOtherRoom	请求跨房通话
disconnectOtherRoom	退出跨房通话
setDefaultStreamRecvMode	设置订阅模式（需要在进入房前设置才能生效）
createSubCloud	创建子房间实例（用于多房间并发观看）
destroySubCloud	销毁子房间实例

updateOtherRoomForwardMode	更改跨房主播在本房间的上行能力
startPublishing	开始向腾讯云直播 CDN 上发布音视频流
stopPublishing	停止向腾讯云直播 CDN 上发布音视频流
startPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流
stopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流
setMixTranscodingConfig	设置云端混流的排版布局和转码参数
startPublishMediaStream	开始发布媒体流
updatePublishMediaStream	更新发布媒体流
stopPublishMediaStream	停止发布媒体流
startLocalPreview	开启本地摄像头的预览画面（移动端）
startLocalPreview	开启本地摄像头的预览画面（桌面端）
updateLocalView	更新本地摄像头的预览画面
stopLocalPreview	停止摄像头预览
muteLocalVideo	暂停/恢复发布本地的视频流
setVideoMuteImage	设置本地画面被暂停期间的替代图片
startRemoteView	订阅远端用户的视频流，并绑定视频渲染控件
updateRemoteView	更新远端用户的视频渲染控件
stopRemoteView	停止订阅远端用户的视频流，并释放渲染控件
stopAllRemoteView	停止订阅所有远端用户的视频流，并释放全部渲染资源
muteRemoteVideoStream	暂停/恢复订阅远端用户的视频流
muteAllRemoteVideoStreams	暂停/恢复订阅所有远端用户的视频流
setVideoEncoderParam	设置视频编码器的编码参数
setNetworkQosParam	设置网络质量控制的相关参数
setLocalRenderParams	设置本地画面的渲染参数
setRemoteRenderParams	设置远端画面的渲染模式

版权所有：腾讯云计算（北京）有限责任公司

enableSmallVideoStream	开启大小画面双路编码模式
setRemoteVideoStreamType	切换指定远端用户的大小画面
snapshotVideo	视频画面截图
setGravitySensorAdaptiveMode	设置重力感应的适配模式（11.7 及以上版本）
startLocalAudio	开启本地音频的采集和发布
stopLocalAudio	停止本地音频的采集和发布
muteLocalAudio	暂停/恢复发布本地的音频流
muteRemoteAudio	暂停/恢复播放远端的音频流
muteAllRemoteAudio	暂停/恢复播放所有远端用户的音频流
setRemoteAudioVolume	设定某一个远端用户的声音播放音量
setAudioCaptureVolume	设定本地音频的采集音量
getAudioCaptureVolume	获取本地音频的采集音量
setAudioPlayoutVolume	设定远端音频的播放音量
getAudioPlayoutVolume	获取远端音频的播放音量
enableAudioVolumeEvaluation	启用音量大小提示
startAudioRecording	开始录音
stopAudioRecording	停止录音
startLocalRecording	开启本地媒体录制
stopLocalRecording	停止本地媒体录制
setRemoteAudioParallelParams	设置远端音频流智能并发播放策略
enable3DSpatialAudioEffect	启用 3D 音效
updateSelf3DSpatialPosition	设置 3D 音效中自身坐标及朝向信息
updateRemote3DSpatialPosition	设置 3D 音效中远端用户坐标信息
set3DSpatialReceivingRange	设置指定用户所发出声音的可被接收范围
getDeviceManager	获取设备管理类（TXDeviceManager）

版权所有：腾讯云计算（北京）有限责任公司

setBeautyStyle	设置美颜、美白、红润等特效
setWaterMark	添加水印
getAudioEffectManager	获取音效管理类（TXAudioEffectManager）
startSystemAudioLoopback	开启系统声音采集（iOS 端暂未支持）
stopSystemAudioLoopback	停止系统声音采集（iOS 端暂未支持）
setSystemAudioLoopbackVolume	设置系统声音的采集音量
startScreenCapture	启动屏幕分享
stopScreenCapture	停止屏幕分享
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持桌面系统）
selectScreenCaptureTarget	选取要分享的屏幕或窗口（该接口仅支持桌面系统）
setSubStreamEncoderParam	设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）
setSubStreamMixVolume	设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）
addExcludedShareWindow	将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）
removeExcludedShareWindow	将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
removeAllExcludedShareWindow	将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）
addIncludedShareWindow	将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）
removeIncludedShareWindow	将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
removeAllIncludedShareWindow	将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）
enableCustomVideoCapture	启用/关闭视频自定义采集模式

版权所有：腾讯云计算（北京）有限责任公司

sendCustomVideoData	向 SDK 投送自己采集的视频帧
enableCustomAudioCapture	启用音频自定义采集模式
sendCustomAudioData	向 SDK 投送自己采集的音频数据
enableMixExternalAudioFrame	启用/关闭自定义音轨
mixExternalAudioFrame	向 SDK 混入自定义音轨
setMixExternalAudioVolume	设置推流时混入外部音频的推流音量和播放音量
generateCustomPTS	生成自定义采集时的时间戳
enableLocalVideoCustomProcess	.1 开启视频第三方美颜
setLocalVideoCustomProcessCallback	.2 设置第三方美颜的视频数据回调
setLocalVideoRenderCallback	设置本地视频自定义渲染回调
setRemoteVideoRenderCallback	设置远端视频自定义渲染回调
setAudioFrameCallback	设置音频数据自定义回调
setCapturedAudioFrameCallbackFormat	设置本地麦克风采集出的音频帧回调格式
setLocalProcessedAudioFrameCallbackFormat	设置经过前处理后的本地音频帧回调格式
setMixedPlayAudioFrameCallbackFormat	设置最终要由系统播放出的音频帧回调格式
enableCustomAudioRendering	开启音频自定义播放
getCustomAudioRenderingFrame	获取可播放的音频数据
sendCustomCmdMsg	使用 UDP 通道发送自定义消息给房间内所有用户
sendSEIMsg	使用 SEI 通道发送自定义消息给房间内所有用户
startSpeedTest	开始进行网速测试（进入房间前使用）
stopSpeedTest	停止网络测速
getSDKVersion	获取 SDK 版本信息
setLogLevel	设置 Log 输出级别
setConsoleEnabled	启用/禁用控制台日志打印
setLogCompressEnabled	启用/禁用日志的本地压缩

版权所有：腾讯云计算（北京）有限责任公司

setLogDirPath	设置本地日志的保存路径
setLogCallback	设置日志回调
showDebugView	显示仪表盘
callExperimentalAPI	调用实验性接口
enablePayloadPrivateEncryption	开启或关闭媒体流私有加密

getTRTCShareInstance

getTRTCShareInstance

ITRTCCloud* getTRTCShareInstance	(void *context)

创建 TRTCCloud 实例（单例模式）

参数	描述
context	仅适用于 Android 平台，SDK 内部会将其转化为 Android 平台的 ApplicationContext 用于调用 Android System API。 如果传入的 context 参数为空，SDK 内部会自动获取当前进程的 ApplicationContext。

注意

1. 如果您使用 delete ITRTCCloud* 会导致编译错误，请使用 destroyTRTCCloud 释放对象指针。
2. 在 Windows、Mac 和 iOS 平台上，请调用 getTRTCShareInstance() 接口。
3. 在 Android 平台上，请调用 getTRTCShareInstance(void *context) 接口。

destroyTRTCShareInstance

destroyTRTCShareInstance

销毁 TRTCCloud 实例（单例模式）

addCallback

addCallback

void addCallback	(ITRTCCloudCallback* callback)

添加 TRTC 事件回调

您可以通过 ITRTCCloudCallback 获得来自 SDK 的各类事件通知（比如：错误码，警告码，音视频状态参数等）。

removeCallback

removeCallback

void removeCallback	(ITRTCCloudCallback* callback)

移除 TRTC 事件回调

enterRoom

enterRoom

void enterRoom	(const TRTCParams& param
	TRTCAppScene scene)

进入房间

TRTC 的所有用户都需要进入房间才能“发布”或“订阅”音视频流，“发布”是指将自己的音频和视频推送到云端，“订阅”是指从云端拉取房间里其他用户的音视频流。
调用该接口时，您需要指定您的应用场景 TRTCAppScene 以获取最佳的音视频传输体验，这些场景可以分成两大类：

实时通话：
包括 TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall 两个可选项，分别是视频通话和语音通话，该模式适合1对1 的音视频通话，或者参会人数在 300 人以内的在线会议。

在线直播：
包括 TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom 两个可选项，分别是视频直播和语音直播，该模式适合十万人以内的在线直播场景，但需要您在接下来介绍的 TRTCParams 参数中指定 **角色(role)** 这个字段，也就是将房间中的用户区分为 **主播** (TRTCRoleAnchor) 和 **观众** (TRTCRoleAudience) 两种不同的角色。

调用该接口后，您会收到来自 ITRTCCloudCallback 中的 onEnterRoom(result) 回调：
如果进房成功，参数 result 会是一个正数（result > 0），表示从函数调用到进入房间所花费的时间，单位是毫秒（ms）。
如果进房失败，参数 result 会是一个负数（result < 0），表示进房失败的**错误码**。

参数	描述
param	进房参数，用于指定用户的身份、角色和安全票据等信息，详情请参见 TRTCParams 。
scene	应用场景，用于指定您的业务场景，同一个房间内的所有用户需要设定相同的 TRTCAppScene。

注意

1. 同一个房间内的所有用户需要设定相同的 scene。不同的 scene 会导致偶现的异常问题。
2. 当您指定参数 scene 为 TRTCAppSceneLIVE 或 TRTCAppSceneVoiceChatRoom 时，您必须通过 TRTCParams 中的 “role” 字段为当前用户设定他/她在房间中的角色。
3. 请您尽量保证 enterRoom 与 exitRoom 前后配对使用，即保证”先退出前一个房间再进入下一个房间”，否则会导致很多异常问题。

exitRoom

exitRoom

离开房间

调用该接口会让用户离开自己所在的音视频房间，并释放摄像头、麦克风、扬声器等设备资源。
等资源释放完毕之后，SDK 会通过 ITRTCCloudCallback 中的 onExitRoom 回调向您通知。

如果您要再次调用 enterRoom 或者切换到其他的供应商的 SDK，建议等待 onExitRoom 回调到来之后再执行之后的操作，以避免摄像头或麦克风被占用的问题。

switchRole

switchRole

void switchRole	(TRTCRoleType role)

切换角色

调用本接口可以实现用户在 主播 和 观众 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了 **只有主播才能发布自己的音视频** 的规则。因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成 **主播**。

您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

参数	描述
role	角色，默认为 **主播**。 TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。 TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成 **主播**，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTCAppSceneLIVE）和语音聊天室（TRTCAppSceneVoiceChatRoom）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，请不要调用这个接口。

switchRole

switchRole

void switchRole	(TRTCRoleType role
	const char* privateMapKey)

切换角色（支持设置权限位）

调用本接口可以实现用户在 `主播` 和 `观众` 两种角色之间来回切换。

由于视频直播和语音聊天室需要支持多达10万名观众同时观看，所以设定了“只有主播才能发布自己的音视频”的规则。
因此，当有些观众希望发布自己的音视频流（以便能跟主播互动）时，就需要先把自己的角色切换成“主播”。
您可以在进入房间时通过 TRTCParams 中的 role 字段事先确定用户的角色，也可以在进入房间后通过 switchRole 接口动态切换角色。

参数	描述
privateMapKey	用于权限控制的权限票据，当您希望某个房间只能让特定的 userId 进入或者上行视频时，需要使用 privateMapKey 进行权限保护。 仅建议有高级别安全需求的客户使用，更多详情请参见 开启高级权限控制。
role	角色，默认为“主播”：

	TRTCRoleAnchor ：主播，可以发布自己的音视频，同一个房间里最多支持50个主播同时发布音视频。 TRTCRoleAudience ：观众，不能发布自己的音视频流，只能观看房间中其他主播的音视频。如果要发布自己的音视频，需要先通过 switchRole 切换成“主播”，同一个房间内同时最多可以容纳 10 万名观众。

注意

1. 该接口仅适用于视频直播（TRTCAppSceneLIVE）和语音聊天室（TRTCAppSceneVoiceChatRoom）这两个场景。
2. 如果您在 enterRoom 时指定的 scene 为 TRTCAppSceneVideoCall 或 TRTCAppSceneAudioCall，请不要调用这个接口。

switchRoom

switchRoom

void switchRoom	(const TRTCSwitchRoomConfig& config)

切换房间

使用该接口可以让用户可以快速从一个房间切换到另一个房间。
如果用户的身份是“观众”，该接口的调用效果等同于 exitRoom(当前房间) + enterRoom（新的房间）。
如果用户的身份是“主播”，该接口在切换房间的同时还会保持自己的音视频发布状态，因此在房间切换过程中，摄像头的预览和声音的采集都不会中断。

该接口适用于在线教育场景中，监课老师在多个房间中进行快速切换的场景。在该场景下使用 switchRoom 可以获得比 exitRoom+enterRoom 更好的流畅性和更少的代码量。
接口调用结果会通过 ITRTCCloudCallback 中的 onSwitchRoom(errCode, errMsg) 回调。

参数	描述
config	房间参数，详情请参见 TRTCSwitchRoomConfig。

注意

由于对老版本 SDK 兼容的需求，参数 config 中同时包含 roomId 与 strRoomId 两个参数，这两个参数的填写格外讲究，请注意如下事项：
1. 若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，将优先选用 roomId。
2. 所有房间需要同时使用 strRoomId 或同时使用 roomId，不可混用，否则将会出现很多预期之外的 bug。

版权所有：腾讯云计算（北京）有限责任公司

connectOtherRoom

connectOtherRoom

void connectOtherRoom	(const char* param)

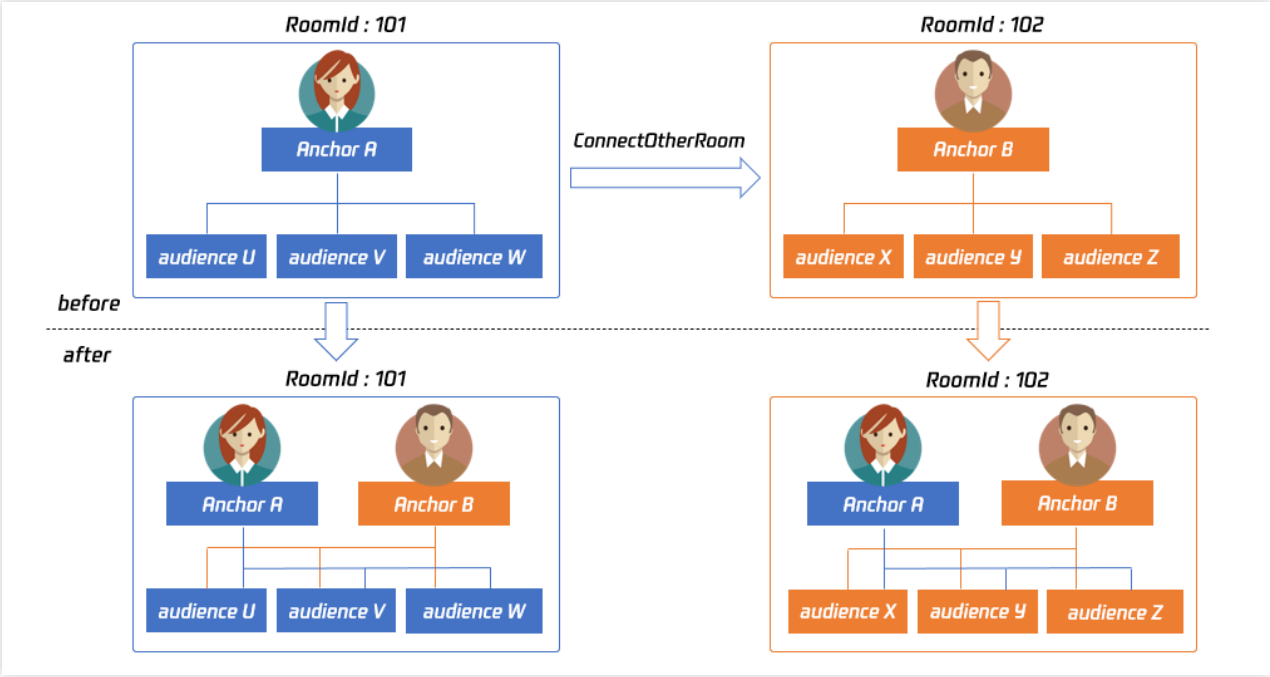
请求跨房通话

默认情况下，只有同一个房间中的用户之间可以进行音视频通话，不同的房间之间的音视频流是相互隔离的。
但您可以通过调用该接口，将另一个房间中某个主播音视频流发布到自己所在的房间中，与此同时，该接口也会将自己的音视频流发布到目标主播的房间中。

也就是说，您可以使用该接口让身处两个不同房间中的主播进行跨房间的音视频流分享，从而让每个房间中的观众都能观看到这两个主播的音视频。该功能可以用来实现主播之间的 PK 功能。

跨房通话的请求结果会通过 ITRTCCloudCallback 中的 onConnectOtherRoom 回调通知给您。

例如：当房间“101”中的主播 A 通过 connectOtherRoom() 跟房间“102”中的主播 B 建立跨房通话后，
房间“101”中的用户都会收到主播 B 的 onRemoteUserEnterRoom(B) 和 onUserVideoAvailable(B,true) 这两个事件回调，即房间“101”中的用户都可以订阅主播 B 的音视频。
房间“102”中的用户都会收到主播 A 的 onRemoteUserEnterRoom(A) 和 onUserVideoAvailable(A,true) 这两个事件回调，即房间“102”中的用户都可以订阅主播 A 的音视频。

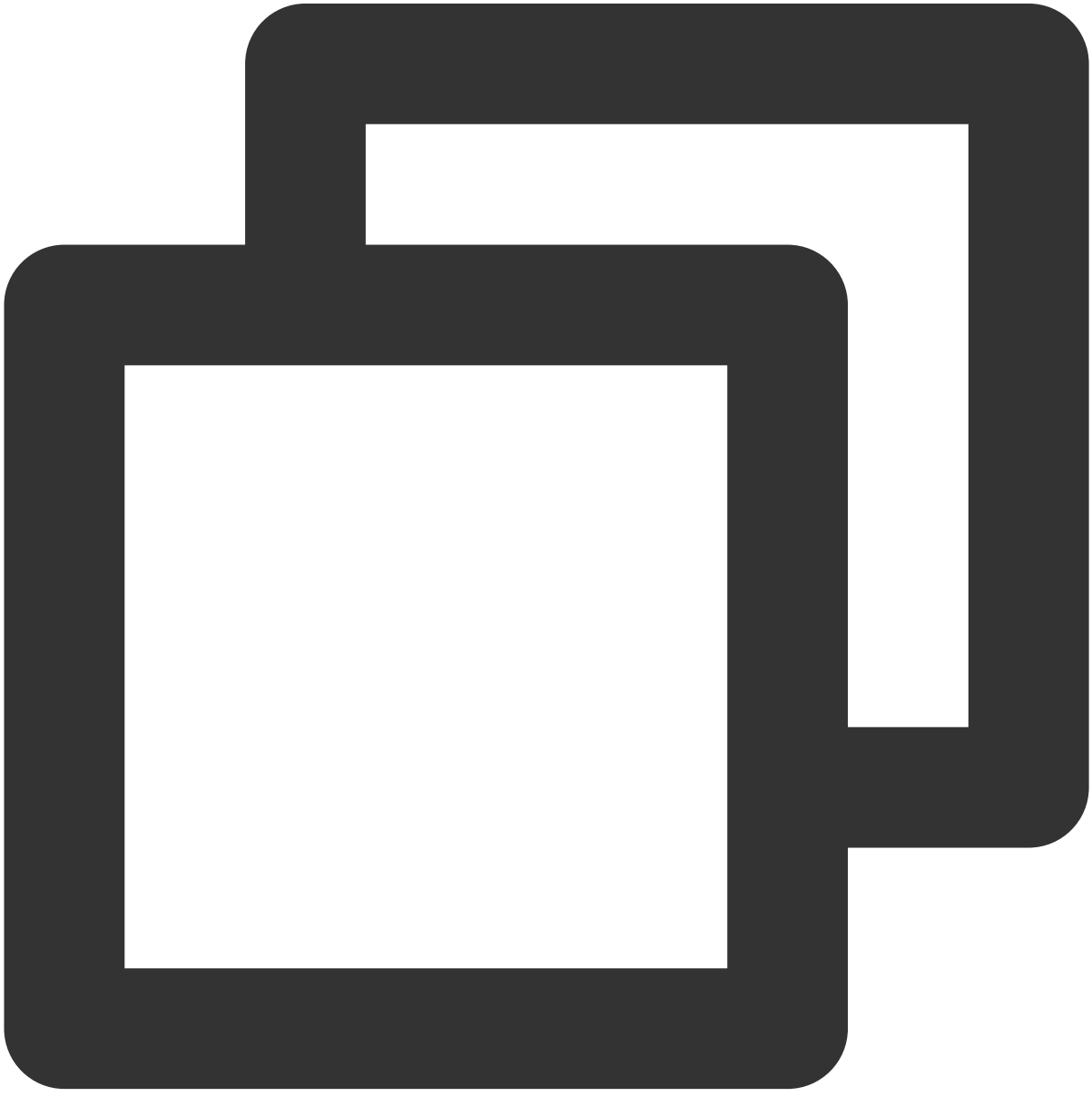


跨房通话的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数：

情况一：数字房间号

如果房间“101”中的主播 A 要跟房间“102”中的主播 B 连麦，那么主播 A 调用该接口时需要传入：{"roomId": 102, "userId": "userB"}

示例代码如下：



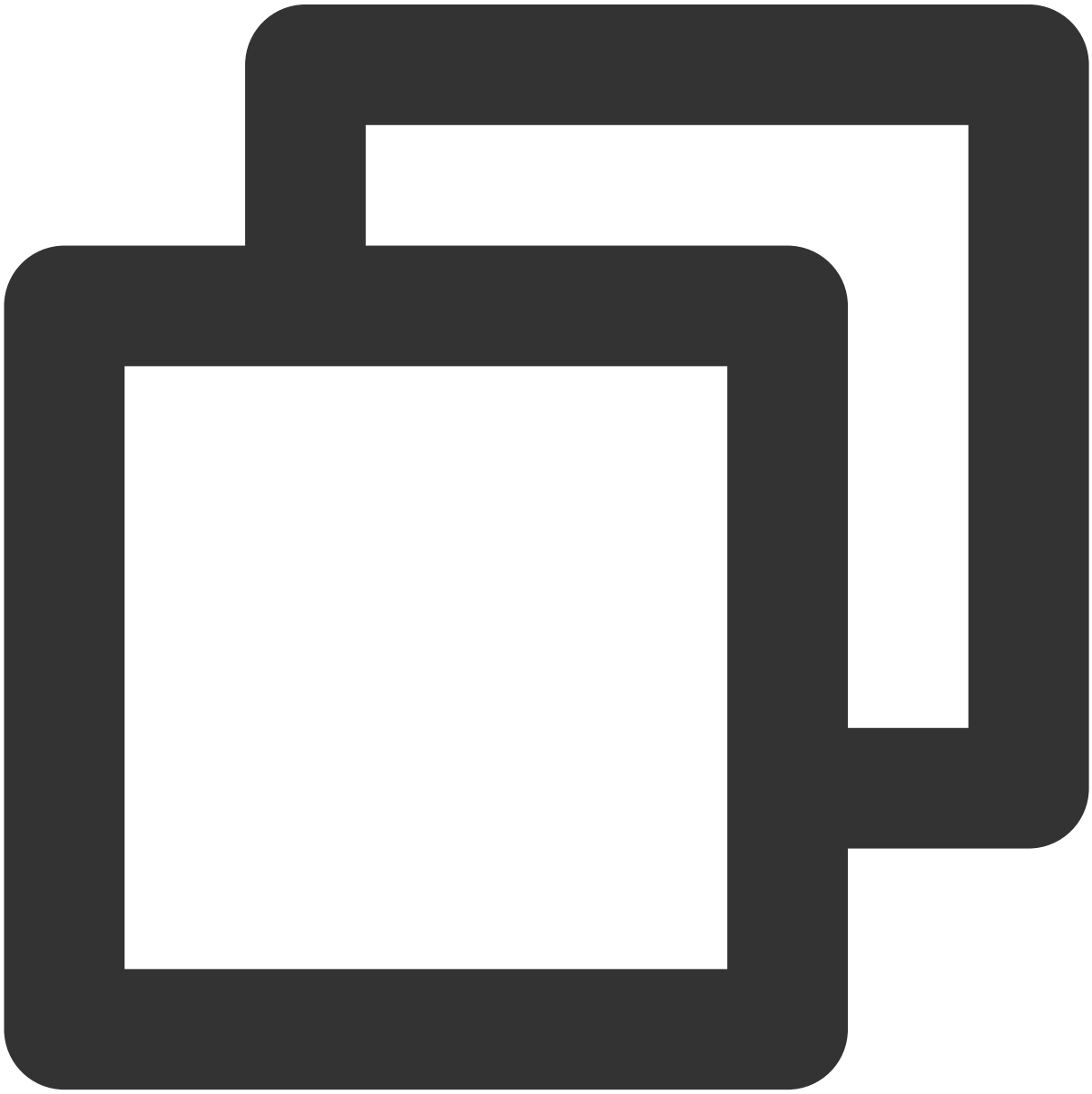
```
Json::Value jsonObj;
jsonObj["roomId"] = 102;
jsonObj["userId"] = "userB";
```

```
Json::FastWriter writer;
std::string params = writer.write(jsonObj);
trtc.ConnectOtherRoom(params.c_str());
```

情况二：字符串房间号

如果您使用的是字符串房间号，务必请将 json 中的 “roomId” 替换成 “strRoomId”: {"strRoomId": "102", "userId": "userB"}

示例代码如下：



```
Json::Value jsonObj;
```

```
jsonObj["strRoomId"] = "102";
jsonObj["userId"] = "userB";
Json::FastWriter writer;
std::string params = writer.write(jsonObj);
trtc.ConnectOtherRoom(params.c_str());
```

参数	描述
param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID。

disconnectOtherRoom

disconnectOtherRoom

退出跨房通话

退出结果会通过 ITRTCCloudCallback 中的 onDisconnectOtherRoom 回调通知给您。

setDefaultStreamRecvMode

setDefaultStreamRecvMode

void setDefaultStreamRecvMode	(bool autoRecvAudio
	bool autoRecvVideo)

设置订阅模式（需要在进入房前设置才能生效）

您可以通过该接口在“自动订阅”和“手动订阅”两种模式下进行切换：
自动订阅：默认模式，用户在进入房间后会立刻接收到该房间中的音视频流，音频会自动播放，视频会自动开始解码（依然需要您通过 startRemoteView 接口绑定渲染控件）。
手动订阅：在用户进入房间后，需要手动调用 startRemoteView 接口才能启动视频流的订阅和解码，需要手动调用 muteRemoteAudio (false) 接口才能启动声音的播放。

在绝大多数场景下，用户进入房间后都会订阅房间中所有主播的音视频流，因此 TRTC 默认采用了自动订阅模式，以求得最佳的“秒开体验”。
如果您的应用场景中每个房间同时会有很多路音视频流在发布，而每个用户只想选择性地订阅其中的 1-2 路，则推荐使用“手动订阅”模式以节省流量费用。

参数	描述

autoRecvAudio	true：自动订阅音频；false：需手动调用 muteRemoteAudio(false) 订阅音频。默认值：true。
autoRecvVideo	true：自动订阅视频；false：需手动调用 startRemoteView 订阅视频。默认值：true。

注意

1. 需要在进入房间前调用该接口进行设置才能生效。
2. 在自动订阅模式下，如果用户在进入房间后没有调用 startRemoteView 订阅视频流，SDK 会自动停止订阅视频流，以便达到节省流量的目的。

createSubCloud

createSubCloud

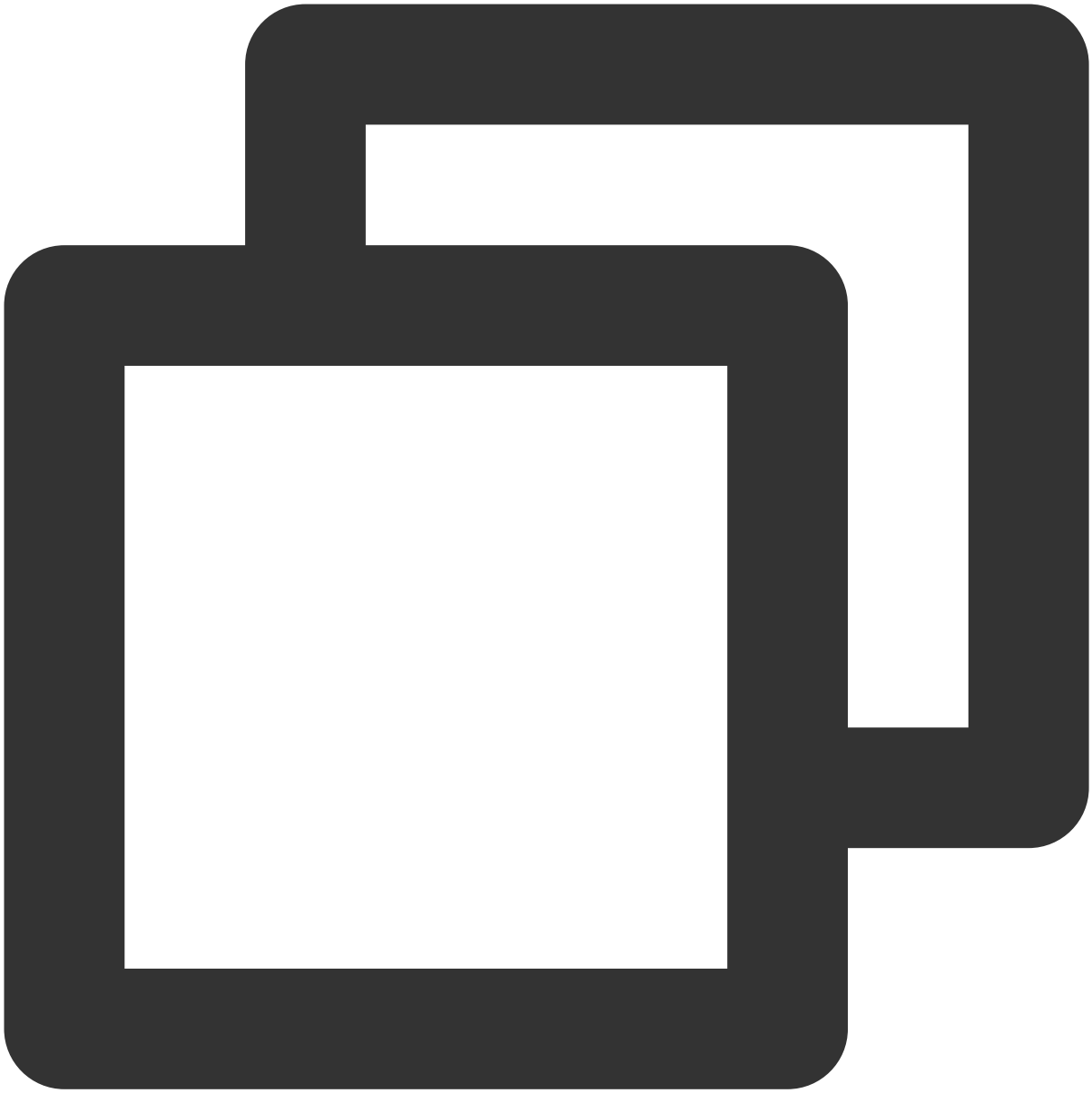
创建子房间实例（用于多房间并发观看）

TRTCCloud 一开始被设计成单例模式，限制了多房间并发观看的能力。
通过调用该接口，您可以创建出多个 TRTCCloud 实例，以便同时进入多个不同的房间观看音视频流。

但需要注意的是，您在多个 TRTCCloud 实例中发布音视频流的能力会受到一定限制。

该功能主要用于在线教育场景中一种被称为“超级小班课”的业务场景中，用于解决“每个 TRTC 的房间中最多只能有 50 人同时发布自己音视频流”的限制。

示例代码如下：



```
//In the small room that needs interaction, enter the room as an anchor and pus
ITRTCCloud *mainCloud = getTRTCShareInstance();
TRTCParams mainParams;
//Fill your params
mainParams.role = TRTCRoleAnchor;
mainCloud->enterRoom(mainParams, TRTCAppSceneLIVE);
//...
mainCloud->startLocalAudio(TRTCAudioQualityDefault);
mainCloud->startLocalPreview(renderView);

//In the large room that only needs to watch, enter the room as an audience and
```

```
ITRTCCloud *subCloud = mainCloud->createSubCloud();
TRTCParams subParams;
//Fill your params
subParams.role = TRTCRoleAudience;
subCloud->enterRoom(subParams, TRTCAppSceneLIVE);
//...
subCloud->startRemoteView(userId, TRTCVideoStreamTypeBig, renderView);
//...
//Exit from new room and release it.
subCloud->exitRoom();
mainCloud->destroySubCloud(subCloud);
```

注意

同一个用户，可以使用同一个 userId 进入多个不同 roomId 的房间。

两台不同的终端设备不可以同时使用同一个 userId 进入同一个 roomId 的房间。

您可以分别为不同实例分别设置 ITRTCCloudCallback 获取各自的事件通知。

同一个用户可以在多个 TRTCCloud 实例中推流，也可以调用子实例中与本地音视频相关的接口。但需要注意：

多实例的音频需要同时为麦克风采集或自定义采集，而且与音频设备相关的接口调用会以最后一次为准；

与摄像头相关的调用会以最后一次为准：startLocalPreview。

返回值说明：

子 TRTCCloud 实例

destroySubCloud

destroySubCloud

void destroySubCloud	(ITRTCCloud *subCloud)

销毁子房间实例

参数	描述
subCloud	子房间实例

updateOtherRoomForwardMode

updateOtherRoomForwardMode

void updateOtherRoomForwardMode	(const char* param)

更改跨房主播在本房间的上行能力

通常情况下，在调用 connectOtherRoom 接口与另一个房间的主播进行跨房通话后，本房间内的所有观众都将收到该主播发布的音视频流。
您可以通过调用该接口，限制跨房主播在本房间内的上行能力，禁止或允许跨房主播发布音频/主路视频/辅路视频，该行为会影响房间内的所有用户。
在禁用跨房主播某种上行能力后，本房间内所有用户将无法收到对应音视频流，且无法再订阅对应的音视频。

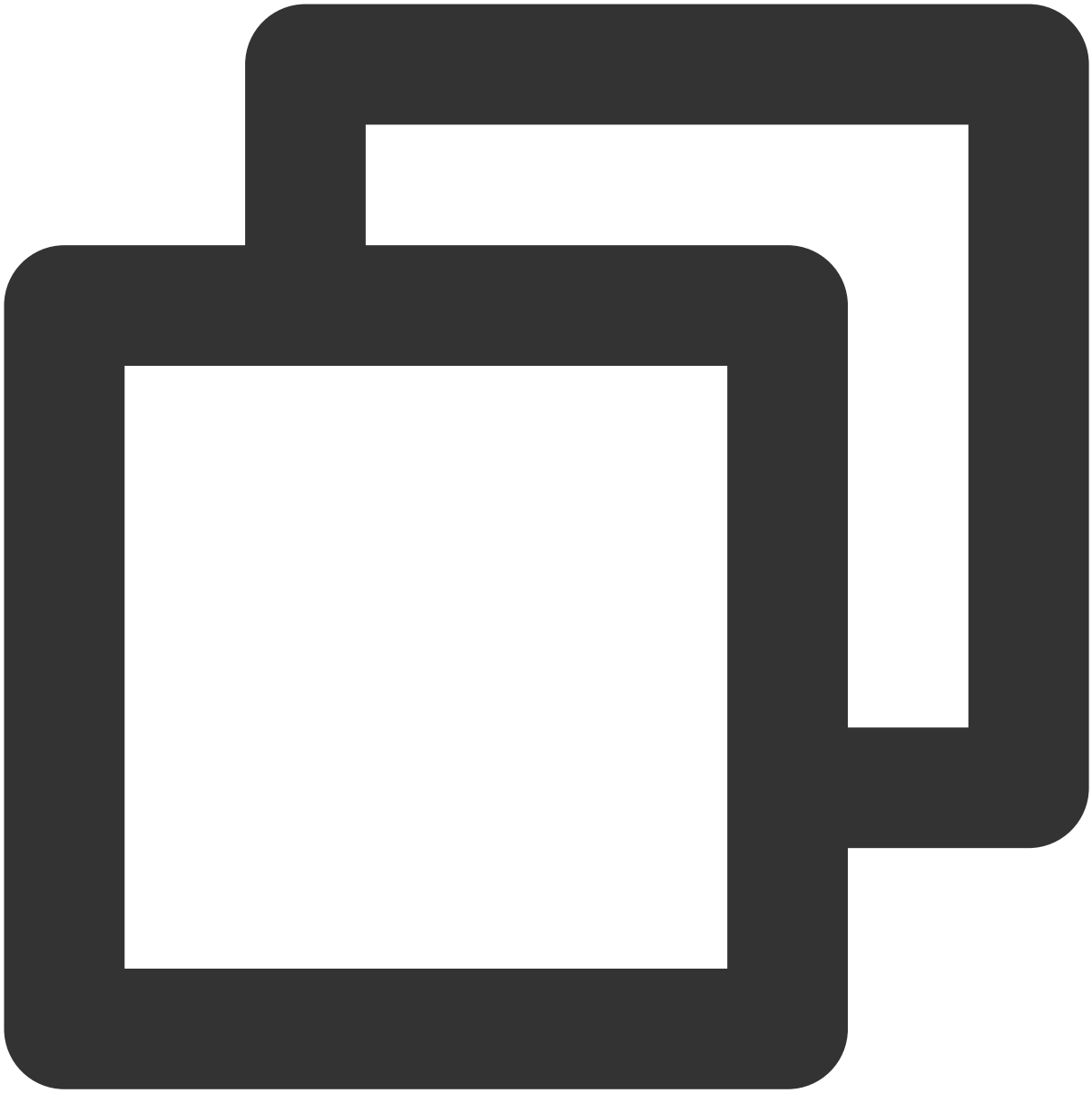
需要注意的是，该接口只能由进行跨房通话的主播调用，且通过该接口设置的限制会因为跨房通话的中断或对应主播退房重置。

该接口的调用结果会通过 ITRTCCloudCallback 中的 onUpdateOtherRoomForwardMode 回调通知给您。

例如：
房间“101”中有主播 A 与观众 B，房间“102”中有主播 C，主播 C 正常发布音视频流。主播 A 通过 connectOtherRoom() 跟主播 C 建立跨房通话。
此时，主播 A 与观众 B 都会收到主播 C 的 onRemoteUserEnterRoom(C) 和 onUserVideoAvailable(C,true)、onUserAudioAvailable(C,true) 这三个事件回调，可以订阅主播 C 的音视频。

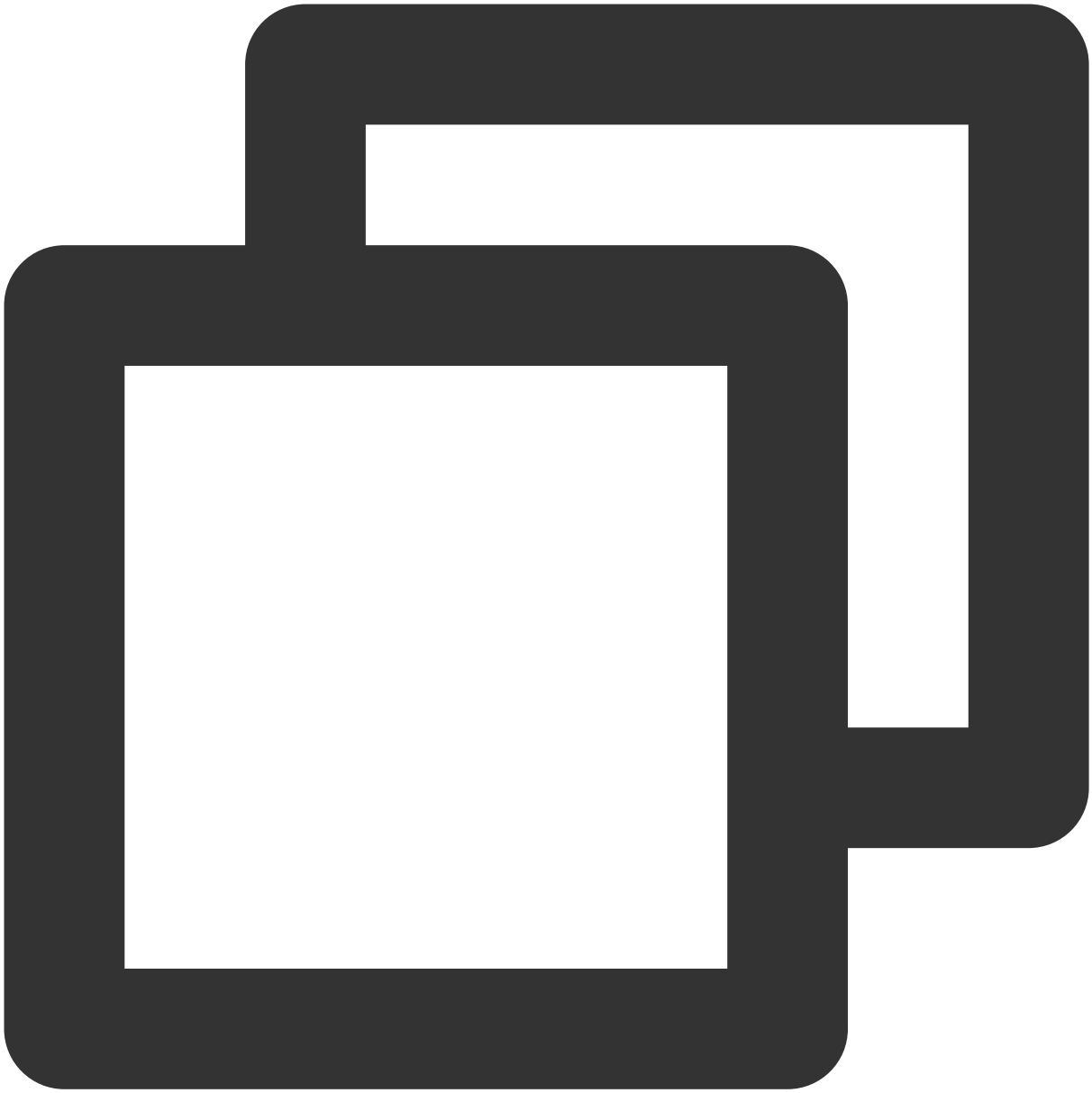
之后，主播 A 调用该接口禁用主播 C 在本房间发布音频的能力。
此后，主播 C 的音频流将无法发布至房间“101”，主播 A 与观众 B 都将收到 onUserAudioAvailable(C,false) 事件回调，且无法再通过 muteRemoteAudio(C,false) 订阅主播 C 的音频。
主播 C 的视频流将不受影响。房间“102”中的其他观众也不受影响，可以正常订阅主播 C 的音频。

该接口的参数考虑到后续扩展字段的兼容性问题，暂时采用了 JSON 格式的参数，示例如下：
情况一：数字房间号



```
{
  "roomId":102,
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

情况二：字符串房间号



```
{
  "strRoomId":"102",
  "userId":"userC",
  "muteAudio":false,
  "muteVideo":true,
  "muteSubStream":false
}
```

参数	描述

param	需要你传入 JSON 格式的字符串参数，roomId 代表数字格式的房间号，strRoomId 代表字符串格式的房间号，userId 代表目标主播的用户 ID，muteAudio、muteVideo、muteSubStream均为可选项，分表代表禁止或允许跨房主播发布音频/主路视频/辅路视频的能力。

startPublishing

startPublishing

void startPublishing	(const char* streamId
	TRTCVideoStreamType streamType)

开始向腾讯云直播 CDN 上发布音视频流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流旁路到直播 CDN 上。您可以通过参数 streamId 设定直播流的 StreamId，从而可以指定该用户的音视频流对应在直播 CDN 上的播放地址。

例如：您可以通过该接口将当前用户的直播流 ID 指定为 user_stream_001，那么该用户音视频流对应的 CDN 播放地址为：
“http://yourdomain/live/user_stream_001.flv”，其中 yourdomain 为您自己备案的播放域名，

您可以在直播控制台 配置您的播放域名，腾讯云不提供默认的播放域名。

您也可以在设置 enterRoom 的参数 TRTCParams 时指定 streamId, 而且我们更推荐您采用这种方案。

参数	描述
streamId	自定义流 ID。
streamType	仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub。

注意

您需要提前在 实时音视频控制台 中的功能配置页面上开启“启用旁路推流”才能生效。
若您选择“指定流旁路”，则您可以通过该接口将对应音视频流推送到腾讯云 CDN 且指定为填写的流 ID。
若您选择“全局自动旁路”，则您可以通过该接口调整默认的流 ID。

stopPublishing

stopPublishing

停止向腾讯云直播 CDN 上发布音视频流

startPublishCDNStream

startPublishCDNStream

void startPublishCDNStream	(const TRTCPublishCDNParam& param)

开始向非腾讯云 CDN 上发布音视频流

该接口跟 startPublishing 功能类似，不同之处在于，startPublishing 仅支持向腾讯云的 CDN 发布，而本接口支持向非腾讯云的直播 CDN 上转推音视频流。

参数	描述
param	CDN 转推参数，详情请参见 TRTCPublishCDNParam

注意

使用 startPublishing 接口向腾讯云的直播 CDN 上发布音视频流不会收取额外费用。
使用 startPublishCDNStream 接口向非腾讯云的直播 CDN 上发布音视频流，需要收取额外的转推带宽费用。

stopPublishCDNStream

stopPublishCDNStream

停止向非腾讯云 CDN 上发布音视频流

setMixTranscodingConfig

setMixTranscodingConfig

void setMixTranscodingConfig	(TRTCTranscodingConfig* config)

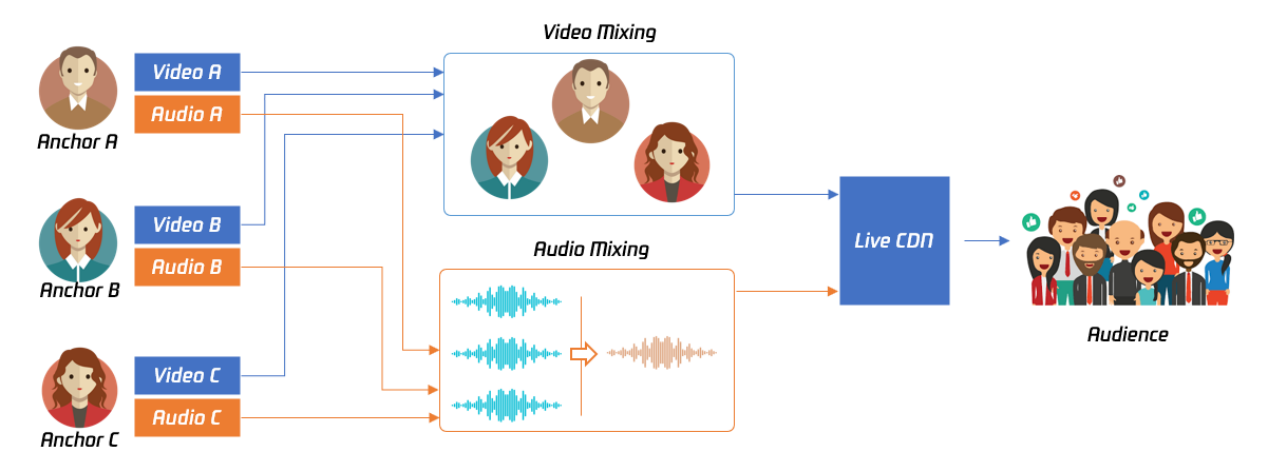
设置云端混流的排版布局和转码参数

在一个直播间中可能同时会有多个主播发布自己的音视频流，但对于直播 CDN 上的观众而言，只需要观看一条 HTTP-FLV 或 HLS 格式的视频流即可。
当您调用本接口函数时，SDK 会向腾讯云的 TRTC 混流服务器发送一条指令，混流服务器会将房间里的多路音视频流混合成一路。

版权所有：腾讯云计算（北京）有限责任公司

您可以通过 TRTCTranscodingConfig 参数来调整每一路画面的排版布局，也可以设置混合后的音视频流的各项编码参数。

参考文档：云端混流转码。



参数	描述
config	如果 config 不为空，则开启云端混流，如果 config 为空则停止云端混流。详情请参见 TRTCTranscodingConfig 。

注意

关于云端混流的注意事项：

混流转码为收费功能，调用接口将产生云端混流转码费用，详见 云端混流转码计费说明 。

调用该接口的用户，如果没设定 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到当前用户所对应的音视频流上，即 A + B => A。

调用该接口的用户，如果设定了 config 参数中的 streamId 字段，TRTC 会将房间中的多路画面混合到您指定的 streamId 上，即 A + B => streamId。

请注意，若您还在房间中且不再需要混流，请务必再次调用本接口并将 config 设置为空以进行取消，不及时取消混流可能会引起不必要的计费损失。

请放心，当您退房时 TRTC 会自动取消混流状态。

startPublishMediaStream

startPublishMediaStream

void startPublishMediaStream	(TRTCPublishTarget * target

	TRTCStreamEncoderParam * params
	TRTCStreamMixingConfig * config)

开始发布媒体流

该接口会向 TRTC 服务器发送指令，要求其将当前用户的音视频流转推/转码到直播 CDN 或者回推到 TRTC 房间中您可以通过 TRTCPublishTarget 配置中的 TRTCPublishMode 指定具体的发布模式

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。

注意

1. SDK 会通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）。
2. 同一个任务（TRTCPublishMode 与 TRTCPublishCdnUrl 均相同）仅支持启动一次。若您后续需要更新或者停止该项任务，需要记录并使用返回的 taskId，通过 updatePublishMediaStream 或者 stopPublishMediaStream 来操作。
3. target 支持同时配置多个 CDN URL（最多同时 10 个）。若您的同一个转推/转码任务需要发布至多路 CDN，则仅需要在 target 中配置多个 CDN URL 即可。同一个转码任务即使有多个转推地址，对应的转码计费仍只收取一份。
4. 使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。一种推荐的方案是 URL 中使用 “sdkappid_roomid_userid_main” 作为区分标识，这种命名方式容易辨认且不会在您的多个应用中发生冲突。

updatePublishMediaStream

updatePublishMediaStream

void updatePublishMediaStream	(const char* taskId
	TRTCPublishTarget * target
	TRTCStreamEncoderParam * params
	TRTCStreamMixingConfig * config)

更新发布媒体流

该接口会向 TRTC 服务器发送指令，更新通过 startPublishMediaStream 启动的媒体流

参数	描述
config	媒体流转码配置参数。具体配置参考 TRTCStreamMixingConfig。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码配置参数。转推模式下则无效。
params	媒体流编码输出参数，具体配置参考 TRTCStreamEncoderParam。转码和回推到 TRTC 房间中时为必填项，您需要指定您预期的转码输出参数。在转推时，为了更好的转推稳定性和 CDN 兼容性，也建议您进行配置。
target	媒体流发布的目标地址，具体配置参考 TRTCPublishTarget。支持转推/转码到腾讯或者第三方 CDN，也支持转码回推到 TRTC 房间中。
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 您可以通过本接口来更新发布的 CDN URL（支持增删，最多同时 10 个），但您使用时需要注意不要多个任务同时往相同的 URL 地址推送，以免引起异常推流状态。
2. 您可以通过 taskId 来更新调整转推/转码任务。例如在 pk 业务中，您可以先通过 startPublishMediaStream 发起转推，接着在主播发起 pk 时，通过 taskId 和本接口将转推更新为转码任务。此时，CDN 播放将连续并且不会发生断流（您需要保持媒体流编码输出参数 param 一致）。
3. 同一个任务不支持纯音频、音视频、纯视频之间的切换。

stopPublishMediaStream

stopPublishMediaStream

void stopPublishMediaStream	(const char* taskId)

停止发布媒体流

该接口会向 TRTC 服务器发送指令，停止通过 startPublishMediaStream 启动的媒体流

参数	描述
taskId	通过回调 onStartPublishMediaStream 带给您后台启动的任务标识（即 taskId）

注意

1. 若您的业务后台并没有保存该 taskId，在您的主播异常退房重进后，如果您需要重新获取 taskId，您可以再次调用 startPublishMediaStream 启动任务。此时 TRTC 后台会返回任务启动失败，同时带给您上一次启动的 taskId

版权所有：腾讯云计算（北京）有限责任公司

2. 若 taskId 填空字符串，将会停止该用户所有通过 startPublishMediaStream 启动的媒体流，如果您只启动了一个媒体流或者想停止所有通过您启动的媒体流，推荐使用这种方式。

startLocalPreview

startLocalPreview

void startLocalPreview	(bool frontCamera
	TXView view)

开启本地摄像头的预览画面（移动端）

在 enterRoom 之前调用此函数，SDK 只会开启摄像头，并一直等到您调用 enterRoom 之后才开始推流。
在 enterRoom 之后调用此函数，SDK 会开启摄像头并自动开始视频推流。当开始渲染首帧摄像头画面时，您会收到 ITRTCCloudCallback 中的 onCameraDidReady 回调通知。

参数	描述
frontCamera	true：前置摄像头；false：后置摄像头。
view	承载视频画面的控件。

注意

如果希望开播前预览摄像头画面并通过 BeautyManager 调节美颜参数，您可以：
方案一：在调用 enterRoom 之前调用 startLocalPreview。
方案二：在调用 enterRoom 之后调用 startLocalPreview + muteLocalVideo(true)。

startLocalPreview

startLocalPreview

void startLocalPreview	(TXView view)

开启本地摄像头的预览画面（桌面端）

在调用该接口之前，您可以先调用 setCurrentCameraDevice 选择使用 Mac 自带摄像头或外接摄像头。
在 enterRoom 之前调用此函数，SDK 只会开启摄像头，并一直等到您调用 enterRoom 之后才开始推流。
在 enterRoom 之后调用此函数，SDK 会开启摄像头并自动开始视频推流。
当开始渲染首帧摄像头画面时，您会收到 TRTCCloudDelegate 中的 onCameraDidReady 回调通知。

参数	描述

view	承载视频画面的控件。

注意

如果希望开播前预览摄像头画面并通过 BeautyManager 调节美颜参数，您可以：

方案一：在调用 enterRoom 之前调用 startLocalPreview。

方案二：在调用 enterRoom 之后调用 startLocalPreview + muteLocalVideo(true)。

updateLocalView

updateLocalView

void updateLocalView	(TXView view)

更新本地摄像头的预览画面

stopLocalPreview

stopLocalPreview

停止摄像头预览

muteLocalVideo

muteLocalVideo

void muteLocalVideo	(TRTCVideoStreamType streamType
	bool mute)

暂停/恢复发布本地的视频流

该接口可以暂停（或恢复）发布本地的视频画面，暂停之后，同一房间中的其他用户将无法继续看到自己画面。

该接口在指定 TRTCVideoStreamTypeBig 时等效于 start/stopLocalPreview 这两个接口，但具有更好的响应速度。

因为 start/stopLocalPreview 需要打开和关闭摄像头，而打开和关闭摄像头都是硬件设备相关的操作，非常耗时。

相比之下，muteLocalVideo 只需要在软件层面对数据流进行暂停或者放行即可，因此效率更高，也更适合需要频繁打开关闭的场景。

当暂停/恢复发布指定 TRTCVideoStreamTypeBig 后，同一房间中的其他用户将会收到 onUserVideoAvailable 回调通知。
当暂停/恢复发布指定 TRTCVideoStreamTypeSub 后，同一房间中的其他用户将会收到 onUserSubStreamAvailable 回调通知。

参数	描述
mute	true：暂停；false：恢复。
streamType	要暂停/恢复的视频流类型（仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub）。

setVideoMuteImage

setVideoMuteImage

void setVideoMuteImage	(TRTCImageBuffer* image
	int fps)

设置本地画面被暂停期间的替代图片

当您调用 muteLocalVideo(true) 暂停本地画面时，您可以通过调用本接口设置一张替代图片，设置后，房间中的其他用户会看到这张替代图片，而不是黑屏画面。

参数	描述
fps	设置替代图片帧率，最小值为5，最大值为10，默认5。
image	设置替代图片，空值代表在 muteLocalVideo 之后不再发送视频流数据，默认值为空。

startRemoteView

startRemoteView

void startRemoteView	(const char* userId
	TRTCVideoStreamType streamType
	TXView view)

订阅远端用户的视频流，并绑定视频渲染控件

调用该接口可以让 SDK 拉取指定 userid 的视频流，并渲染到参数 view 指定的渲染控件上。您可以通过 setRemoteRenderParams 设置画面的显示模式。
如果您已经知道房间中有视频流的用户的 userid，可以直接调用 startRemoteView 订阅该用户的画面。
如果您不知道房间中有哪些用户在发布视频，您可以在 enterRoom 之后等待来自 onUserVideoAvailable 的通知。

调用本接口只是启动视频流的拉取，此时画面还需要加载和缓冲，当缓冲完毕后您会收到来自 onFirstVideoFrame 的通知。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall（需要远端用户通过 enableSmallVideoStream 开启双路编码后才有效果）。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。
view	用于承载视频画面的渲染控件。

注意

注意几点规则需要您关注：
1. SDK 支持同时观看某 userid 的大画面和辅路画面，或者同时观看某 userid 的小画面和辅路画面，但不支持同时观看大画面和小画面。
2. 只有当指定的 userid 通过 enableSmallVideoStream 开启双路编码后，才能观看该用户的小画面。
3. 当指定的 userid 的小画面不存在时，SDK 默认切换到该用户的大画面。

updateRemoteView

updateRemoteView

void updateRemoteView	(const char* userId
	TRTCVideoStreamType streamType
	TXView view)

更新远端用户的视频渲染控件

该接口可用于更新远端视频画面的渲染控件，常被用于切换显示区域的交互场景中。

参数	描述

streamType	要设置预览窗口的流类型（仅支持 TRTCVideoStreamTypeBig 和 TRTCVideoStreamTypeSub）。
userId	指定远端用户的 ID。
view	承载视频画面的控件。

stopRemoteView

stopRemoteView

void stopRemoteView	(const char* userId
	TRTCVideoStreamType streamType)

停止订阅远端用户的视频流，并释放渲染控件

调用此接口会让 SDK 停止接收该用户的视频流，并释放该路视频流的解码和渲染资源。

参数	描述
streamType	指定要观看 userId 的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。

stopAllRemoteView

stopAllRemoteView

停止订阅所有远端用户的视频流，并释放全部渲染资源

调用此接口会让 SDK 停止接收所有来自远端的视频流，并释放全部的解码和渲染资源。

注意

如果当前有正在显示的辅路画面（屏幕分享）也会一并被停止。

muteRemoteVideoStream

muteRemoteVideoStream

void muteRemoteVideoStream	(const char* userId
	TRTCVideoStreamType streamType
	bool mute)

暂停/恢复订阅远端用户的视频流

该接口仅暂停/恢复接收指定用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。
streamType	要暂停/恢复的视频流类型。 高清大画面：TRTCVideoStreamTypeBig。 低清小画面：TRTCVideoStreamTypeSmall。 辅流画面（常用于屏幕分享）：TRTCVideoStreamTypeSub。
userId	指定远端用户的 ID。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。

调用该接口暂停接收指定用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(false) 或 muteAllRemoteVideoStreams(false) 才能恢复。

muteAllRemoteVideoStreams

muteAllRemoteVideoStreams

void muteAllRemoteVideoStreams	(bool mute)

暂停/恢复订阅所有远端用户的视频流

该接口仅暂停/恢复接收所有用户的视频流，但并不释放显示资源，视频画面会被冻屏在接口调用时的最后一帧。

参数	描述
mute	是否暂停接收。

注意

该接口支持您在进入房间（enterRoom）前调用，暂停状态在退出房间（exitRoom）之后会被重置。

调用该接口暂停接收所有用户的视频流后，只调用 startRemoteView 接口无法播放指定用户的视频，需要调用 muteRemoteVideoStream(false) 或 muteAllRemoteVideoStreams(false) 才能恢复。

setVideoEncoderParam

setVideoEncoderParam

void setVideoEncoderParam	(const TRTCVideoEncParam& param)

设置视频编码器的编码参数

该设置能够决定远端用户看到的画面质量，同时也能决定云端录制出的视频文件的画面质量。

参数	描述
param	用于设置视频编码器的相关参数，详情请参见 TRTCVideoEncParam。

注意

从v11.5版本开始，编码输出分辨率会按照宽8高2字节对齐，并且是向下调整，eg:输入分辨率540x960，实际编码输出分辨率536x960。

setNetworkQosParam

setNetworkQosParam

void setNetworkQosParam	(const TRTCNetworkQosParam& param)

设置网络质量控制的相关参数

该设置决定在差网络环境下的质量调控策略，如“画质优先”或“流畅优先”等策略。

参数	描述
param	用于设置网络质量控制的相关参数，详情请参见 TRTCNetworkQosParam。

setLocalRenderParams

setLocalRenderParams

void setLocalRenderParams	(const TRTCRenderParams &params)

版权所有：腾讯云计算（北京）有限责任公司

设置本地画面的渲染参数

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述
params	画面渲染参数，详情请参见 TRTCRenderParams。

setRemoteRenderParams

setRemoteRenderParams

void setRemoteRenderParams	(const char* userId
	TRTCVideoStreamType streamType
	const TRTCRenderParams &params)

设置远端画面的渲染模式

可设置的参数包括有：画面的旋转角度、填充模式以及左右镜像等。

参数	描述
params	画面渲染参数，详情请参见 TRTCRenderParams。
streamType	可以设置为主路画面（TRTCVideoStreamTypeBig）或辅路画面（TRTCVideoStreamTypeSub）。
userId	指定远端用户的 ID。

enableSmallVideoStream

enableSmallVideoStream

void enableSmallVideoStream	(bool enable
	const TRTCVideoEncParam& smallVideoEncParam)

开启大小画面双路编码模式

开启双路编码模式后，当前用户的编码器会同时输出【高清大画面】和【低清小画面】两路视频流（但只有一路音频流）。

版权所有：腾讯云计算（北京）有限责任公司

如此以来，房间中的其他用户就可以根据自身的网络情况或屏幕大小选择订阅【高清大画面】或是【低清小画面】。

参数	描述
enable	是否开启小画面编码，默认值：false。
smallVideoEncParam	小流的视频参数。

注意

双路编码开启后，会消耗更多的 CPU 和 网络带宽，所以 Mac、Windows 或者高性能 Pad 可以考虑开启，不建议手机端开启。

返回值说明：

0：成功；-1：当前大画面已被设置为较低画质，开启双路编码已无必要。

setRemoteVideoStreamType

setRemoteVideoStreamType

void setRemoteVideoStreamType	(const char* userId
	TRTCVideoStreamType streamType)

切换指定远端用户的大小画面

当房间中某个主播开启了双路编码之后，房间中其他用户通过 startRemoteView 订阅到的画面默认会是【高清大画面】。

您可以通过此接口选定希望订阅的画面是大画面还是小画面，该接口在 startRemoteView 之前和之后调用均可生效。

参数	描述
streamType	视频流类型，即选择看大画面还是小画面，默认为大画面。
userId	指定远端用户的 ID。

注意

此功能需要目标用户已经通过 enableSmallVideoStream 提前开启了双路编码模式，否则此调用无实际效果。

snapshotVideo

snapshotVideo

void snapshotVideo	(const char* userId
	TRTCVideoStreamType streamType
	TRTCSnapshotSourceType sourceType)

视频画面截图

您可以通过本接口截取本地的视频画面，远端用户的主路画面以及远端用户的辅路（屏幕分享）画面。

参数	描述
sourceType	画面来源，可选择截取视频流画面（TRTCSnapshotSourceTypeStream）、视频渲染画面（TRTCSnapshotSourceTypeView）或 采集画面（TRTCSnapshotSourceTypeCapture），采集画面截图更清晰。
streamType	视频流类型，可选择截取主路画面（TRTCVideoStreamTypeBig，常用于摄像头）或辅路画面（TRTCVideoStreamTypeSub，常用于屏幕分享）。
userId	用户 ID，如指定空置表示截取本地的视频画面。

注意

Windows 平台目前仅支持截取 TRTCSnapshotSourceTypeStream 来源的视频画面。

setGravitySensorAdaptiveMode

setGravitySensorAdaptiveMode

void setGravitySensorAdaptiveMode	(TRTCGravitySensorAdaptiveMode mode)

设置重力感应的适配模式（11.7 及以上版本）

开启重力感应后，如果采集端的设备发生旋转，采集端和观众端的画面都会进行相应地渲染以确保视野中的画面始终朝上。

只在sdk内部摄像头采集场景生效，并且只在移动端生效。

1. 该接口仅对采集端起作用，如果只是观看房间中的画面，开启此接口是无效的

2. 当采集端设备发生 90 度或 270 度旋转时，采集端或者观众看到的画面可能会被裁剪以保持比例的协调

参数	描述
mode	重力感应模式，详情请参见 TRTCGravitySensorAdaptiveMode_Disable、TRTCGravitySensorAdaptiveMode_FillByCenterCrop

	和 TRTCGravitySensorAdaptiveMode_FitWithBlackBorder 默认值：TRTCGravitySensorAdaptiveMode_Disable。

startLocalAudio

startLocalAudio

void startLocalAudio	(TRTCAudioQuality quality)

开启本地音频的采集和发布

SDK 默认不开启麦克风，当用户需要发布本地音频时，需要调用该接口开启麦克风采集，并将音频编码并发布到当前的房间中。

开启本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, true) 的通知。

参数	描述
quality	声音音质 TRTCAudioQualitySpeech，流畅：采样率：16k；单声道；音频裸码率：16kbps；适合语音通话为主的场景，比如在线会议，语音通话。 TRTCAudioQualityDefault，默认：采样率：48k；单声道；音频裸码率：50kbps；SDK 默认的音频质量，如无特殊需求推荐选择之。 TRTCAudioQualityMusic，高音质：采样率：48k；双声道 + 全频带；音频裸码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

注意

该函数会检查麦克风的使用权限，如果当前 App 没有麦克风权限，SDK 会自动向用户申请麦克风使用权限。

stopLocalAudio

stopLocalAudio

停止本地音频的采集和发布

停止本地音频的采集和发布后，房间中的其他用户会收到 onUserAudioAvailable(userId, false) 的通知。

muteLocalAudio

muteLocalAudio

void muteLocalAudio	(bool mute)

暂停/恢复发布本地的音频流

当您暂停发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, false) 的通知。
当您恢复发布本地音频流之后，房间中的其他用户会收到 onUserAudioAvailable(userId, true) 的通知。

与 stopLocalAudio 的不同之处在于，muteLocalAudio(true) 并不会释放麦克风权限，而是继续发送码率极低的静音包。
这对于需要云端录制的场景非常适用，因为 MP4 等格式的视频文件，对于音频数据的连续性要求很高，使用 stopLocalAudio 会导致录制出的 MP4 文件不易播放。
因此在对录制文件的质量要求较高的场景中，建议选择 muteLocalAudio 而不建议使用 stopLocalAudio。

参数	描述
mute	true：静音；false：恢复。

muteRemoteAudio

muteRemoteAudio

void muteRemoteAudio	(const char* userId
	bool mute)

暂停/恢复播放远端的音频流

当您静音某用户的远端音频时，SDK 会停止播放指定用户的声音，同时也会停止拉取该用户的音频数据。

参数	描述
mute	true：静音；false：取消静音。
userId	用于指定远端用户的 ID。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 false。

muteAllRemoteAudio

muteAllRemoteAudio

void muteAllRemoteAudio	(bool mute)

暂停/恢复播放所有远端用户的音频流

当您静音所有用户的远端音频时，SDK 会停止播放所有来自远端的音频流，同时也会停止拉取所有用户的音频数据。

参数	描述
mute	true：静音；false：取消静音。

注意

在进入房间（enterRoom）之前或之后调用本接口均生效，静音状态在退出房间（exitRoom）之后会被重置为 false。

setRemoteAudioVolume

setRemoteAudioVolume

void setRemoteAudioVolume	(const char *userId
	int volume)

设定某一个远端用户的声音播放音量

您可以通过 setRemoteAudioVolume(userId, 0) 将某一个远端用户的声音静音。

参数	描述
userId	用于指定远端用户的 ID。
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setAudioCaptureVolume

setAudioCaptureVolume

void setAudioCaptureVolume	(int volume)

版权所有：腾讯云计算（北京）有限责任公司

设定本地音频的采集音量

参数	描述
volume	音量大小，取值范围为0 - 100；默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioCaptureVolume

getAudioCaptureVolume

获取本地音频的采集音量

setAudioPlayoutVolume

setAudioPlayoutVolume

void setAudioPlayoutVolume	(int volume)

设定远端音频的播放音量

该接口会控制 SDK 最终交给系统播放的声音音量，调节效果会影响到本地音频录制文件的音量大小，但不会影响到耳返的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

getAudioPlayoutVolume

getAudioPlayoutVolume

获取远端音频的播放音量

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(bool enable
	const TRTCAudioVolumeEvaluateParams& params)

启用音量大小提示

开启此功能后，SDK 会在 ITRTCCloudCallback 中的 onUserVoiceVolume 回调中反馈本地或远端用户的音频音量评估信息，包括音量大小、人声检测、音频频谱等。

参数	描述
enable	是否启用音量提示，默认为关闭状态。
params	音量评估等相关参数，请参见 TRTCAudioVolumeEvaluateParams。

startAudioRecording

startAudioRecording

int startAudioRecording	(const TRTCAudioRecordingParams& param)

开始录音

当您调用该接口后， SDK 会将本地和远端的所有音频（包括本地音频，远端音频，背景音乐和音效等）混合并录制到一个本地文件中。

该接口在进入房间前后调用均可生效，如果录制任务在退出房间前尚未通过 stopAudioRecording 停止，则退出房间后录制任务会自动被停止。

本次录制的启动、完成状态会通过本地录制相关回调进行通知。参见 TRTCCloud 相关回调。

参数	描述
param	录音参数，请参见 TRTCAudioRecordingParams。

注意

自 v11.5 版本，音频录制的状态结果由返回值统一调整为异步回调进行通知。参见 TRTCCloud 相关回调。

返回值说明：

0：成功；-1：录音已开始；-2：文件或目录创建失败；-3：后缀指定的音频格式不支持。

stopAudioRecording

stopAudioRecording

停止录音

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

startLocalRecording

startLocalRecording

void startLocalRecording	(const TRTCLocalRecordingParams& params)

开启本地媒体录制

开启后把直播过程中的音视频内容录制到本地的一个文件中。

参数	描述
params	录制参数，请参见 TRTCLocalRecordingParams。

stopLocalRecording

stopLocalRecording

停止本地媒体录制

如果录制任务在退出房间前尚未通过本接口停止，则退出房间后录音任务会自动被停止。

setRemoteAudioParallelParams

setRemoteAudioParallelParams

void setRemoteAudioParallelParams	(const TRTCAudioParallelParams& params)

设置远端音频流智能并发播放策略

设置远端音频流智能并发播放策略，适用于上麦人数比较多的场景。

参数	描述
params	音频并发参数，请参见 TRTCAudioParallelParams。

版权所有：腾讯云计算（北京）有限责任公司

enable3DSpatialAudioEffect

enable3DSpatialAudioEffect

void enable3DSpatialAudioEffect	(bool enabled)

启用 3D 音效

启用 3D 音效。注意需使用流畅音质 TRTCAudioQualitySpeech 或默认音质 TRTCAudioQualityDefault。

参数	描述
enabled	是否启用 3D 音效，默认为关闭状态。

updateSelf3DSpatialPosition

updateSelf3DSpatialPosition

void updateSelf3DSpatialPosition	(int position[3]
	float axisForward[3]
	float axisRight[3]
	float axisUp[3])

设置 3D 音效中自身坐标及朝向信息

更新自身在世界坐标系中的位置和朝向， SDK 会根据该方法参数计算自身和远端用户之间的相对位置，进而渲染出空间音效。注意各参数应分别传入长度为 3 的数组。

参数	描述
axisForward	自身坐标系前轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisRight	自身坐标系右轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
axisUp	自身坐标系上轴在世界坐标系中的单位向量，三个值依次表示前、右、上坐标值。
position	自身在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。

注意

请适当限制调用频率，推荐两次坐标设置至少间隔 100ms。

版权所有：腾讯云计算（北京）有限责任公司

updateRemote3DSpatialPosition

updateRemote3DSpatialPosition

void updateRemote3DSpatialPosition	(const char* userId
	int position[3])

设置 3D 音效中远端用户坐标信息

更新远端用户在世界坐标系中的位置，SDK 会根据自身和远端用户之间的相对位置，进而渲染出空间音效。注意参数为长度等于 3 的数组。

参数	描述
position	该远端用户在世界坐标系中的坐标，三个值依次表示前、右、上坐标值。
userId	指定远端用户的 ID。

注意

请适当限制调用频率，推荐同一远端用户两次坐标设置至少间隔 100ms。

set3DSpatialReceivingRange

set3DSpatialReceivingRange

void set3DSpatialReceivingRange	(const char* userId
	int range)

设置指定用户所发出声音的可被接收范围

设置该范围大小之后，该指定用户的声音将在该范围内可被听见，超出该范围将被衰减为 0。

参数	描述
range	声音最大可被接收范围。
userId	指定远端用户的 ID。

getDeviceManager

getDeviceManager

获取设备管理类（TXDeviceManager）

setBeautyStyle

setBeautyStyle

void setBeautyStyle	(TRTCBeautyStyle style
	uint32_t beautyLevel
	uint32_t whitenessLevel
	uint32_t ruddinessLevel)

设置美颜、美白、红润等特效

SDK 内部集成了两套风格不同的磨皮算法：
-“光滑”：算法比较激进，磨皮效果比较明显，适用于秀场直播。
-“自然”：算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。

参数	描述
beautyLevel	美颜级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显。
ruddinessLevel	红润级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显。
style	磨皮算法，有“光滑”和“自然”两种算法。
whitenessLevel	美白级别，取值范围0 - 9，0表示关闭，1 - 9值越大，效果越明显。

setWaterMark

setWaterMark

void setWaterMark	(TRTCVideoStreamType streamType
	const char* srcData
	TRTCWaterMarkSrcType srcType
	uint32_t nWidth
	uint32_t nHeight
	float xOffset

	float yOffset
	float fWidthRatio
	bool isVisibleOnLocalPreview = false)

添加水印

水印的位置是通过 xOffset, yOffset, fWidthRatio 来指定的。

xOffset：水印的坐标，取值范围为0 - 1的浮点数。

yOffset：水印的坐标，取值范围为0 - 1的浮点数。

fWidthRatio：水印的大小比例，取值范围为0 - 1的浮点数。

参数	描述
fWidthRatio	水印显示的宽度占画面宽度比例（水印按该参数等比例缩放显示）。
isVisibleOnLocalPreview	true：本地预览显示水印；false：本地预览不显示水印，仅在win/mac下生效。
nHeight	水印图片像素高度（源数据为文件路径时忽略该参数）。
nWidth	水印图片像素宽度（源数据为文件路径时忽略该参数）。
srcData	水印图片源数据（传 nullptr 表示去掉水印）。
srcType	水印图片源数据类型。
streamType	要设置水印的流类型（TRTCVideoStreamTypeBig、TRTCVideoStreamTypeSub）。
xOffset	水印显示的左上角 x 轴偏移。
yOffset	水印显示的左上角 y 轴偏移。

注意

本接口只支持给主路视频添加图片水印。

getAudioEffectManager

getAudioEffectManager

获取音效管理类（TXAudioEffectManager）

TXAudioEffectManager 是音效管理接口，您可以通过该接口实现如下功能：

背景音乐：支持在线音乐和本地音乐，支持变速、变调等特效、支持原生和伴奏并播放和循环播放。
耳机耳返：麦克风捕捉的声音实时通过耳机播放，常用于音乐直播。
混响效果：KTV、小房间、大会堂、低沉、洪亮...。
变声特效：萝莉、大叔、重金属...。
短音效：鼓掌声、欢笑声等简短的音效文件（对于小于10秒的文件，请将 isShortFile 参数设置为 true）。

startSystemAudioLoopback

startSystemAudioLoopback

void startSystemAudioLoopback	(const char* deviceName = nullptr)

开启系统声音采集（iOS 端暂未支持）

该接口会从电脑的声卡中采集音频数据，并将其混入到 SDK 当前的音频数据流中，从而使房间中的其他用户也能听到主播的电脑所播放出的声音。
在线教育场景中，老师可以使用此功能让 SDK 采集教学影片中的声音，并广播给同房间中的学生。
音乐直播场景中，主播可以使用此功能让 SDK 采集音乐播放器中的音乐，从而为自己的直播间增加背景音乐。

参数	描述
deviceName	您可以指定该参数为空值（nullptr），代表让 SDK 采集整个系统的声音。

注意

在 Windows 平台下，您也可以将参数 deviceName 设置为某个应用程序的可执行文件（如 QQMuisc.exe）的绝对路径，此时 SDK 只会采集该应用程序的声音（仅支持 32 位版本的 SDK）。
您也可以指定该参数为某个扬声器设备的名称来采集特定扬声器声音（通过接口 TXDeviceManager 中的 getDevicesList 接口，可以获取类型为 TXMediaDeviceTypeSpeaker 的扬声器设备）。

stopSystemAudioLoopback

stopSystemAudioLoopback

停止系统声音采集（iOS 端暂未支持）

setSystemAudioLoopbackVolume

setSystemAudioLoopbackVolume

void setSystemAudioLoopbackVolume	(uint32_t volume)

设置系统声音的采集音量

参数	描述
volume	设置的音量大小，范围是：[0 ~ 150]，默认值为100。

startScreenCapture

startScreenCapture

void startScreenCapture	(TXView view
	TRTCVideoStreamType streamType
	TRTCVideoEncParam* encParam)

启动屏幕分享

该接口可以抓取整个屏幕的内容，或抓取您指定的某个应用的窗口内容，并将其分享给同房间中的其他用户。

参数	描述
encParam	屏幕分享的画面编码参数，SDK 会优先使用您通过此接口设置的编码参数： 如果您设置 encParam 为空值，且您已通过 setSubStreamEncoderParam 设置过辅路视频编码参数，SDK 将使用您设置过的辅路编码参数进行屏幕分享。 如果您设置 encParam 为空值，且您未通过 setSubStreamEncoderParam 设置过辅路视频编码参数，SDK 将自动选择一个最佳的编码参数进行屏幕分享。
streamType	屏幕分享使用的线路，可以设置为主路（TRTCVideoStreamTypeBig）或者辅路（TRTCVideoStreamTypeSub），推荐使用辅路。
view	渲染控件所在的父控件，可以设置为空值，表示不显示屏幕分享的预览效果。

注意

1. 同一个用户同时最多只能发布一路主路（TRTCVideoStreamTypeBig）画面和一路辅路（TRTCVideoStreamTypeSub）画面。
2. 默认情况下，屏幕分享使用辅路画面。如果使用主路做屏幕分享，您需要提前停止摄像头采集（stopLocalPreview）以避免相互冲突。
3. 同一个房间中同时只能有一个用户使用辅路做屏幕分享，也就是说，同一个房间中同时只允许一个用户开启辅路。

版权所有：腾讯云计算（北京）有限责任公司

4. 当房间中已经有其他用户在使用辅路分享屏幕时，此时调用该接口会收到来自 ITRTCCloudCallback 的 onError(ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO) 回调。

stopScreenCapture

stopScreenCapture

停止屏幕分享

pauseScreenCapture

pauseScreenCapture

暂停屏幕分享

注意

从v11.5版本开始，暂停屏幕采集会使用最后一帧按照1fps帧率输出。

resumeScreenCapture

resumeScreenCapture

恢复屏幕分享

getScreenCaptureSources

getScreenCaptureSources

ITRTCScreenCaptureSourceList* getScreenCaptureSources	(const SIZE &thumbnailSize
	const SIZE &iconSize)

枚举可分享的屏幕和窗口（该接口仅支持桌面系统）

当您在对接桌面端系统的屏幕分享功能时，一般都需要展示一个选择分享目标的界面，这样用户能够使用这个界面选择是分享整个屏幕还是某个窗口。

通过本接口，您就可以查询到当前系统中可用于分享的窗口的 ID、名称以及缩略图。我们在 Demo 中提供了一份默认的界面实现供您参考。

参数	描述

iconSize	指定要获取的窗口图标大小。
thumbnailSize	指定要获取的窗口缩略图大小，缩略图可用于绘制在窗口选择界面上。

注意

1. 返回的列表中包含屏幕和应用窗口，屏幕是列表中的第一个元素。如果用户有多个显示器，那么每个显示器都是一个分享目标。
2. 请不要使用 delete ITRTCScreenCaptureSourceList* 删除 SourceList，这很容易导致崩溃，请使用 ITRTCScreenCaptureSourceList 中的 release 方法释放列表。

返回值说明：

窗口列表包括屏幕。

selectScreenCaptureTarget

selectScreenCaptureTarget

void selectScreenCaptureTarget	(const TRTCScreenCaptureSourceInfo &source
	const RECT& captureRect
	const TRTCScreenCaptureProperty &property)

选取要分享的屏幕或窗口（该接口仅支持桌面系统）

当您通过 getScreenCaptureSources 获取到可以分享的屏幕和窗口之后，您可以调用该接口选定期望分享的目标屏幕或目标窗口。
在屏幕分享的过程中，您也可以随时调用该接口以切换分享目标。

支持如下四种情况：
共享整个屏幕：sourceInfoList 中 type 为 Screen 的 source，captureRect 设为 { 0, 0, 0, 0 }
共享指定区域：sourceInfoList 中 type 为 Screen 的 source，captureRect 设为非 nullptr，例如 { 100, 100, 300, 300 }
共享整个窗口：sourceInfoList 中 type 为 Window 的 source，captureRect 设为 { 0, 0, 0, 0 }
共享窗口区域：sourceInfoList 中 type 为 Window 的 source，captureRect 设为非 nullptr，例如 { 100, 100, 300, 300 }

参数	描述
captureRect	指定捕获的区域
property	指定屏幕分享目标的属性，包括捕获鼠标，高亮捕获窗口等，详情参考

版权所有：腾讯云计算（北京）有限责任公司

	TRTCScreenCaptureProperty 定义
source	指定分享源

注意

设置高亮边框颜色、宽度参数在 Mac 平台不生效。

setSubStreamEncoderParam

setSubStreamEncoderParam

void setSubStreamEncoderParam	(const TRTCVideoEncParam& param)

设置屏幕分享（即辅路）的视频编码参数（桌面系统和移动系统均已支持）

该接口可以设定远端用户所看到的屏幕分享（即辅路）的画面质量，同时也能决定云端录制出的视频文件中屏幕分享的画面质量。

请注意如下两个接口的差异：

setVideoEncoderParam 用于设置主路画面（TRTCVideoStreamTypeBig，一般用于摄像头）的视频编码参数。

setSubStreamEncoderParam 用于设置辅路画面（TRTCVideoStreamTypeSub，一般用于屏幕分享）的视频编码参数。

参数	描述
param	辅流编码参数，详情请参见 TRTCVideoEncParam。

setSubStreamMixVolume

setSubStreamMixVolume

void setSubStreamMixVolume	(uint32_t volume)

设置屏幕分享时的混音音量大小（该接口仅支持桌面系统）

这个数值越高，屏幕分享音量的占比就越高，麦克风音量占比就越小，所以不推荐设置得太大，否则麦克风的声音就被压制了。

参数	描述
volume	设置的混音音量大小，范围0 - 100。

addExcludedShareWindow

addExcludedShareWindow

void addExcludedShareWindow	(TXView windowID)

将指定窗口加入屏幕分享的排除列表中（该接口仅支持桌面系统）

加入排除列表中的窗口不会被分享出去，常见的用法是将某个应用的窗口加入到排除列表中以避免隐私问题。
支持启动屏幕分享前设置过滤窗口，也支持屏幕分享过程中动态添加过滤窗口。

参数	描述
windowID	不希望分享出去的窗口

注意

1. 该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeScreen 时生效，即只有在分享整个屏幕内容时，排除指定窗口的功能才生效。
2. 使用该接口添加到排除列表中的窗口会在退出房间后被 SDK 自动清除。
3. Mac 平台下请传入窗口 ID（即 CGWindowID），您可以通过 TRTCScreenCaptureSourceInfo 中的 sourceId 成员获得。

removeExcludedShareWindow

removeExcludedShareWindow

void removeExcludedShareWindow	(TXView windowID)

将指定窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）

参数	描述
windowID	要排除的窗口 id。

removeAllExcludedShareWindow

removeAllExcludedShareWindow

将所有窗口从屏幕分享的排除列表中移除（该接口仅支持桌面系统）

addIncludedShareWindow

addIncludedShareWindow

void addIncludedShareWindow	(TXView windowID)

将指定窗口加入屏幕分享的包含列表中（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。即只有在分享窗口内容时，额外包含指定窗口的功能才生效。

您在 startScreenCapture 之前和之后调用均可。

参数	描述
windowID	希望被分享出去的窗口（Windows 平台下为窗口句柄： HWND）

注意

通过该方法添加到包含列表中的窗口，会在退出房间后被 SDK 自动清除。

removeIncludedShareWindow

removeIncludedShareWindow

void removeIncludedShareWindow	(TXView windowID)

将指定窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。即只有在分享窗口内容时，额外包含指定窗口的功能才生效。

参数	描述
windowID	希望被分享出去的窗口（Mac 平台：窗口 ID；Windows 平台：HWND）

removeAllIncludedShareWindow

removeAllIncludedShareWindow

将全部窗口从屏幕分享的包含列表中移除（该接口仅支持桌面系统）

该接口只有在 TRTCScreenCaptureSourceInfo 中的 type 指定为 TRTCScreenCaptureSourceTypeWindow 时生效。即只有在分享窗口内容时，额外包含指定窗口的功能才生效。

版权所有：腾讯云计算（北京）有限责任公司

enableCustomVideoCapture

enableCustomVideoCapture

void enableCustomVideoCapture	(TRTCVideoStreamType streamType
	bool enable)

启用/关闭视频自定义采集模式

开启该模式后，SDK 不在运行原有的视频采集流程，即不再继续从摄像头采集数据和美颜，而是只保留视频编码和发送能力。

您需要通过 sendCustomVideoData 不断地向 SDK 塞入自己采集的视频画面。

参数	描述
enable	是否启用，默认值：false。
streamType	用于指定视频流类型，TRTCVideoStreamTypeBig：高清大画面；TRTCVideoStreamTypeSub：辅路画面。

sendCustomVideoData

sendCustomVideoData

void sendCustomVideoData	(TRTCVideoStreamType streamType
	TRTCVideoFrame* frame)

向 SDK 投送自己采集的视频帧

使用此接口可以向 SDK 投送自己采集的视频帧，SDK 会将视频帧进行编码并通过自身的网络模块传输出去。

参数 TRTCVideoFrame 推荐下列填写方式（其他字段不需要填写）：

pixelFormat：Windows 和 Android 平台仅支持 TRTCVideoPixelFormat_I420，iOS 和 Mac平台支持 TRTCVideoPixelFormat_I420 和 TRTCVideoPixelFormat_BGRA32。

bufferType：推荐选择 TRTCVideoBufferType_Buffer。

data：用于承载视频帧数据的 buffer。

length：视频帧数据长度，如果 pixelFormat 设定为 I420 格式，length 可以按照如下公式计算：length = width × height × 3 / 2。

width：视频图像的宽度，如 640 px。

height：视频图像的高度，如 480 px。

版权所有：腾讯云计算（北京）有限责任公司

timestamp：时间戳，单位为毫秒（ms），请使用视频帧在采集时被记录下来的时间戳（可以在采集到一帧视频帧之后，通过调用 generateCustomPTS 获取时间戳）。

参考文档：自定义采集和渲染。

参数	描述
frame	视频数据，支持 I420 格式数据。
streamType	用于指定视频流类型，TRTCVideoStreamTypeBig：高清大画面；TRTCVideoStreamTypeSub：辅路画面。

注意

1. 推荐您在采集到的一帧视频帧后，即调用 generateCustomPTS 接口获取该帧的 timestamp 数值，这样可以获得最佳的音画同步效果。
2. SDK 最终编码出的视频帧率并不是由您调用本接口的频率决定的，而是由您在 setVideoEncoderParam 中所设置的 FPS 决定的。
3. 请尽量保持本接口的调用间隔是均匀的，否则会导致编码器输出帧率不稳或者音画不同步等问题。
4. iOS 和 Mac平台目前支持传入 TRTCVideoPixelFormat_I420 或 TRTCVideoPixelFormat_BGRA32 格式的视频帧。
5. Windows 和 Android 平台目前仅支持传入 TRTCVideoPixelFormat_I420 格式的视频帧。

enableCustomAudioCapture

enableCustomAudioCapture

void enableCustomAudioCapture	(bool enable)

启用音频自定义采集模式

开启该模式后，SDK 不在运行原有的音频采集流程，即不再继续从麦克风采集音频数据，而是只保留音频编码和发送能力。

您需要通过 sendCustomAudioData 不断地向 SDK 塞入自己采集的音频数据。

参数	描述
enable	是否启用，默认值：false。

注意

由于回声抵消（AEC）需要严格的控制声音采集和播放的时间，所以开启自定义音频采集后，AEC 能力可能会失效。

版权所有：腾讯云计算（北京）有限责任公司

sendCustomAudioData

sendCustomAudioData

void sendCustomAudioData	(TRTCAudioFrame* frame)

向 SDK 投送自己采集的音频数据

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：

audioFormat：音频数据格式，仅支持 TRTCAudioFrameFormatPCM。

data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。

sampleRate：采样率，支持：16000、24000、32000、44100、48000。

channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。

timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在采集到一帧音频帧之后，通过调用 generateCustomPTS 获取时间戳）。

参考文档：自定义采集和渲染。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

enableMixExternalAudioFrame

enableMixExternalAudioFrame

void enableMixExternalAudioFrame	(bool enablePublish
	bool enablePlayout)

启用/关闭自定义音轨

开启后，您可以通过本接口向 SDK 混入一条自定义的音轨。通过两个布尔型参数，您可以控制该音轨是否要在远端和本地播放。

参数	描述
enablePlayout	控制混入的音轨是否要在本地播放，默认值：false。

enablePublish	控制混入的音轨是否要在远端播放，默认值：false。

注意

如果您指定参数 enablePublish 和 enablePlayout 均为 false，代表完全关闭您的自定义音轨。

mixExternalAudioFrame

mixExternalAudioFrame

int mixExternalAudioFrame	(TRTCAudioFrame* frame)

向 SDK 混入自定义音轨

调用该接口之前，您需要先通过 enableMixExternalAudioFrame 开启自定义音轨，之后就可以通过本接口将自己的音轨以 PCM 格式混入到 SDK 中。

理想情况下，我们期望您的代码能够以非常均匀的速度向 SDK 提供音轨数据。但我们也非常清楚，完美的调用间隔是一个巨大的挑战。

所以 SDK 内部会开启一个音轨数据的缓冲区，该缓冲区的作用类似一个“蓄水池”，它能够暂存您传入的音轨数据，平抑由于接口调用间隔的抖动问题。

本接口的返回值代表这个音轨缓冲区的大小，单位是毫秒（ms），比如：如果该接口返回 50，则代表当前的音轨缓冲区有 50ms 的音轨数据。因此只要您在 50ms 内再次调用本接口，SDK 就能保证您混入的音轨数据是连续的。

当您调用该接口后，如果发现返回值 > 100ms，则可以等待一帧音频帧的播放时间之后再次调用；如果返回值 < 100ms，则代表缓冲区比较小，您可以再次混入一些音轨数据以确保音轨缓冲区的大小维持在“安全水位”以上。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：

data：音频帧 buffer。音频帧数据只支持 PCM 格式，支持[5ms ~ 100ms]帧长，推荐使用 20ms 帧长，长度计算方法：【48000采样率、单声道的帧长度：48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节】。

sampleRate：采样率，支持：16000、24000、32000、44100、48000。

channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。

timestamp：时间戳，单位为毫秒（ms），请使用音频帧在采集时被记录下来的时间戳（可以在获得一帧音频帧之后，通过调用 generateCustomPTS 获得时间戳）。

参数	描述
frame	音频数据

注意

请您精准地按每帧时长的间隔调用本接口，数据投送间隔不均匀时极易触发声音卡顿。

返回值说明：

>= 0 缓冲的长度，单位：ms。< 0 错误（-1 未启用 mixExternalAudioFrame）

setMixExternalAudioVolume

setMixExternalAudioVolume

void setMixExternalAudioVolume	(int publishVolume
	int playoutVolume)

设置推流时混入外部音频的推流音量和播放音量

参数	描述
playoutVolume	设置的播放音量大小，范围0 - 100，-1表示不改变。
publishVolume	设置的推流音量大小，范围0 - 100，-1表示不改变。

generateCustomPTS

generateCustomPTS

生成自定义采集时的时间戳

本接口仅适用于自定义采集模式，用于解决音视频帧的采集时间（capture time）和投送时间（send time）不一致所导致的音画不同步问题。

当您通过 sendCustomVideoData 或 sendCustomAudioData 等接口进行自定义视频或音频采集时，请按照如下操作使用该接口：
1. 首先，在采集到一帧视频或音频帧时，通过调用本接口获得当时的 PTS 时间戳。
2. 之后可以将该视频或音频帧送入您使用的前处理模块（如第三方美颜组件，或第三方音效组件）。
3. 在真正调用 sendCustomVideoData 或 sendCustomAudioData 进行投送时，请将该帧在采集时记录的 PTS 时间戳赋值给 TRTCVideoFrame 或 TRTCAudioFrame 中的 timestamp 字段。

返回值说明：

时间戳（单位：ms）

enableLocalVideoCustomProcess

enableLocalVideoCustomProcess

int enableLocalVideoCustomProcess	(bool enable
	TRTCVideoPixelFormat pixelFormat
	TRTCVideoBufferType bufferType)

.1 开启视频第三方美颜

开启后，您可以通过 ITRTCVideoFrameCallback 获取到指定像素格式和视频数据结构类型的图像帧。

参数	描述
bufferType	指定视频数据结构类型。
enable	是否启用视频第三方美颜，默认为关闭状态。
pixelFormat	指定回调的像素格式。

返回值说明：

0：成功；<0：错误

setLocalVideoCustomProcessCallback

setLocalVideoCustomProcessCallback

void setLocalVideoCustomProcessCallback	(ITRTCVideoFrameCallback* callback)

.2 设置第三方美颜的视频数据回调

设置该回调之后，SDK 会把采集到的视频帧通过您设置的 callback 回调出来，用于第三方美颜组件进行二次处理，之后 SDK 会将处理后的视频帧进行编码和发送。

参数	描述
callback	自定义美颜回调，详见 ITRTCVideoFrameCallback。

setLocalVideoRenderCallback

setLocalVideoRenderCallback

int setLocalVideoRenderCallback	(TRTCVideoPixelFormat pixelFormat

	TRTCVideoBufferType bufferType
	ITRTCVideoRenderCallback* callback)

设置本地视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
您可以通过调用 setLocalVideoRenderCallback(TRTCVideoPixelFormat_Unknown, TRTCVideoBufferType_Unknown, nullptr) 停止回调。
iOS、Mac、Windows 平台目前仅支持回调 TRTCVideoPixelFormat_I420 或 TRTCVideoPixelFormat_BGRA32 像素格式的视频帧。
Android 平台目前仅支持传入 TRTCVideoPixelFormat_I420, TRTCVideoPixelFormat_RGBA32 或 TRTCVideoPixelFormat_Texture_2D 像素格式的视频帧。

参数	描述
bufferType	指定视频数据结构类型
callback	自定义渲染回调
pixelFormat	指定回调的像素格式

返回值说明：

0：成功；<0：错误

setRemoteVideoRenderCallback

setRemoteVideoRenderCallback

int setRemoteVideoRenderCallback	(const char* userId
	TRTCVideoPixelFormat pixelFormat
	TRTCVideoBufferType bufferType
	ITRTCVideoRenderCallback* callback)

设置远端视频自定义渲染回调

设置该回调之后，SDK 内部会跳过原来的渲染流程，并把采集到的数据回调出来，您需要自己完成画面渲染。
您可以通过调用 setRemoteVideoRenderCallback(TRTCVideoPixelFormat_Unknown, TRTCVideoBufferType_Unknown, nullptr) 停止回调。

iOS、Mac、Windows 平台目前仅支持回调 TRTCVideoPixelFormat_I420 或 TRTCVideoPixelFormat_BGRA32 像素格式的视频帧。

Android 平台目前仅支持传入 TRTCVideoPixelFormat_I420, TRTCVideoPixelFormat_RGBA32 或 TRTCVideoPixelFormat_Texture_2D 像素格式的视频帧。

参数	描述
bufferType	指定视频数据结构类型，目前只支持 TRTCVideoBufferType_Buffer
callback	自定义渲染回调
pixelFormat	指定回调的像素格式
userId	远端用户id

注意

实际使用时，需要先调用 startRemoteView(userid, nullptr) 来获取远端用户的视频流（view 设置为 nullptr 即可），否则不会有数据回调出来。

返回值说明：

0：成功；<0：错误

setAudioFrameCallback

setAudioFrameCallback

int setAudioFrameCallback	(ITRTCAudioFrameCallback* callback)

设置音频数据自定义回调

设置该回调之后，SDK 内部会把音频数据（PCM 格式）回调出来，包括：

onCapturedAudioFrame：本地麦克风采集到的音频数据回调

onLocalProcessedAudioFrame：本地采集并经过音频模块前处理后的音频数据回调

onPlayAudioFrame：混音前的每一路远程用户的音频数据

onMixedPlayAudioFrame：将各路音频混合之后并最终要由系统播放出的音频数据回调

注意

设置回调为空即代表停止自定义音频回调，反之，设置回调不为空则代表启动自定义音频回调。

setCapturedAudioFrameCallbackFormat

setCapturedAudioFrameCallbackFormat

int setCapturedAudioFrameCallbackFormat	(TRTCAudioFrameCallbackFormat* format)

设置本地麦克风采集出的音频帧回调格式

本接口用于设置 onCapturedAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调 20ms 帧长的数据，则采样点数应该填：960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意
最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setLocalProcessedAudioFrameCallbackFormat

setLocalProcessedAudioFrameCallbackFormat

int setLocalProcessedAudioFrameCallbackFormat	(TRTCAudioFrameCallbackFormat* format)

设置经过前处理后的本地音频帧回调格式

本接口用于设置 onLocalProcessedAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述

format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

setMixedPlayAudioFrameCallbackFormat

setMixedPlayAudioFrameCallbackFormat

int setMixedPlayAudioFrameCallbackFormat	(TRTCAudioFrameCallbackFormat* format)

设置最终要由系统播放出的音频帧回调格式

本接口用于设置 onMixedPlayAudioFrame 回调出来的 AudioFrame 的格式：
sampleRate：采样率，支持：16000、32000、44100、48000。
channel：声道数（如果是立体声，数据是交叉的），单声道：1；双声道：2。
samplesPerCall：采样点数，定义回调数据帧长。帧长必须为 10ms 的整数倍。

如果希望用毫秒数计算回调帧长，则将毫秒数转换成采样点数的公式为：采样点数 = 毫秒数 * 采样率 / 1000。
举例：48000 采样率希望回调20ms帧长的数据，则采样点数应该填: 960 = 20 * 48000 / 1000。

参数	描述
format	音频数据回调格式。

注意

最终回调的帧长度是以字节为单位，采样点数转换成字节数的计算公式为：字节数 = 采样点数 * channel * 2（位宽）
举例：48000 采样率，双声道，20ms 帧长，采样点数为 960，字节数为 3840 = 960 * 2 * 2

返回值说明：

0：成功；<0：错误

enableCustomAudioRendering

enableCustomAudioRendering

void enableCustomAudioRendering	(bool enable)

开启音频自定义播放

如果您需要外接一些特定的音频设备，或者希望自己掌控音频的播放逻辑，您可以通过该接口启用音频自定义播放。
启用音频自定义播放后，SDK 将不再调用系统的音频接口播放数据，您需要通过 getCustomAudioRenderingFrame 获取 SDK 要播放的音频帧并自行播放。

参数	描述
enable	是否启用音频自定义播放，默认为关闭状态。

注意

需要您在进入房间前设置才能生效，暂不支持进入房间后再设置。

getCustomAudioRenderingFrame

getCustomAudioRenderingFrame

void getCustomAudioRenderingFrame	(TRTCAudioFrame* audioFrame)

获取可播放的音频数据

调用该接口之前，您需要先通过 enableCustomAudioRendering 开启音频自定义播放。

参数 TRTCAudioFrame 推荐下列填写方式（其他字段不需要填写）：
sampleRate：采样率，必填，支持 16000、24000、32000、44100、48000。
channel：声道数，必填，单声道请填1，双声道请填2，双声道时数据是交叉的。
data：用于获取音频数据的 buffer。需要您根据一帧音频帧的帧长度分配好 data 的内存大小。获取的 PCM 数据支持 10ms 或 20ms 两种帧长，推荐使用 20ms 的帧长。

计算公式为：采样率 x 播放时长 x 声道数量 x 2（每个样点的字节数），例如：48000采样率、单声道、且播放时长为 20ms 的一帧音频帧的 buffer 大小为 48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节。

参数	描述
audioFrame	音频数据帧。

注意

1. 参数 audioFrame 中的 sampleRate、channel 需提前设置好，同时分配好所需读取帧长的 data 空间。
2. SDK 内部会根据 sampleRate 和 channel 自动填充 data 数据。
3. 建议由系统的音频播放线程直接驱动该函数的调用，在播放完一帧音频之后，即调用该接口获取下一帧可播放的音频数据。

sendCustomCmdMsg

sendCustomCmdMsg

bool sendCustomCmdMsg	(uint32_t cmdId
	const uint8_t* data
	uint32_t dataSize
	bool reliable
	bool ordered)

使用 UDP 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 UDP 通道，向当前房间里的其他用户广播自定义数据，以达到传输信令的目的。
房间中的其他用户可以通过 ITRTCCloudCallback 中的 onRecvCustomCmdMsg 回调接收消息。

参数	描述
cmdID	消息 ID，取值范围为1 - 10。
data	待发送的消息，单个消息的最大长度被限制为 1KB。
ordered	是否要求有序，即是否要求接收端的数据包顺序和发送端的数据包顺序一致（这会带来一定的接收延时）。
reliable	是否可靠发送，可靠发送可以获得更高的发送成功率，但可靠发送比不可靠发送会带来更大的接收延迟。

注意

1. 发送消息到房间内所有用户（暂时不支持 Web/小程序端），每秒最多能发送30条消息。
2. 每个包最大为 1KB，超过则很有可能会被中间路由器或者服务器丢弃。
3. 每个客户端每秒最多能发送总计 8KB 数据。
4. 请将 reliable 和 ordered 同时设置为 true 或同时设置为 false，暂不支持交叉设置。
5. 强烈建议您将不同类型的消息设定为不同的 cmdID，这样可以在要求有序的情况下减小消息时延。
6. 目前仅支持主播身份。

版权所有：腾讯云计算（北京）有限责任公司

返回值说明：

true：消息已经发出；false：消息发送失败。

sendSEIMsg

sendSEIMsg

bool sendSEIMsg	(const uint8_t* data
	uint32_t dataSize
	int32_t repeatCount)

使用 SEI 通道发送自定义消息给房间内所有用户

该接口可以让您借助 TRTC 的 SEI 通道，向当前房间里的其他用户广播自定义数据，已达到传输信令的目的。

视频帧的头部有一个叫做 SEI 的头部数据块，该接口的原理就是利用这个被称为 SEI 的头部数据块，将您要发送的自定义信令嵌入其中，使其同视频帧一并发送出去。
因此，与 sendCustomCmdMsg 相比，SEI 通道传输的信令具有更好的兼容性：信令可以伴随着视频帧一直传输到直播 CDN 上。
不过，由于视频帧头部的数据块不能太大，建议您使用该接口时，尽量将信令控制在几个字节的大小。

最常见的用法是把自定义的时间戳（timestamp）用本接口嵌入视频帧中，实现消息和画面的完美对齐（比如：教育场景下的课件和视频信号的对齐）。
房间中的其他用户可以通过 ITRTCCloudCallback 中的 onRecvSEIMsg 回调接收消息。

参数	描述
data	待发送的数据，最大支持 1KB（1000字节）的数据大小
repeatCount	发送数据次数

注意

本接口有以下限制：
1. 数据在接口调用完后不会被即时发送出去，而是从下一帧视频帧开始与视频帧一起发送。
2. 发送消息到房间内所有用户，每秒最多能发送 30 条消息（与 sendCustomCmdMsg 共享限制）。
3. 每个包最大为 1KB，若发送大量数据，会导致视频码率增大，可能导致视频画质下降甚至卡顿（与 sendCustomCmdMsg 共享限制）。
4. 每个客户端每秒最多能发送总计8KB数据（与 sendCustomCmdMsg 共享限制）。

版权所有：腾讯云计算（北京）有限责任公司

5. 若指定多次发送（repeatCount > 1），则数据会被带在后续的连续 repeatCount 个视频帧中发送出去，同样会导致视频码率增大。
6. 如果 repeatCount > 1，多次发送，接收消息 onRecvSEIMsg 回调也可能会收到多次相同的消息，需要去重。

返回值说明：

true：消息已通过限制，等待后续视频帧发送；false：消息被限制发送

startSpeedTest

startSpeedTest

int startSpeedTest	(const TRTCSpeedTestParams& params)

开始进行网速测试（进入房间前使用）

参数	描述
params	测速选项

注意

1. 测速过程将产生少量的基础服务费用，详见 计费概述 > 基础服务 文档说明。
2. 请在进入房间前进行网速测试，在房间中网速测试会影响正常的音视频传输效果，而且由于干扰过多，网速测试结果也不准确。
3. 同一时间只允许一项网速测试任务运行。

返回值说明：

接口调用结果，< 0：失败

stopSpeedTest

stopSpeedTest

停止网络测速

getSDKVersion

getSDKVersion

获取 SDK 版本信息

setLogLevel

setLogLevel

void setLogLevel	(TRTCLogLevel level)

设置 Log 输出级别

参数	描述
level	参见 TRTCLogLevel，默认值：TRTCLogLevelNone

setConsoleEnabled

setConsoleEnabled

void setConsoleEnabled	(bool enabled)

启用/禁用控制台日志打印

参数	描述
enabled	指定是否启用，默认：禁止状态。

setLogCompressEnabled

setLogCompressEnabled

void setLogCompressEnabled	(bool enabled)

启用/禁用日志的本地压缩

开启压缩后，Log 存储体积明显减小，但需要腾讯云提供的 Python 脚本解压后才能阅读。
禁用压缩后，Log 采用明文存储，可以直接用记事本打开阅读，但占用空间较大。

参数	描述
enabled	指定是否启用，默认为启动状态

setLogDirPath

setLogDirPath

void setLogDirPath	(const char* path)

设置本地日志的保存路径

通过该接口您可以更改 SDK 本地日志的默认存储路径，SDK 默认的本地日志的存储位置：
Windows 平台：在 C:/Users/[系统用户名]/AppData/Roaming/liteav/log，即 %appdata%/liteav/log 下。
iOS 或 Mac 平台：在 sandbox Documents/log 下。
Android 平台：在 /app私有目录/files/log/liteav/ 下。

参数	描述
path	存储日志的路径

注意

请务必在所有其他接口之前调用，并且保证您指定的目录是存在的，并且您的应用程序拥有对该目录的读写权限。

setLogCallback

setLogCallback

void setLogCallback	(ITRTCLogCallback* callback)

设置日志回调

showDebugView

showDebugView

void showDebugView	(int showType)

显示仪表盘

“仪表盘”是位于视频渲染控件之上的一个半透明的调试信息浮层，用于展示音视频信息和事件信息，便于对接和调试。

参数	描述
showType	0：不显示；1：显示精简版（仅显示音视频信息）；2：显示完整版（包含音视频信息和事

版权所有：腾讯云计算（北京）有限责任公司

	件信息）。

callExperimentalAPI

callExperimentalAPI

char* callExperimentalAPI	(const char *jsonStr)

调用实验性接口

enablePayloadPrivateEncryption

enablePayloadPrivateEncryption

int enablePayloadPrivateEncryption	(bool enabled
	const TRTCPayloadPrivateEncryptionConfig& config)

开启或关闭媒体流私有加密

在安全要求较高的场景下，TRTC 建议您在加入房间前，调用 enablePayloadPrivateEncryption 方法开启媒体流私有加密。
用户退出房间后，SDK 会自动关闭私有加密。如需重新开启私有加密，您需要在用户再次加入房间前调用该方法。

参数	描述
config	配置媒体流私有加密的算法和密钥，参见 TRTCPayloadPrivateEncryptionConfig。
enabled	是否开启媒体流私有加密。

注意

TRTC 已经内置对媒体流进行加密后再传输，启用媒体流私有加密后将使用您传入的密匙与初始向量进行再次加密。

返回值说明：

接口调用结果，0: 方法调用成功， -1: 传入参数无效， -2: 功能已过期。若需解锁：请前往中国大陆站点开通 TRTC 旗舰版套餐，并填写 **申请**，审核通过后方可使用。

TRTCCloudCallback

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: ITRTCCloudCallback @ TXLiteAVSDK

Function: 腾讯云实时音视频的事件回调接口

TRTCCloudCallback

ITRTCCloudCallback

函数列表	描述
onError	错误事件回调
onWarning	警告事件回调
onEnterRoom	进入房间成功与否的事件回调
onExitRoom	离开房间的事件回调
onSwitchRole	切换角色的事件回调
onSwitchRoom	切换房间的结果回调
onConnectOtherRoom	请求跨房通话的结果回调
onDisconnectOtherRoom	结束跨房通话的结果回调
onUpdateOtherRoomForwardMode	更改跨房主播上行能力的结果回调
onRemoteUserEnterRoom	有用户加入当前房间
onRemoteUserLeaveRoom	有用户离开当前房间
onUserVideoAvailable	某远端用户发布/取消了主路视频画面
onUserSubStreamAvailable	某远端用户发布/取消了辅路视频画面
onUserAudioAvailable	某远端用户发布/取消了自己的音频

onFirstVideoFrame	SDK 开始渲染自己本地或远端用户的首帧画面
onFirstAudioFrame	SDK 开始播放远端用户的首帧音频
onSendFirstLocalVideoFrame	自己本地的首个视频帧已被发布出去
onSendFirstLocalAudioFrame	自己本地的首个音频帧已被发布出去
onRemoteVideoStatusUpdated	远端视频状态变化的事件回调
onRemoteAudioStatusUpdated	远端音频状态变化的事件回调
onUserVideoSizeChanged	用户视频大小发生改变回调
onNetworkQuality	网络质量的实时统计回调
onStatistics	音视频技术指标的实时统计回调
onSpeedTestResult	网速测试的结果回调
onConnectionLost	SDK 与云端的连接已经断开
onTryToReconnect	SDK 正在尝试重新连接到云端
onConnectionRecovery	SDK 与云端的连接已经恢复
onCameraDidReady	摄像头准备就绪
onMicDidReady	麦克风准备就绪
onUserVoiceVolume	音量大小的反馈回调
onDeviceChange	本地设备的通断状态发生变化（仅适用于桌面系统）
onAudioDeviceCaptureVolumeChanged	当前麦克风的系统采集音量发生变化
onAudioDevicePlayoutVolumeChanged	当前系统的播放音量发生变化
onSystemAudioLoopbackError	系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）
onTestMicVolume	测试麦克风时的音量回调
onTestSpeakerVolume	测试扬声器时的音量回调
onRecvCustomCmdMsg	收到自定义消息的事件回调
onMissCustomCmdMsg	自定义消息丢失的事件回调
onRecvSEIMsg	收到 SEI 消息的回调

版权所有：腾讯云计算（北京）有限责任公司

onStartPublishing	开始向腾讯云直播 CDN 上发布音视频流的事件回调
onStopPublishing	停止向腾讯云直播 CDN 上发布音视频流的事件回调
onStartPublishCDNStream	开始向非腾讯云 CDN 上发布音视频流的事件回调
onStopPublishCDNStream	停止向非腾讯云 CDN 上发布音视频流的事件回调
onSetMixTranscodingConfig	设置云端混流的排版布局和转码参数的事件回调
onStartPublishMediaStream	开始发布媒体流的事件回调
onUpdatePublishMediaStream	更新媒体流的事件回调
onStopPublishMediaStream	停止媒体流的事件回调
onCdnStreamStateChanged	RTMP/RTMPS 推流状态发生改变回调
onScreenCaptureStarted	屏幕分享开启的事件回调
onScreenCapturePaused	屏幕分享暂停的事件回调
onScreenCaptureResumed	屏幕分享恢复的事件回调
onScreenCaptureStoped	屏幕分享停止的事件回调
onScreenCaptureCovered	屏幕分享的目标窗口被遮挡的事件回调（仅适用于 Windows 操作系统）
onLocalRecordBegin	本地录制任务已经开始的事件回调
onLocalRecording	本地录制任务正在进行中的进展事件回调
onLocalRecordFragment	本地录制分片的事件回调
onLocalRecordComplete	本地录制任务已经结束的事件回调
onSnapshotComplete	本地截图完成的事件回调
onUserEnter	有主播加入当前房间（已废弃）
onUserExit	有主播离开当前房间（已废弃）
onAudioEffectFinished	音效播放已结束（已废弃）
onPlayBGMBegin	开始播放背景音乐（已废弃）
onPlayBGMProgress	背景音乐的播放进度回调（已废弃）

版权所有：腾讯云计算（北京）有限责任公司

onPlayBGMComplete	背景音乐播放已经结束（已废弃）
onSpeedTest	服务器测速的结果回调（已废弃）

ITRTCVideoRenderCallback

函数列表	描述
onRenderVideoFrame	自定义视频渲染回调

ITRTCVideoFrameCallback

函数列表	描述
onGLContextCreated	SDK 内部 OpenGL 环境已经创建的通知
onProcessVideoFrame	用于对接第三方美颜组件的视频处理回调
onGLContextDestroy	SDK 内部 OpenGL 环境被销毁的通知

ITRTCAudioFrameCallback

函数列表	描述
onCapturedAudioFrame	本地采集并经过音频模块前处理后的音频数据回调
onLocalProcessedAudioFrame	本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调
onPlayAudioFrame	混音前的每一路远程用户的音频数据
onMixedPlayAudioFrame	将各路待播放音频混合之后并在最终提交系统播放之前的数据回调
onMixedAllAudioFrame	SDK 所有音频混合后的音频数据（包括采集到的和待播放的）

ITRTCLogCallback

函数列表	描述

版权所有：腾讯云计算（北京）有限责任公司

onLog	本地 LOG 的打印回调

onError

onError

void onError	(TXLiteAVError errCode
	const char* errMsg
	void* extraInfo)

错误事件回调

错误事件，表示 SDK 抛出的不可恢复的错误，比如进入房间失败或设备开启失败等。
参考文档：**错误码表**

参数	描述
errCode	错误码
errMsg	错误信息
extInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题

onWarning

onWarning

void onWarning	(TXLiteAVWarning warningCode
	const char* warningMsg
	void* extraInfo)

警告事件回调

警告事件，表示 SDK 抛出的提示性问题，比如视频出现卡顿或 CPU 使用率太高等。
参考文档：**错误码表**

参数	描述
extInfo	扩展信息字段，个别警告码可能会带额外的信息帮助定位问题

版权所有：腾讯云计算（北京）有限责任公司

warningCode	警告码
warningMsg	警告信息

onEnterRoom

onEnterRoom

void onEnterRoom	(int result)

进入房间成功与否的事件回调

调用 TRTCCloud 中的 enterRoom() 接口执行进房操作后，会收到来自 ITRTCCloudCallback 的 onEnterRoom(result) 回调：
如果加入成功，回调 result 会是一个正数（result > 0），代表进入房间所消耗的时间，单位是毫秒（ms）。
如果加入失败，回调 result 会是一个负数（result < 0），代表失败原因的错误码。
进房失败的错误码含义请参见**错误码表**。

参数	描述
result	result > 0 时为进房耗时（ms），result < 0 时为进房错误码。

注意

1. 在 Ver6.6 之前的版本，只有进房成功会抛出 onEnterRoom(result) 回调，进房失败由 onError() 回调抛出。
2. 在 Ver6.6 之后的版本：无论进房成功或失败，均会抛出 onEnterRoom(result) 回调，同时进房失败也会有 onError() 回调抛出。

onExitRoom

onExitRoom

void onExitRoom	(int reason)

离开房间的事件回调

调用 TRTCCloud 中的 exitRoom 接口会执行退出房间的相关逻辑，例如释放音视频设备资源和编解码器资源等。待 SDK 占用的所有资源释放完毕后，SDK 会抛出 onExitRoom() 回调通知到您。

如果您要再次调用 enterRoom 或者切换到其他的音视频 SDK，请等待 onExitRoom 回调到来后再执行相关操作。否则可能会遇到例如摄像头、麦克风设备被强占等各种异常问题。

参数	描述
reason	离开房间原因，0：主动调用 exitRoom 退出房间；1：被服务器踢出当前房间；2：当前房间整个被解散。

onSwitchRole

onSwitchRole

void onSwitchRole	(TXLiteAVError errCode
	const char* errMsg)

切换角色的事件回调

调用 TRTCCloud 中的 switchRole() 接口会切换主播和观众的角色，该操作会伴随一个线路切换的过程，待 SDK 切换完成后，会抛出 onSwitchRole() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onSwitchRoom

onSwitchRoom

void onSwitchRoom	(TXLiteAVError errCode
	const char* errMsg)

切换房间的结果回调

调用 TRTCCloud 中的 switchRoom() 接口可以让用户快速地从一个房间切换到另一个房间，待 SDK 切换完成后，会抛出 onSwitchRoom() 事件回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。

onConnectOtherRoom

onConnectOtherRoom

void onConnectOtherRoom	(const char* userId
	TXLiteAVError errCode
	const char* errMsg)

请求跨房通话的结果回调

调用 TRTCCloud 中的 connectOtherRoom() 接口会将两个不同房间中的主播拉通视频通话，也就是所谓的“主播 PK”功能。

调用者会收到 onConnectOtherRoom() 回调来获知跨房通话是否成功，

如果成功，两个房间中的所有用户都会收到来自另一个房间中的 PK 主播的 onUserVideoAvailable() 回调。

参数	描述
errCode	错误码，ERR_NULL 代表切换成功，其他请参见**错误码**。
errMsg	错误信息。
userId	要跨房通话的另一个房间中的主播的用户 ID。

onDisconnectOtherRoom

onDisconnectOtherRoom

void onDisconnectOtherRoom	(TXLiteAVError errCode
	const char* errMsg)

结束跨房通话的结果回调

onUpdateOtherRoomForwardMode

onUpdateOtherRoomForwardMode

void onUpdateOtherRoomForwardMode	(TXLiteAVError errCode
	const char* errMsg)

更改跨房主播上行能力的结果回调

onRemoteUserEnterRoom

onRemoteUserEnterRoom

void onRemoteUserEnterRoom	(const char* userId)

有用户加入当前房间

出于性能方面的考虑，在 TRTC 两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：

直播类场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）：该场景下的用户区分主播和观众两种角色，只有主播进入房间时才会触发该通知，观众进入房间时不会触发该通知。

通话类场景（TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall）：该场景下的用户没有角色的区分（可认为都是主播），任何用户进入房间都会触发该通知。

参数	描述
userId	远端用户的用户标识。

注意

1. 事件回调 onRemoteUserEnterRoom 和 onRemoteUserLeaveRoom 只适用于维护当前房间里的“用户列表”，有此事件回调不代表一定有视频画面。
2. 如果需要显示远程画面，请监听代表某个用户是否有视频画面的 onUserVideoAvailable() 事件回调。

onRemoteUserLeaveRoom

onRemoteUserLeaveRoom

void onRemoteUserLeaveRoom	(const char* userId
	int reason)

有用户离开当前房间

与 onRemoteUserEnterRoom 相对应，在两种不同的应用场景（即 AppScene，在 enterRoom 时通过第二个参数指定）下，该通知的行为会有差别：

直播类场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）：只有主播离开房间时才会触发该通知，观众离开房间不会触发该通知。

版权所有：腾讯云计算（北京）有限责任公司

通话类场景（TRTCAppSceneVideoCall 和 TRTCAppSceneAudioCall）：该场景下用户没有角色的区别，任何用户的离开都会触发该通知。

参数	描述
reason	离开原因，0表示用户主动退出房间，1表示用户超时退出，2表示被踢出房间，3表示主播因切换到观众退出房间。
userId	远端用户的用户标识。

onUserVideoAvailable

onUserVideoAvailable

void onUserVideoAvailable	(const char* userId
	bool available)

某远端用户发布/取消了主路视频画面

主路画面 一般被用于承载摄像头画面。当您收到 onUserVideoAvailable(userId, true) 通知时，表示该路画面已经有可播放的视频帧到达。
此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

当您收到 onUserVideoAvailable(userId, false) 通知时，表示该路远程画面已经被关闭，关闭的原因可能是该用户调用了 muteLocalVideo 或 stopLocalPreview。

参数	描述
available	该用户是否发布（或取消发布）了主路视频画面，true：发布；false：取消发布。
userId	远端用户的用户标识。

onUserSubStreamAvailable

onUserSubStreamAvailable

void onUserSubStreamAvailable	(const char* userId
	bool available)

某远端用户发布/取消了辅路视频画面

“辅路画面”一般被用于承载屏幕分享的画面。当您收到 onUserSubStreamAvailable(userId, true) 通知时，表示该路画面已经有可播放的视频帧到达。
此时，您需要调用 startRemoteView 接口订阅该用户的远程画面，订阅成功后，您会继续收到该用户的首帧画面渲染回调 onFirstVideoFrame(userid)。

参数	描述
available	该用户是否发布（或取消发布）了辅路视频画面，true: 发布；false：取消发布。
userId	远端用户的用户标识。

注意
显示辅路画面使用的函数是 startRemoteView 而非 startRemoteSubStreamView，startRemoteSubStreamView 已经被废弃。

onUserAudioAvailable

onUserAudioAvailable

void onUserAudioAvailable	(const char* userId
	bool available)

某远端用户发布/取消了自己的音频

当您收到 onUserAudioAvailable(userId, true) 通知时，表示该用户发布了自己的声音，此时 SDK 的表现为：
在自动订阅模式下，您无需做任何操作，SDK 会自动播放该用户的声音。
在手动订阅模式下，您可以通过 muteRemoteAudio(userid, false) 来播放该用户的声音。

参数	描述
available	该用户是否发布（或取消发布）了自己的音频，true: 发布；false：取消发布。
userId	远端用户的用户标识。

注意
SDK 默认使用自动订阅模式，您可以通过 setDefaultStreamRecvMode 设置为手动订阅，但需要在您进入房间之前调用才生效。

onFirstVideoFrame

onFirstVideoFrame

void onFirstVideoFrame	(const char* userId
	const TRTCVideoStreamType streamType
	const int width
	const int height)

SDK 开始渲染自己本地或远端用户的首帧画面

SDK 会在渲染自己本地或远端用户的首帧画面时抛出该事件，您可以通过回调事件中的 userId 参数来判断事件来自于“本地”还是来自于“远端”。

如果 userId 为空值，代表 SDK 已经开始渲染自己本地的视频画面，不过前提是您已经调用了 startLocalPreview 或 startScreenCapture。

如果 userId 不为空，代表 SDK 已经开始渲染远端用户的视频画面，不过前提是您已经调用了 startRemoteView 订阅了该用户的视频画面。

参数	描述
height	画面的高度。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	本地或远端的用户标识，如果 userId 为空值代表自己本地的首帧画面已到来，userId 不为空则代表远端用户的首帧画面已到来。
width	画面的宽度。

注意

1. 只有当您调用了 startLocalPreview 或 startScreenCapture 之后，才会触发自己本地的首帧画面事件回调。
2. 只有当您调用了 startRemoteView 或 startRemoteSubStreamView 之后，才会触发远端用户的首帧画面事件回调。

onFirstAudioFrame

onFirstAudioFrame

void onFirstAudioFrame	(const char* userId)

SDK 开始播放远端用户的首帧音频

SDK 会在播放远端用户的首帧音频时抛出该事件，本地音频的首帧事件暂不抛出。

参数	描述
userId	远端用户的用户标识。

onSendFirstLocalVideoFrame

onSendFirstLocalVideoFrame

void onSendFirstLocalVideoFrame	(const TRTCVideoStreamType streamType)

自己本地的首个视频帧已被发布出去

当您成功进入房间并通过 startLocalPreview 或 startScreenCapture 开启本地视频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行视频编码，并通过自身的网络模块向云端发布自己本地的视频数据。当 SDK 成功地向云端送出自己的第一帧视频数据帧以后，就会抛出 onSendFirstLocalVideoFrame 事件回调。

参数	描述
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。

onSendFirstLocalAudioFrame

onSendFirstLocalAudioFrame

自己本地的首个音频帧已被发布出去

当您成功进入房间并通过 startLocalAudio 开启本地音频采集之后（开启采集和进入房间的先后顺序无影响），SDK 就会开始进行音频编码，并通过自身的网络模块向云端发布自己本地的音频数据。当 SDK 成功地向云端送出自己的第一帧音频数据帧以后，就会抛出 onSendFirstLocalAudioFrame 事件回调。

onRemoteVideoStatusUpdated

onRemoteVideoStatusUpdated

void onRemoteVideoStatusUpdated	(const char* userId
	TRTCVideoStreamType streamType
	TRTCAVStatusType status
	TRTCAVStatusChangeReason reason

	void *extrainfo)

远端视频状态变化的事件回调

您可以通过此事件回调获取远端每一路画面的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。
reason	视频状态改变的原因。
status	视频状态：包括 Playing、Loading 和 Stopped 三种状态。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onRemoteAudioStatusUpdated

onRemoteAudioStatusUpdated

void onRemoteAudioStatusUpdated	(const char* userId
	TRTCAVStatusType status
	TRTCAVStatusChangeReason reason
	void *extrainfo)

远端音频状态变化的事件回调

您可以通过此事件回调获取远端每一路音频的播放状态（包括 Playing、Loading 和 Stopped 三种状态），从而进行相应的 UI 展示。

参数	描述
extrainfo	额外信息。
reason	音频状态改变的原因。
status	音频状态：包括 Playing、Loading 和 Stopped 三种状态。
userId	用户标识。

onUserVideoSizeChanged

onUserVideoSizeChanged

void onUserVideoSizeChanged	(const char* userId
	TRTCVideoStreamType streamType
	int newWidth
	int newHeight)

用户视频大小发生改变回调

当您收到 onUserVideoSizeChanged(userId, streamtype, newWidth, newHeight) 通知时，表示该路画面大小发生了调整，调整的原因可能是该用户调用了 setVideoEncoderParam 或者 setSubStreamEncoderParam 重新设置了画面尺寸。

参数	描述
newHeight	视频流的高度（像素）。
newWidth	视频流的宽度（像素）。
streamType	视频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	用户标识。

onNetworkQuality

onNetworkQuality

void onNetworkQuality	(TRTCQualityInfo localQuality
	TRTCQualityInfo* remoteQuality
	uint32_t remoteQualityCount)

网络质量的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 感知到的当前网络的上行和下行质量。
SDK 会使用一组内嵌的自研算法对当前网络的延迟高低、带宽大小以及稳定情况进行评估，并计算出一个评估结果：如果评估结果为 1（Excellent） 代表当前的网络情况非常好，如果评估结果为 6（Down）代表当前网络无法支

撑 TRTC 的正常通话。

参数	描述
localQuality	上行网络质量。
remoteQuality	下行网络质量，代表数据流历经“远端->云端->本端”这样一条完整的传输链路后，最终在本端统计到的数据质量。因此，这里的下行网络质量代表的是远端上行链路与本端下行链路共同影响的结果。

注意

暂时无法通过该接口单独判定远端用户的上行质量。

onStatistics

onStatistics

void onStatistics	(const TRTCStatistics& statistics)

音视频技术指标的实时统计回调

该统计回调每间隔2秒抛出一次，用于通知 SDK 内部音频、视频以及网络相关的专业技术指标，这些信息在 TRTCStatistics 均有罗列：

视频统计信息：视频的分辨率（resolution）、帧率（FPS）和比特率（bitrate）等信息。

音频统计信息：音频的采样率（samplerate）、声道（channel）和比特率（bitrate）等信息。

网络统计信息：SDK 和云端一次往返（SDK > Cloud > SDK）的网络耗时（rtt）、丢包率（loss）、上行流量（sentBytes）和下行流量（receivedBytes）等信息。

参数	描述
statistics	统计数据，包括自己本地的统计信息和远端用户的统计信息，详情请参考 TRTCStatistics。

注意

如果您只需要获知当前网络质量的好坏，并不需要花太多时间研究本统计回调，更推荐您使用 onNetworkQuality 来解决问题。

onSpeedTestResult

onSpeedTestResult

void onSpeedTestResult	(const TRTCSpeedTestResult& result)

网速测试的结果回调

该统计回调由 startSpeedTest: 触发。

参数	描述
result	网速测试结果，包括丢包、往返延迟、上下行的带宽速率，详情请参见 TRTCSpeedTestResult。

onConnectionLost

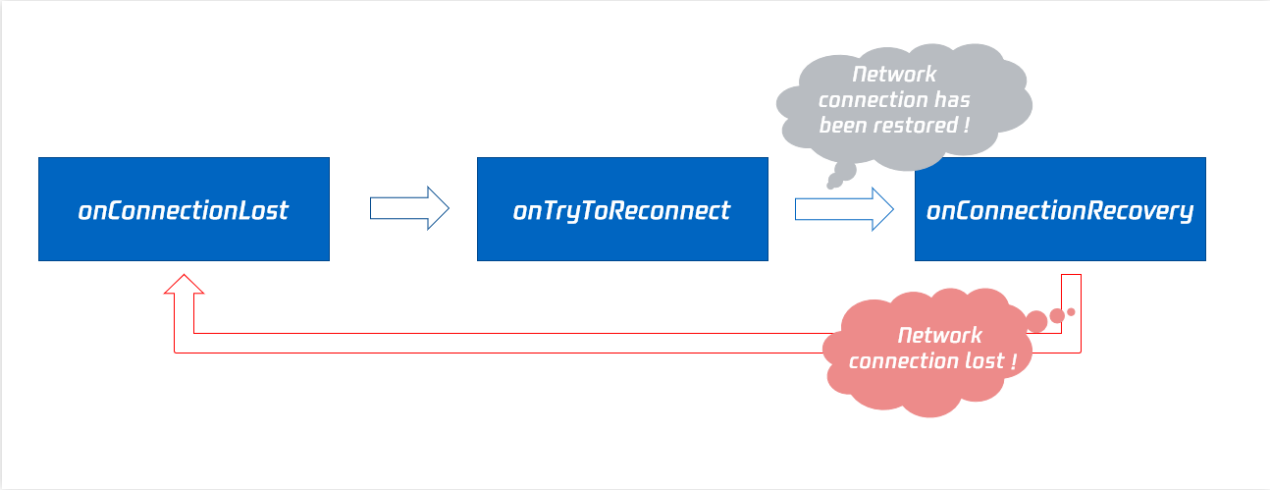
onConnectionLost

SDK 与云端的连接已经断开

SDK 会在跟云端的连接断开时抛出此事件回调，导致断开的原因大多是网络不可用或者网络切换所致，比如用户在通话中走进电梯时就可能会遇到此事件。

在抛出此事件之后，SDK 会努力跟云端重新建立连接，重连过程中会抛出 onTryToReconnect，连接恢复后会抛出 onConnectionRecovery 。

所以，SDK 会在如下三个连接相关的事件中按如下规律切换：



onTryToReconnect

onTryToReconnect

SDK 正在尝试重新连接到云端

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出本事件，连接恢复后会抛出 onConnectionRecovery。

onConnectionRecovery

onConnectionRecovery

SDK 与云端的连接已经恢复

SDK 会在跟云端的连接断开时抛出 onConnectionLost，之后会努力跟云端重新建立连接并抛出 onTryToReconnect，连接恢复后会抛出本事件回调。

onCameraDidReady

onCameraDidReady

摄像头准备就绪

当您调用 startLocalPreivew 之后，SDK 会尝试启动摄像头，如果摄像头能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问摄像头的权限，或者摄像头当前正在被其他程序独占使用中。
您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onMicDidReady

onMicDidReady

麦克风准备就绪

当您调用 startLocalAudio 之后，SDK 会尝试启动麦克风，如果麦克风能够启动成功就会抛出本事件。
如果启动失败，大概率是因为当前应用没有获得访问麦克风的权限，或者麦克风当前正在被其他程序独占使用中。
您可以通过捕获 onError 事件回调获知这些异常情况并通过 UI 界面提示用户。

onUserVoiceVolume

onUserVoiceVolume

void onUserVoiceVolume	(TRTCVolumeInfo* userVolumes
	uint32_t userVolumesCount

	uint32_t totalVolume)

音量大小的反馈回调

SDK 可以评估每一路音频的音量大小，并每隔一段时间抛出该事件回调，您可以根据音量大小在 UI 上做出相应的提示，比如 波形图 或 音量槽 。
要完成这个功能， 您需要先调用 enableAudioVolumeEvaluation 开启这个能力并设定事件抛出的时间间隔。
需要补充说明的是，无论当前房间中是否有人说话，SDK 都会按照您设定的时间间隔定时抛出此事件回调。

参数	描述
totalVolume	所有远端用户的总音量大小，取值范围 0 - 100。
userVolumes	是一个数组，用于承载所有正在说话的用户的音量大小，取值范围 0 - 100。

注意

userVolumes 为一个数组，对于数组中的每一个元素，当 userId 为空时表示本地麦克风采集的音量大小，当 userId 不为空时代表远端用户的音量大小。

onDeviceChange

onDeviceChange

void onDeviceChange	(const char* deviceId
	TRTCDeviceType type
	TRTCDeviceState state)

本地设备的通断状态发生变化（仅适用于桌面系统）

当本地设备（包括摄像头、麦克风以及扬声器）被插入或者拔出时，SDK 便会抛出此事件回调。

参数	描述
deviceId	设备 ID。
deviceType	设备类型。
state	通断状态，0：设备已添加；1：设备已被移除；1：设备已启用。

onAudioDeviceCaptureVolumeChanged

onAudioDeviceCaptureVolumeChanged

void onAudioDeviceCaptureVolumeChanged	(uint32_t volume
	bool muted)

当前麦克风的系统采集音量发生变化

在 Mac 或 Windows 这样的桌面操作系统上，用户可以在设置中心找到声音相关的设置面板，并设置麦克风的采集音量大小。
用户将麦克风的采集音量设置得越大，麦克风采集到的声音的原始音量也就会越大，反之就会越小。
在有些型号的键盘以及笔记本电脑上，用户还可以通过按下 `禁用麦克风` 按钮（图标是一个 `话筒` 叠加了一道代表禁用的斜线）来将麦克风静音。

当用户通过系统设置界面或者通过键盘上的快捷键设定操作系统的麦克风采集音量时，SDK 便会抛出此事件。

参数	描述
muted	麦克风是否被用户禁用了：true 被禁用，false 被启用。
volume	系统采集音量，取值范围 0 - 100，用户可以在系统的声音设置面板上进行拖拽调整。

注意

您需要调用 enableAudioVolumeEvaluation 接口并设定 interval 大于 0 开启次事件回调，设定 interval 等于 0 关闭此事件回调。

onAudioDevicePlayoutVolumeChanged

onAudioDevicePlayoutVolumeChanged

void onAudioDevicePlayoutVolumeChanged	(uint32_t volume
	bool muted)

当前系统的播放音量发生变化

在 Mac 或 Windows 这样的桌面操作系统上，用户可以在设置中心找到声音相关的设置面板，并设置系统的播放音量大小。
在有些型号的键盘以及笔记本电脑上，用户还可以通过按下“静音”按钮（图标是一个喇叭上叠加了一道代表禁用的斜线）来将系统静音。

当用户通过系统设置界面或者通过键盘上的快捷键设定操作系统的播放音量时，SDK 便会抛出此事件。

参数	描述
muted	系统是否被用户静音了：true 被静音，false 已恢复。
volume	系统播放音量，取值范围 0 - 100，用户可以在系统的声音设置面板上进行拖拽调整。

注意

您需要调用 enableAudioVolumeEvaluation 接口并设定 interval 大于 0 开启次事件回调，设定 interval 等于 0 关闭此事件回调。

onSystemAudioLoopbackError

onSystemAudioLoopbackError

void onSystemAudioLoopbackError	(TXLiteAVError errCode)

系统声音采集是否被成功开启的事件回调（仅适用于 Mac 系统）

在 Mac 系统上，您可以通过调用 startSystemAudioLoopback 为当前系统安装一个音频驱动，并让 SDK 通过该音频驱动捕获当前 Mac 系统播放出的声音。

当用于播片教学或音乐直播中，比如老师端可以使用此功能，让 SDK 能够采集老师所播放的电影中的声音，使同房间的学生端也能听到电影中的声音。

SDK 会将系统声音采集是否被成功开启的结果，通过本事件回调抛出，需要您关注参数中的错误码。

参数	描述
err	ERR_NULL 表示成功，其余值表示失败。

onTestMicVolume

onTestMicVolume

void onTestMicVolume	(uint32_t volume)

测试麦克风时的音量回调

当您调用 startMicDeviceTest 测试麦克风是否正常工作时，SDK 会不断地抛出此回调，参数中的 volume 代表当前麦克风采集到的音量大小。

如果在测试期间 volume 出现了大小波动的情况，说明麦克风状态健康；如果 volume 的数值始终是 0，说明麦克风的状态异常，需要提示用户进行更换。

参数	描述

volume	麦克风采集到的音量值，取值范围0 - 100。

onTestSpeakerVolume

onTestSpeakerVolume

void onTestSpeakerVolume	(uint32_t volume)

测试扬声器时的音量回调

当您调用 startSpeakerDeviceTest 测试扬声器是否正常工作时，SDK 会不断地抛出此回调。
参数中的 volume 代表的是 SDK 提交给系统扬声器去播放的声音的音量值大小，如果该数值持续变化，但用户反馈听不到声音，则说明扬声器状态异常。

参数	描述
volume	SDK 提交给扬声器去播放的声音的音量，取值范围0 - 100。

onRecvCustomCmdMsg

onRecvCustomCmdMsg

void onRecvCustomCmdMsg	(const char* userId
	int32_t cmdID
	uint32_t seq
	const uint8_t* message
	uint32_t messageSize)

收到自定义消息的事件回调

当房间中的某个用户使用 sendCustomCmdMsg 发送自定义 UDP 消息时，房间中的其他用户可以通过 onRecvCustomCmdMsg 事件回调接收到该条消息。

参数	描述
cmdID	命令 ID。
message	消息数据。

seq	消息序号。
userId	用户标识。

onMissCustomCmdMsg

onMissCustomCmdMsg

void onMissCustomCmdMsg	(const char* userId
	int32_t cmdID
	int32_t errCode
	int32_t missed)

自定义消息丢失的事件回调

当您使用 sendCustomCmdMsg 发送自定义 UDP 消息时，即使设置了可靠传输（reliable），也无法确 100% 不丢失，只是丢消息概率极低，能满足常规可靠性要求。
在发送端设置了可靠运输（reliable）后，SDK 都会通过此回调通知过去时间段内（通常为5s）传输途中丢失的自定义消息数量统计信息。

参数	描述
cmdID	命令 ID。
errCode	错误码。
missed	丢失的消息数量。
userId	用户标识。

注意

只有在发送端设置了可靠传输（reliable），接收方才能收到消息的丢失回调。

onRecvSEIMsg

onRecvSEIMsg

void onRecvSEIMsg	(const char* userId
	const uint8_t* message

	uint32_t messageSize)

收到 SEI 消息的回调

当房间中的某个用户使用 sendSEIMsg 借助视频数据帧发送 SEI 消息时，房间中的其他用户可以通过 onRecvSEIMsg 事件回调接收到该条消息。

参数	描述
message	数据。
userId	用户标识。

onStartPublishing

onStartPublishing

void onStartPublishing	(int err
	const char *errMsg)

开始向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 startPublishing 开始向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStopPublishing

onStopPublishing

void onStopPublishing	(int err
	const char *errMsg)

停止向腾讯云直播 CDN 上发布音视频流的事件回调

当您调用 stopPublishing 停止向腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。

版权所有：腾讯云计算（北京）有限责任公司

随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onStartPublishCDNStream

onStartPublishCDNStream

void onStartPublishCDNStream	(int errCode
	const char* errMsg)

开始向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 startPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

注意

当您收到成功的事件回调时，只是说明您的发布指令已经同步到腾讯云后台服务器，但如果目标 CDN 厂商的服务器不接收该条视频流，依然可能导致发布失败。

onStopPublishCDNStream

onStopPublishCDNStream

void onStopPublishCDNStream	(int errCode
	const char* errMsg)

停止向非腾讯云 CDN 上发布音视频流的事件回调

当您调用 stopPublishCDNStream 开始向非腾讯云直播 CDN 上发布音视频流时，SDK 会立刻将这一指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	0表示成功，其余值表示失败。
errMsg	具体错误原因。

onSetMixTranscodingConfig

onSetMixTranscodingConfig

void onSetMixTranscodingConfig	(int err
	const char* errMsg)

设置云端混流的排版布局和转码参数的事件回调

当您调用 setMixTranscodingConfig 调整云端混流的排版布局和转码参数时，SDK 会立刻将这一调整指令同步给云端服务器。随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
err	错误码：0表示成功，其余值表示失败。
errMsg	具体的错误原因。

onStartPublishMediaStream

onStartPublishMediaStream

void onStartPublishMediaStream	(const char* taskId
	int code
	const char* message
	void* extraInfo)

开始发布媒体流的事件回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述

code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	当请求成功时，TRTC 后台会在回调中提供给您这项任务的 taskId，后续您可以通过该 taskId 结合 updatePublishMediaStream 和 stopPublishMediaStream 进行更新和停止。

onUpdatePublishMediaStream

onUpdatePublishMediaStream

void onUpdatePublishMediaStream	(const char* taskId
	int code
	const char* message
	void* extraInfo)

更新媒体流的事件回调

当您调用媒体流发布接口 (updatePublishMediaStream) 开始向 TRTC 后台服务更新媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用媒体流发布接口 (updatePublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次更新请求。

onStopPublishMediaStream

onStopPublishMediaStream

void onStopPublishMediaStream	(const char* taskId

	int code
	const char* message
	void* extraInfo)

停止媒体流的事件回调

当您调用停止发布媒体流 (stopPublishMediaStream) 开始向 TRTC 后台服务停止媒体流时，SDK 会立刻将这一指令同步给云端服务器，随后 SDK 会收到来自云端服务器的处理结果，并将指令的执行结果通过本事件回调通知给您。

参数	描述
code	回调结果，0 表示成功，其余值表示失败。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体回调信息。
taskId	您调用停止发布媒体流 (stopPublishMediaStream) 时传入的 taskId，会通过此回调再带回给您，用于标识该回调属于哪一次停止请求。

onCdnStreamStateChanged

onCdnStreamStateChanged

void onCdnStreamStateChanged	(const char* cdnUrl
	int status
	int code
	const char* msg
	void* extraInfo)

RTMP/RTMPS 推流状态发生改变回调

当您调用 startPublishMediaStream 开始向 TRTC 后台服务发布媒体流时，SDK 会立刻将这一指令同步给云端服务。

若您在目标推流配置 (TRTCPublishTarget) 设置了向腾讯或者第三方 CDN 上发布音视频流的 URL 配置，则具体 RTMP 或者 RTMPS 推流状态将通过此回调同步给您。

参数	描述
cdnUrl	您调用 startPublishMediaStream 时通过目标推流配置 (TRTCPublishTarget) 传入的 url，在

	推流状态变更时，会通过此回调同步给您。
code	推流结果，0 表示成功，其余值表示出错。
extraInfo	扩展信息字段，个别错误码可能会带额外的信息帮助定位问题。
message	具体推流信息。
status	推流状态。 0：推流未开始或者已结束。在您调用 stopPublishMediaStream 时会返回该状态。 1：正在连接 TRTC 服务器和 CDN 服务器。若无法立刻成功，TRTC 后台服务会多次重试尝试推流，并返回该状态（5s回调一次）。如成功进行推流，则进入状态 2；如服务器出错或 60 秒内未成功推流，则进入状态 4。 2：CDN 推流正在进行。在成功推流后，会返回该状态。 3：TRTC 服务器和 CDN 服务器推流中断，正在恢复。当 CDN 出现异常，或推流短暂中断时，TRTC 后台服务会自动尝试恢复推流，并返回该状态（5s回调一次）。如成功恢复推流，则进入状态 2；如服务器出错或 60 秒内未成功恢复，则进入状态 4。 4：TRTC 服务器和 CDN 服务器推流中断，且恢复或连接超时。即此时推流失败，你可以再次调用 updatePublishMediaStream 尝试推流。 5：正在断开 TRTC 服务器和 CDN 服务器。在您调用 stopPublishMediaStream 时，TRTC 后台服务会依次同步状态 5 和状态 0。

onScreenCaptureStarted

onScreenCaptureStarted

屏幕分享开启的事件回调

当您通过 startScreenCapture 等相关接口启动屏幕分享时，SDK 便会抛出此事件回调。

onScreenCapturePaused

onScreenCapturePaused

void onScreenCapturePaused	(int reason)

屏幕分享暂停的事件回调

当您通过 pauseScreenCapture 暂停屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	原因。 0：用户主动暂停。

	1：注意此字段的含义在 MAC 和 Windows 平台有稍微差异。屏幕窗口不可见暂停（Mac）。表示设置屏幕分享参数导致的暂停（Windows）。 2：表示屏幕分享窗口被最小化导致的暂停（仅 Windows）。 3：表示屏幕分享窗口被隐藏导致的暂停（仅 Windows）。

onScreenCaptureResumed

onScreenCaptureResumed

void onScreenCaptureResumed	(int reason)

屏幕分享恢复的事件回调

当您通过 resumeScreenCapture 恢复屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	恢复原因。 0：用户主动恢复。 1：注意此字段的含义在 MAC 和 Windows 平台有稍微差异。屏幕窗口恢复可见从而恢复分享（Mac）。屏幕分享参数设置完毕后自动恢复（Windows）。 2：表示屏幕分享窗口从最小化被恢复（仅 Windows）。 3：表示屏幕分享窗口从隐藏被恢复（仅 Windows）。

onScreenCaptureStoped

onScreenCaptureStoped

void onScreenCaptureStoped	(int reason)

屏幕分享停止的事件回调

当您通过 stopScreenCapture 停止屏幕分享时，SDK 便会抛出此事件回调。

参数	描述
reason	停止原因，0：用户主动停止；1：屏幕窗口关闭导致停止；2：表示屏幕分享的显示屏状态变更（如接口被拔出、投影模式变更等）。

onScreenCaptureCovered

onScreenCaptureCovered

屏幕分享的目标窗口被遮挡的事件回调（仅适用于 Windows 操作系统）

当屏幕分享的目标窗口被遮挡无法正常捕获时，SDK 会抛出此事件回调，你可以在捕获到该事件回调后，通过 UI 上的一些变化来提示用户移开遮盖窗口。

onLocalRecordBegin

onLocalRecordBegin

void onLocalRecordBegin	(int errCode
	const char* storagePath)

本地录制任务已经开始的事件回调

当您调用 startLocalRecording 启动本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务是否已经顺利启动。

参数	描述
errCode	状态码。 0：录制任务启动成功。 -1：内部错误导致录制任务启动失败。 -2：文件后缀名有误（比如不支持的录制格式）。 -6：录制已经启动，需要先停止录制。 -7：录制文件已存在，需要先删除文件。 -8：录制目录无写入权限，请检查目录权限问题。
storagePath	录制文件存储路径。

onLocalRecording

onLocalRecording

void onLocalRecording	(long duration
	const char* storagePath)

本地录制任务正在进行中的进展事件回调

当您调用 startLocalRecording 成功启动本地媒体录制任务后，SDK 会定时地抛出本事件回调。
您可通过捕获该事件回调来获知录制任务的健康状况。

版权所有：腾讯云计算（北京）有限责任公司

您可以在 startLocalRecording 时设定本事件回调的抛出间隔。

参数	描述
duration	已经录制的累计时长，单位毫秒。
storagePath	录制文件存储路径。

onLocalRecordFragment

onLocalRecordFragment

void onLocalRecordFragment	(const char* storagePath)

本地录制分片的事件回调

当您开启分片录制时，每完成一个分片，都会通过此接口回调。

参数	描述
storagePath	分片文件存储路径。

onLocalRecordComplete

onLocalRecordComplete

void onLocalRecordComplete	(int errCode
	const char* storagePath)

本地录制任务已经结束的事件回调

当您调用 stopLocalRecording 停止本地媒体录制任务时，SDK 会抛出该事件回调，用于通知您录制任务的最终结果。

参数	描述
errCode	状态码。 0：结束录制任务成功。 -1：录制失败。 -2：切换分辨率或横竖屏导致录制结束。 -3：录制时间太短，或未采集到任何视频或音频数据，请检查录制时长，或是否已开启音、视频采集。

storagePath	录制文件存储路径。

onSnapshotComplete

onSnapshotComplete

void onSnapshotComplete	(const char* userId
	TRTCVideoStreamType type
	char* data
	uint32_t length
	uint32_t width
	uint32_t height
	TRTCVideoPixelFormat format)

本地截图完成的事件回调

参数	描述
bmp	截图结果，如果 bmp 为 null 代表本次截图操作失败。
data	截图数据，为 nullptr 表示截图失败。
format	截图数据格式，目前只支持 TRTCVideoPixelFormat_BGRA32。
height	截图画面的高度。
length	截图数据长度，对于BGRA32而言，length = width * height * 4。
type	视频流类型。
userId	用户标识，如果 userId 为空字符串，则代表截取的是本地画面。
width	截图画面的宽度。

注意

全平台 C++ 接口和 Java 接口在参数上是不一样的，C++ 接口用 7 个参数描述一个截图画面，Java 接口只用一个 Bitmap 描述一个截图画面。

版权所有：腾讯云计算（北京）有限责任公司

onUserEnter

onUserEnter

void onUserEnter	(const char* userId)

有主播加入当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserEnterRoom 替代之。

onUserExit

onUserExit

void onUserExit	(const char* userId
	int reason)

有主播离开当前房间（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onRemoteUserLeaveRoom 替代之。

onAudioEffectFinished

onAudioEffectFinished

void onAudioEffectFinished	(int effectId
	int code)

音效播放已结束（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXAudioEffectManager 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onPlayBGMBegin

onPlayBGMBegin

void onPlayBGMBegin	(TXLiteAVError errCode)

开始播放背景音乐（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXMusicPlayObserver 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onPlayBGMProgress

onPlayBGMProgress

void onPlayBGMProgress	(uint32_t progressMS
	uint32_t durationMS)

背景音乐的播放进度回调（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXMusicPlayObserver 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onPlayBGMComplete

onPlayBGMComplete

void onPlayBGMComplete	(TXLiteAVError errCode)

背景音乐播放已经结束（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 ITXMusicPlayObserver 接口替代之。
新的接口中不再区分背景音乐和音效，而是统一用 startPlayMusic 取代之。

onSpeedTest

onSpeedTest

void onSpeedTest	(const TRTCSpeedTestResult& currentResult
	uint32_t finishedCount
	uint32_t totalCount)

服务器测速的结果回调（已废弃）

@deprecated 新版本开始不推荐使用，建议使用 onSpeedTestResult 接口替代之。

onRenderVideoFrame

onRenderVideoFrame

void onRenderVideoFrame	(const char* userId
	TRTCVideoStreamType streamType
	TRTCVideoFrame* frame)

自定义视频渲染回调

当您设置了本地或者远端的视频自定义渲染回调之后，SDK 就会将原本要交给渲染控件进行渲染的视频帧通过此回调接口抛送给您，以便于您进行自定义渲染。

参数	描述
frame	待渲染的视频帧信息。
streamType	频流类型：主路（Main）一般用于承载摄像头画面，辅路（Sub）一般用于承载屏幕分享画面。
userId	视频源的 userId，如果是本地视频回调（setLocalVideoRenderDelegate），该参数可以忽略。

onGLContextCreated

onGLContextCreated

SDK 内部 OpenGL 环境已经创建的通知

onProcessVideoFrame

onProcessVideoFrame

int onProcessVideoFrame	(TRTCVideoFrame *srcFrame
	TRTCVideoFrame *dstFrame)

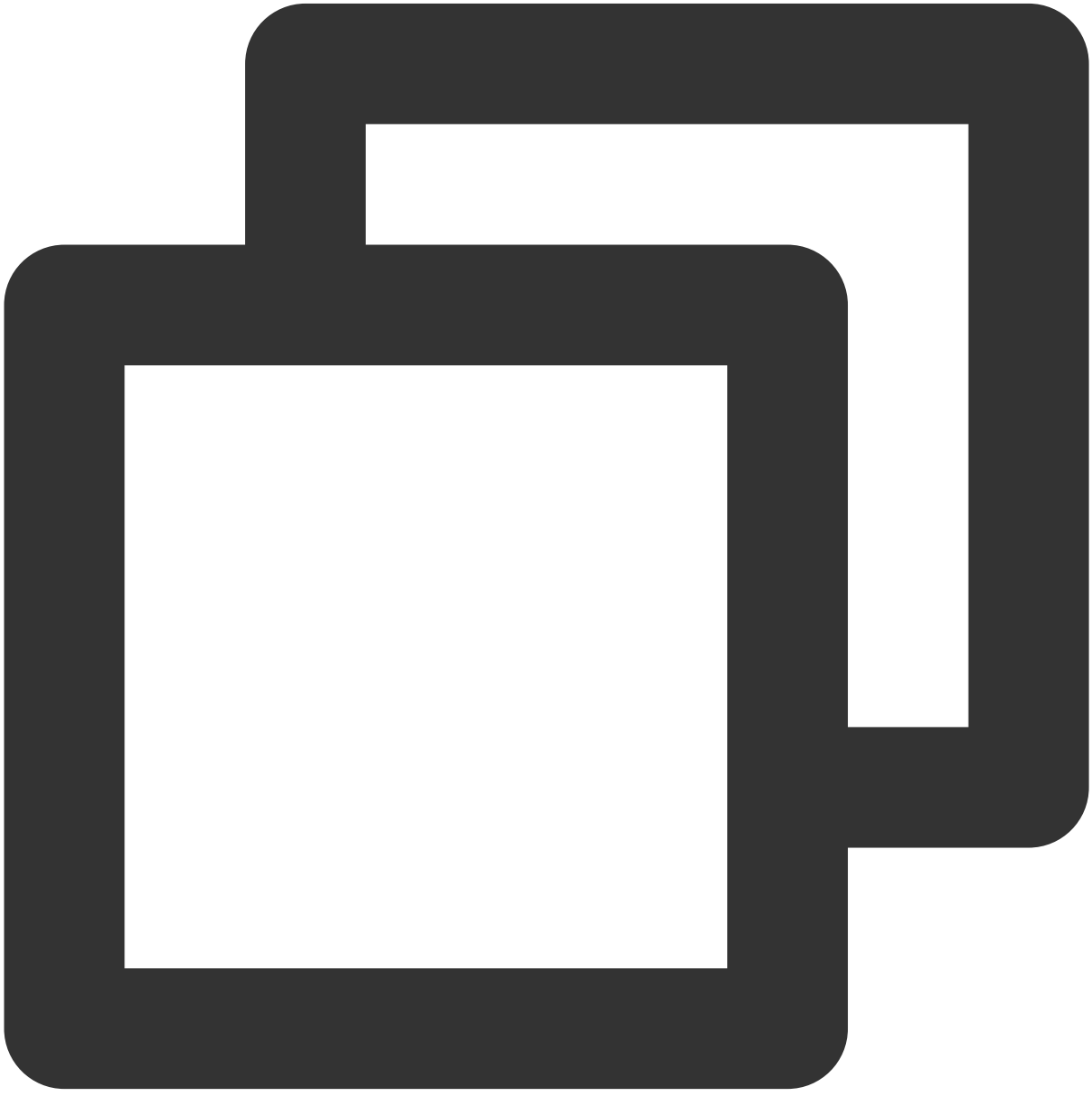
用于对接第三方美颜组件的视频处理回调

如果您选购了第三方美颜组件，就需要在 TRTCCloud 中设置第三方美颜回调，之后 TRTC 就会将原本要进行预处理的视频帧通过此回调接口抛送给您。

版权所有：腾讯云计算（北京）有限责任公司

之后您就可以将 TRTC 抛出的视频帧交给第三方美颜组件进行图像处理，由于抛出的数据是可读且可写的，因此第三方美颜的处理结果也可以同步给 TRTC 进行后续的编码和发送。

情况一：美颜组件自身会产生新的纹理。
如果您使用的美颜组件会在处理图像的过程中产生一帧全新的纹理（用于承载处理后的图像），那请您在回调函数中将 dstFrame.textureId 设置为新纹理的 ID：

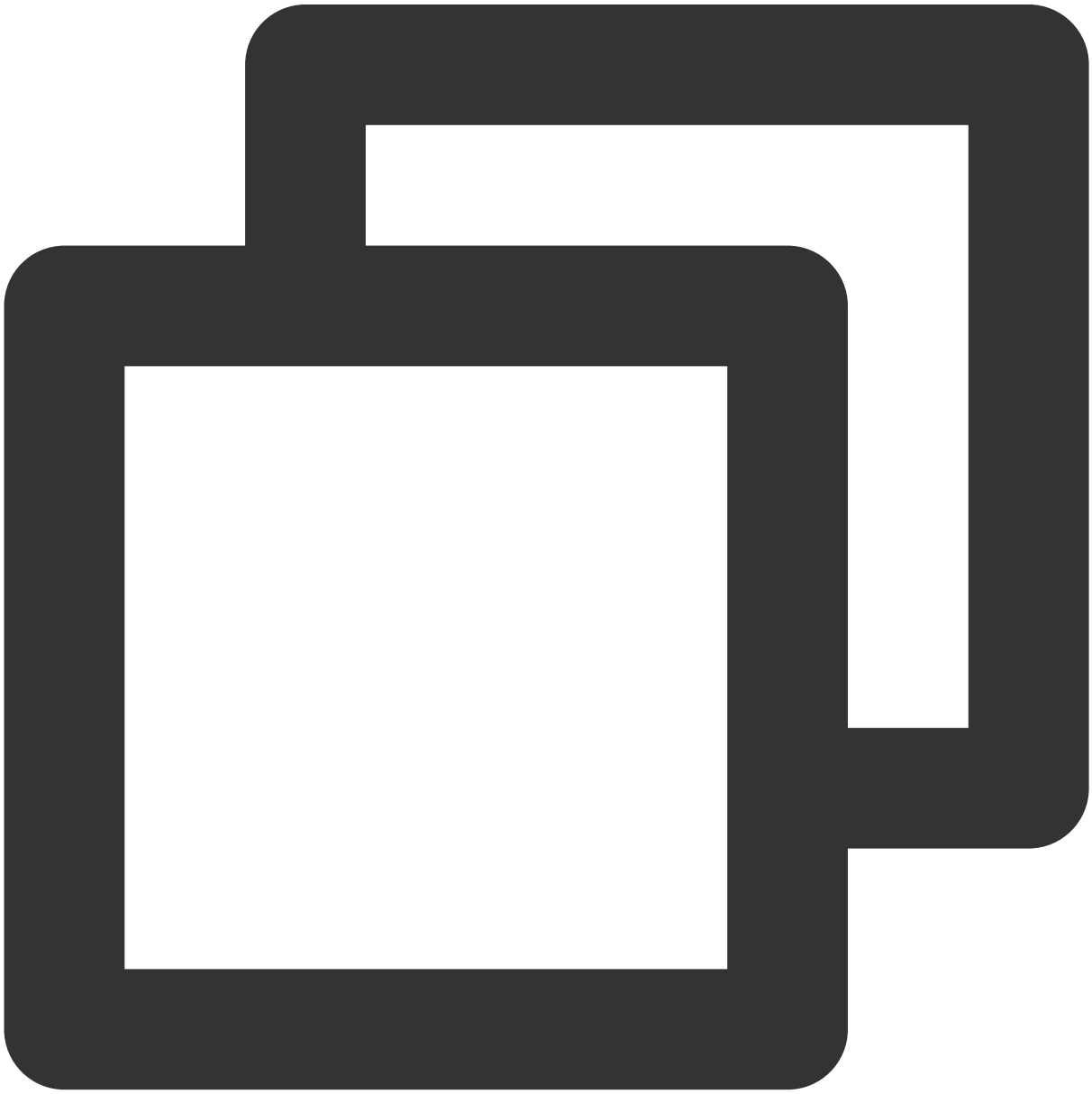


```
int onProcessVideoFrame(TRTCVideoFrame * srcFrame, TRTCVideoFrame *dstFrame) {
    dstFrame->textureId = mFURenderer.onDrawFrameSingleInput(srcFrame->textureId);
    return 0;
```

```
}
```

情况二：美颜组件需要您提供目标纹理。

如果您使用的第三方美颜模块并不生成新的纹理，而是需要您设置给该模块一个输入纹理和一个输出纹理，则可以考虑如下方案：



```
int onProcessVideoFrame(TRTCVideoFrame *srcFrame, TRTCVideoFrame *dstFrame) {
    thirdparty_process(srcFrame->textureId, srcFrame->width, srcFrame->height, dstF
    return 0;
}
```

参数	描述
dstFrame	用于接收第三方美颜处理过的视频画面。
srcFrame	用于承载 TRTC 采集到的摄像头画面。

注意

目前仅支持 OpenGL 纹理方案（ PC 仅支持 TRTCVideoBufferType_Buffer 格式）。

onGLContextDestroy

onGLContextDestroy

SDK 内部 OpenGL 环境被销毁的通知

onCapturedAudioFrame

onCapturedAudioFrame

void onCapturedAudioFrame	(TRTCAudioFrame *frame)

本地采集并经过音频模块前处理后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理(ANS、AEC、AGC）之后的数据，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的音频数据已经经过了前处理(ANS、AEC、AGC），但**不包含**背景音、音效、混响等前处理效果，延迟较低。

版权所有：腾讯云计算（北京）有限责任公司

onLocalProcessedAudioFrame

onLocalProcessedAudioFrame

void onLocalProcessedAudioFrame	(TRTCAudioFrame *frame)

本地采集并经过音频模块前处理、音效处理和混 BGM 后的音频数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把刚采集到并经过前处理、音效处理和混 BGM 之后的数据，在最终进行网络编码之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

特殊说明：

您可以通过设置接口中的 `TRTCAudioFrame.extraData` 字段，达到传输信令的目的。由于音频帧头部的数据块不能太大，建议您写入 `extraData` 时，尽量将信令控制在几个字节的大小，如果超过 100 个字节，写入的数据不会被发送。

房间内其他用户可以通过 ITRTCAudioFrameCallback 中的 `onRemoteUserAudioFrame` 中的 `TRTCAudioFrame.extraData` 字段回调接收数据。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的数据已经经过了前处理(ANS、AEC、AGC）、音效和混 BGM 处理，声音的延迟相比于 onCapturedAudioFrame 要高一些。

onPlayAudioFrame

onPlayAudioFrame

void onPlayAudioFrame	(TRTCAudioFrame *frame

	const char* userId)

混音前的每一路远程用户的音频数据

当您设置完音频数据自定义回调之后，SDK 内部会把远端的每一路原始数据，在最终混音之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。
userId	用户标识。

注意

此接口回调出的音频数据是只读的，不支持修改。

onMixedPlayAudioFrame

onMixedPlayAudioFrame

void onMixedPlayAudioFrame	(TRTCAudioFrame *frame)

将各路待播放音频混合之后并在最终提交系统播放之前的数据回调

当您设置完音频数据自定义回调之后，SDK 内部会把各路待播放的音频混合之后的音频数据，在提交系统播放之前，以 PCM 格式的形式通过本接口回调给您。

此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。

由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。

以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 请不要在此回调函数中做任何耗时操作，由于 SDK 每隔 20ms 就要处理一帧音频数据，如果您的处理时间超过 20ms，就会导致声音异常。
2. 此接口回调出的音频数据是可读写的，也就是说您可以在回调函数中同步修改音频数据，但请保证处理耗时。
3. 此接口回调出的是对各路待播放音频数据的混合，但其中并不包含耳返的音频数据。

onMixedAllAudioFrame

onMixedAllAudioFrame

void onMixedAllAudioFrame	(TRTCAudioFrame *frame)

SDK 所有音频混合后的音频数据（包括采集到的和待播放的）

当您设置完音频数据自定义回调之后，SDK 内部会把所有采集到的和待播放的音频数据混合起来，以 PCM 格式的形式通过本接口回调给您，便于您进行自定义录制。
此接口回调出的音频时间帧长固定为0.02s，格式为 PCM 格式。
由时间帧长转化为字节帧长的公式为 `采样率 × 时间帧长 × 声道数 × 采样点位宽` 。
以 TRTC 默认的音频录制格式48000采样率、单声道、16采样点位宽为例，字节帧长为 `48000 × 0.02s × 1 × 16bit = 15360bit = 1920字节` 。

参数	描述
frame	PCM 格式的音频数据帧。

注意

1. 此接口回调出的是SDK所有音频数据的混合数据，包括：经过 3A 前处理、特效叠加以及背景音乐混音后的本地音频，所有远端音频，但不包括耳返音频。
2. 此接口回调出的音频数据不支持修改。

onLog

onLog

void onLog	(const char* log
	TRTCLogLevel level
	const char* module)

本地 LOG 的打印回调

如果您希望捕获 SDK 的本地日志打印行为，可以通过设置日志回调，让 SDK 将要打印的日志都通过本回调接口抛送给您。

参数	描述
level	日志等级请参见 TRTC_LOG_LEVEL。
log	日志内容。
module	保留字段，暂无具体意义，目前为固定值 TXLiteAVSDK。

ITRTCStatistics

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 音视频统计指标（只读）

Function: TRTC SDK 会以两秒钟一次的频率向您汇报当前实时的音视频指标（帧率、码率、卡顿情况等）

ITRTCStatistics

结构体类型

函数列表	描述
TRTCLocalStatistics	本地的音视频统计指标
TRTCRemoteStatistics	远端的音视频统计指标
TRTCStatistics	网络和性能的汇总统计指标

TRTCLocalStatistics

TRTCLocalStatistics

本地的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，即每秒钟新产生音频数据的多少，单位 Kbps。
audioCaptureState	【字段含义】音频设备采集状态（用于检测音频外设的健康度）。 0：采集设备状态正常；1：检测到长时间静音；2：检测到破音；3：检测到声音异常间断。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
frameRate	【字段含义】本地视频的帧率，即每秒钟会有多少视频帧，单位：FPS。
height	【字段含义】本地视频的高度，单位 px。

streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
videoBitrate	【字段含义】本地视频的码率，即每秒钟新产生视频数据的多少，单位 Kbps。
width	【字段含义】本地视频的宽度，单位 px。

TRTCRemoteStatistics

TRTCRemoteStatistics

远端的音视频统计指标

枚举类型	描述
audioBitrate	【字段含义】本地音频的码率，单位 Kbps
audioBlockRate	【字段含义】音频播放卡顿率，单位 (%) 音频播放卡顿率（audioBlockRate） = 音频播放的累计卡顿时长（audioTotalBlockTime） / 音频播放的总时长
audioPacketLoss	【字段含义】音频流的总丢包率（%）。 audioPacketLoss 代表音频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 audioPacketLoss 越小越好，丢包率为0即表示该路音频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 audioPacketLoss != 0 的情况，说明该路音频流在“云端=>观众”这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
audioSampleRate	【字段含义】本地音频的采样率，单位 Hz。
audioTotalBlockTime	【字段含义】音频播放的累计卡顿时长，单位 ms
finalLoss	【字段含义】该路音视频流的总丢包率（%）。已废弃，不推荐使用；建议使用 audioPacketLoss、videoPacketLoss 替代。
frameRate	【字段含义】远端视频的帧率，单位：FPS。
height	【字段含义】远端视频的高度，单位 px。
jitterBufferDelay	【字段含义】播放延迟，单位 ms 为了避免网络抖动和网络包乱序导致的声音和画面卡顿，TRTC 会在播放端管理一个播放缓冲区，用于对接收到的网络数据包进行整理，该缓冲区的大小会根据当前的网络质量进行自适应调整，该缓冲区的大小折算成以毫秒为单位的时间长度，也就是 jitterBufferDelay。

point2PointDelay	【字段含义】端到端延迟，单位 ms point2PointDelay 代表 “主播=>云端=>观众” 的延迟，更准确地说，它代表了“采集=>编码=>网络传输=>接收=>缓冲=>解码=>播放” 全链路的延迟。 point2PointDelay 需要本地和远端的 SDK 均为 8.5 及以上的版本才生效，若远端用户为 8.5 以前的版本，此数值会一直为0，代表无意义。
remoteNetworkRTT	【字段含义】该用户从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。 该数值越小越好：如果 remoteNetworkRTT < 50ms，意味着较低的音视频通话延迟；如果 remoteNetworkRTT > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，remoteNetworkRTT 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 remoteNetworkUpRTT 和 remoteNetworkDownRTT
remoteNetworkUplinkLoss	【字段含义】该用户从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 remoteNetworkUplinkLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 remoteNetworkUplinkLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。
streamType	【字段含义】视频流类型（高清大画面|低清小画面|辅流画面）。
userId	【字段含义】用户 ID
videoBitrate	【字段含义】远端视频的码率，单位 Kbps。
videoBlockRate	【字段含义】视频播放卡顿率，单位 (%) 视频播放卡顿率（videoBlockRate） = 视频播放的累计卡顿时长（videoTotalBlockTime） / 视频播放的总时长。
videoPacketLoss	【字段含义】该路视频流的总丢包率（%）。 videoPacketLoss 代表该路视频流历经 `主播>云端>观众` 这样一条完整的传输链路后，最终在观众端统计到的丢包率。 videoPacketLoss 越小越好，丢包率为0即表示该路视频流的所有数据均已经完整地到达了观众端。 如果出现了 downLoss == 0 但 videoPacketLoss != 0 的情况，说明该路视频流在 `云端>观众` 这一段链路上没有出现丢包，但是在 `主播>云端` 这一段链路上出现了不可恢复的丢包。
videoTotalBlockTime	【字段含义】视频播放的累计卡顿时长，单位 ms
width	【字段含义】远端视频的宽度，单位 px。

版权所有：腾讯云计算（北京）有限责任公司

TRTCStatistics

TRTCStatistics

网络和性能的汇总统计指标

枚举类型	描述
appCpu	【字段含义】当前应用的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。
appMemoryUsageInMB	【字段含义】当前应用的内存占用，单位 (MB)。
downLoss	【字段含义】从云端到 SDK 的下行丢包率，单位 (%) 该数值越小越好，如果 downLoss 为 0%，则意味着下行链路的网络质量很好，从云端接收的数据包基本不发生丢失。 如果 downLoss 为 30%，则意味着云端向 SDK 传输的音视频数据包中，会有 30% 丢失在传输链路中。
gatewayRtt	【字段含义】从 SDK 到本地路由器的往返时延，单位 ms 该数值代表从 SDK 发送一个网络包到本地路由器网关，再从网关回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 【SDK>网关>SDK【 的总耗时。 该数值越小越好：如果 gatewayRtt < 50ms，意味着较低的音视频通话延迟；如果 gatewayRtt > 200ms，则意味着较高的音视频通话延迟。 当网络类型为蜂窝网时，该值无效。
localStatisticsArray	【字段含义】本地的音视频统计信息 由于本地可能有三路音视频流（即高清大画面，低清小画面，以及辅流画面），因此本地的音视频统计信息是一个数组。
localStatisticsArraySize	【字段含义】数组 localStatisticsArray 的大小
receivedBytes	【字段含义】总接收字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
remoteStatisticsArray	【字段含义】远端的音视频统计信息 因为同时可能有多个远端用户，而且每个远端用户同时可能有多路音视频流（即高清大画面，低清小画面，以及辅流画面），因此远端的音视频统计信息是一个数组。
remoteStatisticsArraySize	【字段含义】数组 remoteStatisticsArray 的大小。
rtt	【字段含义】从 SDK 到云端的往返延时，单位 ms 该数值代表从 SDK 发送一个网络包到云端，再从云端回送一个网络包到 SDK 的总计耗时，也就是一个网络包经历 “SDK=>云端=>SDK” 的总耗时。

	该数值越小越好：如果 rtt < 50ms，意味着较低的音视频通话延迟；如果 rtt > 200ms，则意味着较高的音视频通话延迟。 需要特别解释的是，rtt 代表 “SDK=>云端=>SDK” 的总耗时，所以不需要区分 upRtt 和 downRtt。
sentBytes	【字段含义】总发送字节数（包含信令数据和音视频数据），单位：字节数（Bytes）。
systemCpu	【字段含义】当前系统的 CPU 使用率，单位 (%)，Android 8.0 以上不支持。
systemMemoryInMB	【字段含义】当前系统的内存，单位 (MB)。
systemMemoryUsageInMB	【字段含义】当前系统内存占用，单位 (MB)，iOS/MAC 不支持(MB)。
upLoss	【字段含义】从 SDK 到云端的上行丢包率，单位 (%) 该数值越小越好，如果 upLoss 为 0%，则意味着上行链路的网络质量很好，上传到云端的数据包基本不发生丢失。 如果 upLoss 为 30%，则意味着 SDK 向云端发送的音视频数据包中，会有 30% 丢失在传输链路中。

ITXAudioEffectManager

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: 背景音乐、短音效和人声特效的管理类

Function: 用于对背景音乐、短音效和人声特效进行设置的管理类

ITXAudioEffectManager

ITXMusicPreloadObserver

函数列表	描述
onLoadProgress	背景音乐预加载进度
onLoadError	背景音乐预加载出错

ITXMusicPlayObserver

函数列表	描述
onStart	背景音乐开始播放
onPlayProgress	背景音乐的播放进度
onComplete	背景音乐已经播放完毕

ITXAudioEffectManager

函数列表	描述
enableVoiceEarMonitor	开启耳返
setVoiceEarMonitorVolume	设置耳返音量

setVoiceReverbType	设置人声的混响效果
setVoiceChangerType	设置人声的变声特效
setVoiceCaptureVolume	设置语音音量
setVoicePitch	设置语音音调
setMusicObserver	设置背景音乐的事件回调接口
startPlayMusic	开始播放背景音乐
stopPlayMusic	停止播放背景音乐
pausePlayMusic	暂停播放背景音乐
resumePlayMusic	恢复播放背景音乐
setAllMusicVolume	设置所有背景音乐的本地音量和远端音量的大小
setMusicPublishVolume	设置某一首背景音乐的远端音量的大小
setMusicPlayoutVolume	设置某一首背景音乐的本地音量的大小
setMusicPitch	调整背景音乐的音调高低
setMusicSpeedRate	调整背景音乐的变速效果
getMusicCurrentPosInMS	获取背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS	获取背景音乐的总时长（单位：毫秒）
seekMusicToPosInTime	设置背景音乐的播放进度（单位：毫秒）
setMusicScratchSpeedRate	调整搓碟的变速效果
setPreloadObserver	设置预加载事件回调
preloadMusic	预加载背景音乐
getMusicTrackCount	获取背景音乐的音轨数量
setMusicTrack	指定背景音乐的播放音轨

结构体类型

函数列表	描述

版权所有：腾讯云计算（北京）有限责任公司

AudioMusicParam	背景音乐的播放控制信息

枚举类型

枚举类型	描述
TXVoiceReverbType	混响特效
TXVoiceChangerType	变声特效

onLoadProgress

onLoadProgress

void onLoadProgress	(int id
	int progress)

背景音乐预加载进度

onLoadError

onLoadError

void onLoadError	(int id
	int errorCode)

背景音乐预加载出错

参数	描述
errorCode	-4001:打开文件失败，如音频数据无效，FFMPEG 协议未找到等；-4002:解码失败，如音频文件损坏，网络音频文件服务器无法访问等；-4003:预加载数量超上限，请先调用 stopPlayMusic 释放无用的预加载；-4005:非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件；-4006:非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件；-4007:无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏；-4008:格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，

	aac，m4a，wav，ogg，mp4，mkv】，桌面端支持 【mp3，aac，m4a，wav，mp4，mkv】。

onStart

onStart

void onStart	(int id
	int errCode)

背景音乐开始播放

在背景音乐开始播放成功或者失败时调用。

参数	描述
errCode	错误码。0: 开始播放成功；-4001:打开文件失败，如音频数据无效，FFMPEG 协议未找到等；-4005:非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件；-4006:非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件；-4007:无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏；-4008:格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，aac，m4a，wav，ogg，mp4，mkv】，桌面端支持 【mp3，aac，m4a，wav，mp4，mkv】。
id	音乐 ID。

onPlayProgress

onPlayProgress

void onPlayProgress	(int id
	long curPtsMS
	long durationMS)

背景音乐的播放进度

onComplete

onComplete

void onComplete	(int id
	int errCode)

背景音乐已经播放完毕

在背景音乐播放完毕或播放错误时调用。

参数	描述
errCode	错误码。0: 播放结束；-4002: 解码失败，如音频文件损坏，网络音频文件服务器无法访问等。
id	音乐 ID。

enableVoiceEarMonitor

enableVoiceEarMonitor

void enableVoiceEarMonitor	(bool enable)

开启耳返

主播开启耳返后，可以在耳机里听到麦克风采集到的自己发出的声音，该特效适用于主播唱歌的应用场景中。

需要您注意的是，由于蓝牙耳机的硬件延迟非常高，所以在主播佩戴蓝牙耳机时无法开启此特效，请尽量在用户界面上提示主播佩戴有线耳机。
同时也需要注意，并非所有的手机开启此特效后都能达到优秀的耳返效果，我们已经对部分耳返效果不佳的手机屏蔽了该特效。

参数	描述
enable	true：开启；false：关闭。

注意

仅在主播佩戴耳机时才能开启此特效，同时请您提示主播佩戴有线耳机。

setVoiceEarMonitorVolume

setVoiceEarMonitorVolume

void setVoiceEarMonitorVolume	(int volume)

设置耳返音量

通过该接口您可以设置耳返特效中声音的音量大小。

参数	描述
volume	音量大小，取值范围为 0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoiceReverbType

setVoiceReverbType

void setVoiceReverbType	(TXVoiceReverbType type)

设置人声的混响效果

通过该接口您可以设置人声的混响效果，具体特效请参见枚举定义 TXVoiceReverbType。

注意

设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceChangerType

setVoiceChangerType

void setVoiceChangerType	(TXVoiceChangerType type)

设置人声的变声特效

通过该接口您可以设置人声的变声特效，具体特效请参见枚举定义 TXVoiceChangeType。

注意

设置的效果在退出房间后会自动失效，如果下次进房还需要对应特效，需要调用此接口再次进行设置。

setVoiceCaptureVolume

setVoiceCaptureVolume

版权所有：腾讯云计算（北京）有限责任公司

void setVoiceCaptureVolume	(int volume)

设置语音音量

该接口可以设置语音音量的大小，一般配合音乐音量的设置接口 setAllMusicVolume 协同使用，用于调谐语音和音乐在混音前各自的音量占比。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setVoicePitch

setVoicePitch

void setVoicePitch	(double pitch)

设置语音音调

该接口可以设置语音音调，用于实现变调不变速的目的。

参数	描述
pitch	音调，取值范围为-1.0f~1.0f，默认值：0.0f。

setMusicObserver

setMusicObserver

void setMusicObserver	(int musicId
	ITXMusicPlayObserver* observer)

设置背景音乐的事件回调接口

请在播放背景音乐之前使用该接口设置播放事件回调，以便感知背景音乐的播放进度。

参数	描述

musicId	音乐 ID。
observer	具体参考 ITXMusicPlayObserver 中定义接口。

注意

1. 如果该 ID 不需要使用，observer 可设置为 NULL 彻底释放。

startPlayMusic

startPlayMusic

void startPlayMusic	(AudioMusicParam musicParam)

开始播放背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
musicParam	音乐参数。

注意

1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

stopPlayMusic

stopPlayMusic

void stopPlayMusic	(int id)

停止播放背景音乐

参数	描述
id	音乐 ID。

pausePlayMusic

pausePlayMusic

void pausePlayMusic	(int id)

暂停播放背景音乐

参数	描述
id	音乐 ID。

resumePlayMusic

resumePlayMusic

void resumePlayMusic	(int id)

恢复播放背景音乐

参数	描述
id	音乐 ID。

setAllMusicVolume

setAllMusicVolume

void setAllMusicVolume	(int volume)

设置所有背景音乐的本地音量和远端音量的大小

该接口可以设置所有背景音乐的本地音量和远端音量。
本地音量：即主播本地可以听到的背景音乐的音量大小。
远端音量：即观众端可以听到的背景音乐的音量大小。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPublishVolume

setMusicPublishVolume

void setMusicPublishVolume	(int id
	int volume)

设置某一首背景音乐的远端音量的大小

该接口可以细粒度地控制每一首背景音乐的远端音量，也就是观众端可听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。
volume	音量大小，取值范围为0 - 100；默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

setMusicPlayoutVolume

setMusicPlayoutVolume

void setMusicPlayoutVolume	(int id
	int volume)

设置某一首背景音乐的本地音量的大小

该接口可以细粒度地控制每一首背景音乐的本地音量，也就是主播本地可以听到的背景音乐的音量大小。

参数	描述
id	音乐 ID。
volume	音量大小，取值范围为0 - 100，默认值：60。

注意

如果将 volume 设置成 100 之后感觉音量还是太小，可以将 volume 最大设置成 150，但超过 100 的 volume 会有爆音的风险，请谨慎操作。

版权所有：腾讯云计算（北京）有限责任公司

setMusicPitch

setMusicPitch

void setMusicPitch	(int id
	float pitch)

调整背景音乐的音调高低

参数	描述
id	音乐 ID。
pitch	音调，默认值是0.0f，范围是：[-1 ~ 1] 之间的浮点数。

setMusicSpeedRate

setMusicSpeedRate

void setMusicSpeedRate	(int id
	float speedRate)

调整背景音乐的变速效果

参数	描述
id	音乐 ID。
speedRate	速度，默认值是1.0f，范围是：[0.5 ~ 2] 之间的浮点数。

getMusicCurrentPosInMS

getMusicCurrentPosInMS

long getMusicCurrentPosInMS	(int id)

获取背景音乐的播放进度（单位：毫秒）

参数	描述

id	音乐 ID。

返回值说明：

成功返回当前播放时间，单位：毫秒，失败返回 -1。

getMusicDurationInMS

getMusicDurationInMS

long getMusicDurationInMS	(char* path)

获取背景音乐的总时长（单位：毫秒）

参数	描述
path	音乐文件路径。

返回值说明：

成功返回时长，失败返回 -1。

seekMusicToPosInTime

seekMusicToPosInTime

void seekMusicToPosInTime	(int id
	int pts)

设置背景音乐的播放进度（单位：毫秒）

参数	描述
id	音乐 ID。
pts	单位: 毫秒。

注意

请尽量避免过度频繁地调用该接口，因为该接口可能会再次读写音乐文件，耗时稍高。

因此，当用户拖拽音乐的播放进度条时，请在用户完成拖拽操作后再调用本接口。

因为 UI 上的进度条控件往往会以很高的频率反馈用户的拖拽进度，如不做频率限制，会导致较差的用户体验。

setMusicScratchSpeedRate

setMusicScratchSpeedRate

void setMusicScratchSpeedRate	(int id
	float scratchSpeedRate)

调整搓碟的变速效果

参数	描述
id	音乐 ID。
scratchSpeedRate	搓碟速度，默认值是1.0f，范围是：[-12.0 ~ 12.0] 之间的浮点数, 速度值正/负表示方向正/反，绝对值大小表示速度快慢。

注意

前置条件 preloadMusic 成功。

setPreloadObserver

setPreloadObserver

void setPreloadObserver	(ITXMusicPreloadObserver* observer)

设置预加载事件回调

请在预加载背景音乐之前使用该接口设置回调，以便感知背景音乐的预加载进度。

参数	描述
observer	具体参考 ITXMusicPreloadObserver 中定义接口。

preloadMusic

preloadMusic

void preloadMusic	(AudioMusicParam preloadParam)

预加载背景音乐

每个音乐都需要您指定具体的 ID，您可以通过该 ID 对音乐的开始、停止、音量等进行设置。

参数	描述
preloadParam	预加载音乐参数。

注意

1. 预先加载最多同时支持2个不同 ID 的预加载，且预加载时长不超过10分钟，使用完需 stopPlayMusic，否则内存不释放。
2. 若该ID对应的音乐正在播放中，预加载会失败，需先调用 stopPlayMusic。
3. 当 musicParam 和传入 startPlayMusic 的 musicParam 完全相同时，预加载有效。

getMusicTrackCount

getMusicTrackCount

long getMusicTrackCount	(int id)

获取背景音乐的音轨数量

参数	描述
id	音乐 ID。

setMusicTrack

setMusicTrack

void setMusicTrack	(int id
	int trackIndex)

指定背景音乐的播放音轨

参数	描述
id	音乐 ID。
index	默认播放第一个音轨。取值范围[0, 音轨总数)。

注意

音轨总数量可通过 getMusicTrackCount 接口获取。

TXVoiceReverbType

TXVoiceReverbType

混响特效

混响特效可以作用于人声之上，通过声学算法对声音进行叠加处理，模拟出各种不同环境下的临场感受，目前支持如下几种混响效果：
0：关闭；1：KTV；2：小房间；3：大会堂；4：低沉；5：洪亮；6：金属声；7：磁性；8：空灵；9：录音棚；10：悠扬 11：录音棚2。

枚举	取值	描述
TXLiveVoiceReverbType_0	0	关闭特效
TXLiveVoiceReverbType_1	1	KTV
TXLiveVoiceReverbType_2	2	小房间
TXLiveVoiceReverbType_3	3	大会堂
TXLiveVoiceReverbType_4	4	低沉
TXLiveVoiceReverbType_5	5	洪亮
TXLiveVoiceReverbType_6	6	金属声
TXLiveVoiceReverbType_7	7	磁性
TXLiveVoiceReverbType_8	8	空灵
TXLiveVoiceReverbType_9	9	录音棚
TXLiveVoiceReverbType_10	10	悠扬
TXLiveVoiceReverbType_11	11	录音棚2

TXVoiceChangeType

TXVoiceChangeType

变声特效

变声特效可以作用于人声之上，通过声学算法对人声进行二次处理，以获得与原始声音所不同的音色，目前支持如下几种变声特效：
0：关闭；1：熊孩子；2：萝莉；3：大叔；4：重金属；5：感冒；6：外语腔；7：困兽；8：肥宅；9：强电流；10：重机械；11：空灵。

枚举	取值	描述
TXVoiceChangerType_0	0	关闭
TXVoiceChangerType_1	1	熊孩子
TXVoiceChangerType_2	2	萝莉
TXVoiceChangerType_3	3	大叔
TXVoiceChangerType_4	4	重金属
TXVoiceChangerType_5	5	感冒
TXVoiceChangerType_6	6	外语腔
TXVoiceChangerType_7	7	困兽
TXVoiceChangerType_8	8	肥宅
TXVoiceChangerType_9	9	强电流
TXVoiceChangerType_10	10	重机械
TXVoiceChangerType_11	11	空灵

TXAudioMusicParam

TXAudioMusicParam

背景音乐的播放控制信息

该信息用于在接口 startPlayMusic 中指定背景音乐的相关信息，包括播放 ID、文件路径和循环次数等：
1. 如果要多次播放同一首背景音乐，请不要每次播放都分配一个新的 ID，我们推荐使用相同的 ID。
2. 若您希望同时播放多首不同的音乐，请为不同的音乐分配不同的 ID 进行播放。
3. 如果使用同一个 ID 播放不同音乐，SDK 会先停止播放旧的音乐，再播放新的音乐。

枚举类型	描述
endTimeMS	【字段含义】音乐结束播放时间点，单位毫秒，0表示播放至文件结尾。

id	【字段含义】音乐 ID。 【特殊说明】SDK 允许播放多路音乐，因此需要使用 ID 进行标记，用于控制音乐的开始、停止、音量等。
isShortFile	【字段含义】播放的是否为短音乐文件。 【推荐取值】true：需要重复播放的短音乐文件；false：正常的音乐文件。默认值：false。
loopCount	【字段含义】音乐循环播放的次数。 【推荐取值】取值范围为0 - 任意正整数，默认值：0。0 表示播放音乐一次；1 表示播放音乐两次；以此类推。
path	【字段含义】音效文件的完整路径或 URL 地址。支持的音频格式包括 MP3、AAC、M4A、WAV。
publish	【字段含义】是否将音乐传到远端。 【推荐取值】true：音乐在本地播放的同时，远端用户也能听到该音乐；false：主播只能在本地听到该音乐，远端观众听不到。默认值：false。
startTimeMS	【字段含义】音乐开始播放时间点，单位：毫秒。

ITXDeviceManager

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: 音视频设备管理模块

Function: 用于管理摄像头、麦克风和扬声器等音视频相关的硬件设备

ITXDeviceManager

ITXDeviceManager

函数列表	描述
isFrontCamera	判断当前是否为前置摄像头（仅适用于移动端）
switchCamera	切换前置或后置摄像头（仅适用于移动端）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数（仅适用于移动端）
setCameraZoomRatio	设置摄像头的缩放倍数（仅适用于移动端）
isAutoFocusEnabled	查询是否支持自动识别人脸位置（仅适用于移动端）
enableCameraAutoFocus	开启自动对焦功能（仅适用于移动端）
setCameraFocusPosition	设置摄像头的对焦位置（仅适用于移动端）
enableCameraTorch	开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）
setAudioRoute	设置音频路由（仅适用于移动端）
getDevicesList	获取设备列表（仅适用于桌面端）
setCurrentDevice	设置当前要使用的设备（仅适用于桌面端）
getCurrentDevice	获取当前正在使用的设备（仅适用于桌面端）
setCurrentDeviceVolume	设置当前设备的音量（仅适用于桌面端）
getCurrentDeviceVolume	获取当前设备的音量（仅适用于桌面端）

setCurrentDeviceMute	设置当前设备的静音状态（仅适用于桌面端）
getCurrentDeviceMute	获取当前设备的静音状态（仅适用于桌面端）
enableFollowingDefaultAudioDevice	设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）
startCameraDeviceTest	开始摄像头测试（仅适用于桌面端）
stopCameraDeviceTest	结束摄像头测试（仅适用于桌面端）
startMicDeviceTest	开始麦克风测试（仅适用于桌面端）
startMicDeviceTest	开始麦克风测试（仅适用于桌面端）
stopMicDeviceTest	结束麦克风测试（仅适用于桌面端）
startSpeakerDeviceTest	开始扬声器测试（仅适用于桌面端）
stopSpeakerDeviceTest	结束扬声器测试（仅适用于桌面端）
startCameraDeviceTest	开始摄像头测试（仅适用于桌面端）
setApplicationPlayVolume	设置 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）
getApplicationPlayVolume	获取 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）
setApplicationMuteState	设置 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）
getApplicationMuteState	获取 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）
setCameraCapturerParam	设置摄像头采集偏好
setDeviceObserver	设置 onDeviceChanged 事件回调
setSystemVolumeType	设置系统音量类型（仅适用于移动端）

结构体类型

函数列表	描述
TXCameraCaptureParam	摄像头采集参数

ITXDeviceInfo	音视频设备的相关信息（仅适用于桌面平台）
ITXDeviceCollection	设备信息列表（仅适用于桌面平台）

枚举类型

枚举类型	描述
TXSystemVolumeType	系统音量类型（仅适用于移动设备）
TXAudioRoute	音频路由（即声音的播放模式）
TXMediaDeviceType	设备类型（仅适用于桌面平台）
TXMediaDeviceState	设备操作
TXCameraCaptureMode	摄像头采集偏好

isFrontCamera

isFrontCamera

判断当前是否为前置摄像头（仅适用于移动端）

switchCamera

switchCamera

int switchCamera	(bool frontCamera)

切换前置或后置摄像头（仅适用于移动端）

getCameraZoomMaxRatio

getCameraZoomMaxRatio

获取摄像头的最大缩放倍数（仅适用于移动端）

版权所有：腾讯云计算（北京）有限责任公司

setCameraZoomRatio

setCameraZoomRatio

int setCameraZoomRatio	(float zoomRatio)

设置摄像头的缩放倍数（仅适用于移动端）

参数	描述
zoomRatio	取值范围1 - 5，取值为1表示最远视角（正常镜头），取值为5表示最近视角（放大镜头）。最大值推荐为5，若超过5，视频数据会变得模糊不清。

isAutoFocusEnabled

isAutoFocusEnabled

查询是否支持自动识别人脸位置（仅适用于移动端）

enableCameraAutoFocus

enableCameraAutoFocus

int enableCameraAutoFocus	(bool enabled)

开启自动对焦功能（仅适用于移动端）

开启后，SDK 会自动检测画面中的人脸位置，并将摄像头的焦点始终对焦在人脸位置上。

setCameraFocusPosition

setCameraFocusPosition

int setCameraFocusPosition	(float x
	float y)

设置摄像头的对焦位置（仅适用于移动端）

您可以通过该接口实现如下交互：

1. 在本地摄像头的预览画面上，允许用户单击操作。
2. 在用户的单击位置显示一个矩形方框，以示摄像头会在此处对焦。
3. 随后将用户点击位置的坐标通过本接口传递给 SDK，之后 SDK 会操控摄像头按照用户期望的位置进行对焦。

参数	描述
position	对焦位置，请传入期望对焦点的坐标值

注意

使用该接口的前提是先通过 enableCameraAutoFocus 关闭自动对焦功能。

返回值说明：

0：操作成功；负数：操作失败。

enableCameraTorch

enableCameraTorch

int enableCameraTorch	(bool enabled)

开启/关闭闪光灯，也就是手电筒模式（仅适用于移动端）

setAudioRoute

setAudioRoute

int setAudioRoute	(TXAudioRoute route)

设置音频路由（仅适用于移动端）

手机有两个音频播放设备：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

getDevicesList

getDevicesList

ITXDeviceCollection* getDevicesList	(TXMediaDeviceType type)

获取设备列表（仅适用于桌面端）

参数	描述
type	设备类型，指定需要获取哪种设备的列表。详见 TXMediaDeviceType 定义。

注意

使用完毕后请调用 release 方法释放资源，这样可以让 SDK 维护 ITXDeviceCollection 对象的生命周期。

不要使用 delete 释放返回的 Collection 对象，delete ITXDeviceCollection* 指针会导致异常崩溃。

type 只支持 TXMediaDeviceTypeMic、TXMediaDeviceTypeSpeaker、TXMediaDeviceTypeCamera。

此接口只支持 Mac 和 Windows 平台。

setCurrentDevice

setCurrentDevice

int setCurrentDevice	(TXMediaDeviceType type
	const char* deviceId)

设置当前要使用的设备（仅适用于桌面端）

参数	描述
deviceId	设备ID，您可以通过接口 getDevicesList 获得设备 ID。
type	设备类型，详见 TXMediaDeviceType 定义。

返回值说明：

0：操作成功；负数：操作失败。

getCurrentDevice

getCurrentDevice

ITXDeviceInfo* getCurrentDevice	(TXMediaDeviceType type)

获取当前正在使用的设备（仅适用于桌面端）

setCurrentDeviceVolume

setCurrentDeviceVolume

int setCurrentDeviceVolume	(TXMediaDeviceType type
	uint32_t volume)

设置当前设备的音量（仅适用于桌面端）

这里的音量指的是麦克风的采集音量或者扬声器的播放音量，摄像头是不支持设置音量的。

参数	描述
volume	音量大小，取值范围为0 - 100，默认值：100。

getCurrentDeviceVolume

getCurrentDeviceVolume

uint32_t getCurrentDeviceVolume	(TXMediaDeviceType type)

获取当前设备的音量（仅适用于桌面端）

这里的音量指的是麦克风的采集音量或者扬声器的播放音量，摄像头是不支持获取音量的。

setCurrentDeviceMute

setCurrentDeviceMute

int setCurrentDeviceMute	(TXMediaDeviceType type
	bool mute)

设置当前设备的静音状态（仅适用于桌面端）

这里的音量指的是麦克风和扬声器，摄像头是不支持静音操作的。

getCurrentDeviceMute

getCurrentDeviceMute

bool getCurrentDeviceMute	(TXMediaDeviceType type)

获取当前设备的静音状态（仅适用于桌面端）

这里的音量指的是麦克风和扬声器，摄像头是不支持静音操作的。

enableFollowingDefaultAudioDevice

enableFollowingDefaultAudioDevice

int enableFollowingDefaultAudioDevice	(TXMediaDeviceType type
	bool enable)

设置 SDK 使用的音频设备根据跟随系统默认设备（仅适用于桌面端）

仅支持设置麦克风和扬声器类型，摄像头暂不支持跟随系统默认设备

参数	描述
enable	是否跟随系统默认的音频设备。 true：跟随。当系统默认音频设备发生改变或者有新音频设备插入时，SDK 立即切换音频设备。 false：不跟随。当系统默认音频设备发生改变或者有新音频设备插入时，SDK 不会切换音频设备。
type	设备类型，详见 TXMediaDeviceType 定义。

startCameraDeviceTest

startCameraDeviceTest

int startCameraDeviceTest	(void* view)

开始摄像头测试（仅适用于桌面端）

注意

在测试过程中可以使用 setCurrentDevice 接口切换摄像头。

stopCameraDeviceTest

stopCameraDeviceTest

结束摄像头测试（仅适用于桌面端）

startMicDeviceTest

startMicDeviceTest

int startMicDeviceTest	(uint32_t interval)

开始麦克风测试（仅适用于桌面端）

该接口可以测试麦克风是否能正常工作，测试到的麦克风采集音量的大小，会以回调的形式通知给您，其中 volume 的取值范围为0 - 100。

参数	描述
interval	麦克风音量的回调间隔。

注意

该接口调用后默认会回播麦克风录制到的声音到扬声器中。

startMicDeviceTest

startMicDeviceTest

int startMicDeviceTest	(uint32_t interval
	bool playback)

开始麦克风测试（仅适用于桌面端）

该接口可以测试麦克风是否能正常工作，测试到的麦克风采集音量的大小，会以回调的形式通知给您，其中 volume 的取值范围为0 - 100。

参数	描述
interval	麦克风音量的回调间隔。
playback	是否开启回播麦克风声音，开启后用户测试麦克风时会听到自己的声音。

stopMicDeviceTest

stopMicDeviceTest

结束麦克风测试（仅适用于桌面端）

startSpeakerDeviceTest

startSpeakerDeviceTest

int startSpeakerDeviceTest	(const char* filePath)

开始扬声器测试（仅适用于桌面端）

该接口通过播放指定的音频文件，用于测试播放设备是否能正常工作。如果用户在测试时能听到声音，说明播放设备能正常工作。

参数	描述
filePath	声音文件的路径

stopSpeakerDeviceTest

stopSpeakerDeviceTest

结束扬声器测试（仅适用于桌面端）

startCameraDeviceTest

startCameraDeviceTest

int startCameraDeviceTest	(ITRTCVideoRenderCallback* callback)

开始摄像头测试（仅适用于桌面端）

该接口支持自定义渲染，即您可以通过接 ITRTCVideoRenderCallback 回调接口接管摄像头的渲染画面。

setApplicationPlayVolume

setApplicationPlayVolume

int setApplicationPlayVolume	(int volume)

设置 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）

getApplicationPlayVolume

getApplicationPlayVolume

获取 Windows 系统音量合成器中当前进程的音量（仅适用于 Windows 系统）

setApplicationMuteState

setApplicationMuteState

int setApplicationMuteState	(bool bMute)

设置 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）

getApplicationMuteState

getApplicationMuteState

获取 Windows 系统音量合成器中当前进程的静音状态（仅适用于 Windows 系统）

setCameraCapturerParam

setCameraCapturerParam

void setCameraCapturerParam	(const TXCameraCaptureParam& params)

设置摄像头采集偏好

setDeviceObserver

setDeviceObserver

void setDeviceObserver	(ITXDeviceObserver* observer)

设置 onDeviceChanged 事件回调

版权所有：腾讯云计算（北京）有限责任公司

setSystemVolumeType

setSystemVolumeType

int setSystemVolumeType	(TXSystemVolumeType type)

设置系统音量类型（仅适用于移动端）

@deprecated v9.5 版本开始不推荐使用，建议使用 `TRTCCloud` 中的 startLocalAudio(quality) 接口替代之，通过 quality 参数来决策音质。

TXSystemVolumeType(Deprecated)

TXSystemVolumeType(Deprecated)

系统音量类型（仅适用于移动设备）

@deprecated v9.5 版本开始不推荐使用。

现代智能手机中一般都具备两套系统音量类型，即“通话音量”和“媒体音量”。
通话音量：手机专门为接打电话所设计的音量类型，自带回声抵消（AEC）功能，并且支持通过蓝牙耳机上的麦克风进行拾音，缺点是音质比较一般。
当您通过手机侧面的音量按键下调手机音量时，如果无法将其调至零（也就是无法彻底静音），说明您的手机当前处于通话音量。
媒体音量：手机专门为音乐场景所设计的音量类型，无法使用系统的 AEC 功能，并且不支持通过蓝牙耳机的麦克风进行拾音，但具备更好的音乐播放效果。
当您通过手机侧面的音量按键下调手机音量时，如果能够将手机音量调至彻底静音，说明您的手机当前处于媒体音量。

SDK 目前提供了三种系统音量类型的控制模式：自动切换模式、全程通话音量模式、全程媒体音量模式。

枚举	取值	描述
TXSystemVolumeTypeAuto	0	自动切换模式
TXSystemVolumeTypeMedia	1	全程媒体音量
TXSystemVolumeTypeVOIP	2	全程通话音量

TXAudioRoute

版权所有：腾讯云计算（北京）有限责任公司

TXAudioRoute

音频路由（即声音的播放模式）

音频路由，即声音是从手机的扬声器还是从听筒中播放出来，因此该接口仅适用于手机等移动端设备。
手机有两个扬声器：一个是位于手机顶部的听筒，一个是位于手机底部的立体声扬声器。
设置音频路由为听筒时，声音比较小，只有将耳朵凑近才能听清楚，隐私性较好，适合用于接听电话。
设置音频路由为扬声器时，声音比较大，不用将手机贴脸也能听清，因此可以实现“免提”的功能。

枚举	取值	描述
TXAudioRouteSpeakerphone	0	Speakerphone：使用扬声器播放（即“免提”），扬声器位于手机底部，声音偏大，适合外放音乐。
TXAudioRouteEarpiece	1	Earpiece：使用听筒播放，听筒位于手机顶部，声音偏小，适合需要保护隐私的通话场景。

TXMediaDeviceType

TXMediaDeviceType

设备类型（仅适用于桌面平台）

该枚举值用于定义三种类型的音视频设备，即摄像头、麦克风和扬声器，以便让一套设备管理接口可以操控三种不同类型的设备。

枚举	取值	描述
TXMediaDeviceTypeUnknown	-1	未定义的设备类型
TXMediaDeviceTypeMic	0	麦克风类型设备
TXMediaDeviceTypeSpeaker	1	扬声器类型设备
TXMediaDeviceTypeCamera	2	摄像头类型设备

TXMediaDeviceState

TXMediaDeviceState

设备操作

该枚举值用于本地设备的状态变化通知 onDeviceChanged。

枚举	取值	描述
TXMediaDeviceStateAdd	0	设备已被插入
TXMediaDeviceStateRemove	1	设备已被移除
TXMediaDeviceStateActive	2	设备已启用
TXMediaDefaultDeviceChanged	3	系统默认设备变更

TXCameraCaptureMode

TXCameraCaptureMode

摄像头采集偏好

该枚举类型用于摄像头采集参数设置。

枚举	取值	描述
TXCameraResolutionStrategyAuto	0	自动调整采集参数。 SDK 根据实际的采集设备性能及网络情况，选择合适的摄像头输出参数，在设备性能及视频预览质量之间，维持平衡。
TXCameraResolutionStrategyPerformance	1	优先保证设备性能。 SDK 根据用户设置编码器的分辨率和帧率，选择最接近的摄像头输出参数，从而保证设备性能。
TXCameraResolutionStrategyHighQuality	2	优先保证视频预览质量。 SDK选择较高的摄像头输出参数，从而提高预览视频的质量。在这种情况下，会消耗更多的 CPU 及内存做视频前处理。
TXCameraCaptureManual	3	允许用户设置本地摄像头采集的视频宽高。

TXCameraCaptureParam

TXCameraCaptureParam

摄像头采集参数

该设置能决定本地预览图像画质。

枚举类型	描述
height	**字段含义：** 采集图像宽度
mode	**字段含义：** 摄像头采集偏好，请参见 TXCameraCaptureMode
width	**字段含义：** 采集图像长度

TXMediaDeviceInfo

TXMediaDeviceInfo

音视频设备的相关信息（仅适用于桌面平台）

该结构体用于描述一个音视频设备的关键信息，比如设备 ID、设备名称等等，以便用户能够在用户界面上选择自己期望使用的音视频设备。

枚举类型	描述
getDeviceName()	设备名称 （UTF-8）
getDevicePID()	设备 ID （UTF-8）

ITXDeviceCollection

ITXDeviceCollection

设备信息列表（仅适用于桌面平台）

此结构体的作用相当于 std::vector<ITXDeviceInfo>，用于解决不同版本的 STL 容器的二进制兼容问题。

枚举类型	描述
getCount()	设备数量
index)	设备信息（JSON 格式） **注意** 示例：{"SupportedResolution":[{"width":640,"height":480},{"width":320,"height":240}]} param index 设备索引，值为 [0,getCount)，return 返回 JSON 格式的设备信息
release()	释放设备列表，请不要使用 delete 释放资源 !!!

版权所有：腾讯云计算（北京）有限责任公司

类型定义

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 关键类型定义

Function: 分辨率、质量等级等枚举和常量值的定义

类型定义

结构体类型

函数列表	描述
TRTCParams	进房参数
TRTCVideoEncParam	视频编码参数
TRTCNetworkQosParam	网络流控（Qos）参数集
TRTCRenderParams	视频画面的渲染参数
TRTCQualityInfo	网络质量
TRTCVolumeInfo	音量大小
TRTCSpeedTestParams	测速参数
TRTCSpeedTestResult	网络测速结果
TRTCTexture	视频纹理数据
TRTCVideoFrame	视频帧信息
TRTCAudioFrame	音频帧数据
TRTCMixUser	云端混流中各路画面的描述信息
TRTCTranscodingConfig	云端混流的排版布局和转码参数
TRTCPublishCDNParam	向非腾讯云 CDN 上发布音视频流时需设置的转推参数

TRTCAudioRecordingParams	本地音频文件的录制参数
TRTCLocalRecordingParams	本地媒体文件的录制参数
TRTCAudioEffectParam	音效参数（已废弃）
TRTCSwitchRoomConfig	房间切换参数
TRTCAudioFrameCallbackFormat	音频自定义回调的格式参数
TRTCImageBuffer	TRTC 屏幕分享图标信息以及 mute image 垫片
TRTCScreenCaptureSourceInfo	屏幕分享的目标信息（仅适用于桌面系统）
TRTCScreenCaptureProperty	屏幕分享的进阶控制参数
TRTCAudioParallelParams	远端音频流智能并发播放策略的参数
TRTCUser	媒体流发布相关配置的用户信息
TRTCPublishCdnUrl	向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置
TRTCPublishTarget	目标推流配置
TRTCVideoLayout	转码视频布局
TRTCWatermark	水印布局
TRTCStreamEncoderParam	媒体流编码输出参数
TRTCStreamMixingConfig	媒体流转码配置参数
TRTCPayloadPrivateEncryptionConfig	媒体流私有加密配置
TRTCAudioVolumeEvaluateParams	音量评估等相关参数设置

枚举类型

枚举类型	描述
TRTCVideoResolution	视频分辨率
TRTCVideoResolutionMode	视频宽高比模式
TRTCVideoStreamType	视频流类型
TRTCVideoFillMode	视频画面填充模式

版权所有：腾讯云计算（北京）有限责任公司

TRTCVideoRotation	视频画面旋转方向
TRTCBeautyStyle	美颜（磨皮）算法
TRTCVideoPixelFormat	视频像素格式
TRTCVideoBufferType	视频数据传递方式
TRTCVideoMirrorType	视频的镜像类型
TRTCSnapshotSourceType	本地视频截图的数据源
TRTCAppScene	应用场景
TRTCRoleType	角色
TRTCQosControlMode	流控模式（已废弃）
TRTCVideoQosPreference	画质偏好
TRTCQuality	网络质量
TRTCAVStatusType	音视频状态类型
TRTCAVStatusChangeReason	音视频状态变化原因类型
TRTCAudioQuality	声音音质
TRTCAudioFrameFormat	音频帧的内容格式
TRTCAudioFrameOperationMode	音频回调数据读写模式
TRTCLogLevel	Log 级别
TRTCScreenCaptureSourceType	屏幕分享的目标类型（仅适用于桌面端）
TRTCTranscodingConfigMode	云端混流的排版模式
TRTCLocalRecordType	媒体录制类型
TRTCMixInputType	混流输入类型
TRTCWaterMarkSrcType	水印图片的源类型
TRTCAudioRecordingContent	音频录制内容类型
TRTCPublishMode	媒体流发布模式
TRTCEncryptionAlgorithm	加密算法

版权所有：腾讯云计算（北京）有限责任公司

TRTCSpeedTestScene	测速场景
TRTCGravitySensorAdaptiveMode	设置重力感应的适配模式（仅适用于移动端）

TRTCVideoResolution

TRTCVideoResolution

视频分辨率

此处仅定义横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTCVideoResolution_120_120	1	宽高比 1:1；分辨率 120x120；建议码率（VideoCall）80kbps; 建议码率（LIVE）120kbps。
TRTCVideoResolution_160_160	3	宽高比 1:1 分辨率 160x160；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTCVideoResolution_270_270	5	宽高比 1:1；分辨率 270x270；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_480_480	7	宽高比 1:1；分辨率 480x480；建议码率（VideoCall）350kbps; 建议码率（LIVE）500kbps。
TRTCVideoResolution_160_120	50	宽高比4:3；分辨率 160x120；建议码率（VideoCall）100kbps; 建议码率（LIVE）150kbps。
TRTCVideoResolution_240_180	52	宽高比 4:3；分辨率 240x180；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。
TRTCVideoResolution_280_210	54	宽高比 4:3；分辨率 280x210；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_320_240	56	宽高比 4:3；分辨率 320x240；建议码率（VideoCall）250kbps; 建议码率（LIVE）375kbps。
TRTCVideoResolution_400_300	58	宽高比 4:3；分辨率 400x300；建议码率（VideoCall）300kbps; 建议码率（LIVE）450kbps。
TRTCVideoResolution_480_360	60	宽高比 4:3；分辨率 480x360；建议码率（VideoCall）400kbps; 建议码率（LIVE）600kbps。
TRTCVideoResolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率

版权所有：腾讯云计算（北京）有限责任公司

		（VideoCall）600kbps; 建议码率（LIVE）900kbps。
TRTCVideoResolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps; 建议码率（LIVE）1500kbps。
TRTCVideoResolution_160_90	100	宽高比 16:9；分辨率 160x90；建议码率（VideoCall）150kbps; 建议码率（LIVE）250kbps。
TRTCVideoResolution_256_144	102	宽高比 16:9；分辨率 256x144；建议码率（VideoCall）200kbps; 建议码率（LIVE）300kbps。
TRTCVideoResolution_320_180	104	宽高比 16:9；分辨率 320x180；建议码率（VideoCall）250kbps; 建议码率（LIVE）400kbps。
TRTCVideoResolution_480_270	106	宽高比 16:9；分辨率 480x270；建议码率（VideoCall）350kbps; 建议码率（LIVE）550kbps。
TRTCVideoResolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps; 建议码率（LIVE）900kbps。
TRTCVideoResolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps; 建议码率（LIVE）1300kbps。
TRTCVideoResolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps; 建议码率（LIVE）1800kbps。
TRTCVideoResolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps; 建议码率（LIVE）3000kbps。

TRTCVideoResolutionMode

TRTCVideoResolutionMode

视频宽高比模式

TRTCVideoResolution 中仅定义了横屏分辨率（如 640 × 360），如需使用竖屏分辨率（如360 × 640），需要同时指定 TRTCVideoResolutionMode 为 Portrait。

枚举	取值	描述
TRTCVideoResolutionModeLandscape	0	横屏分辨率，例如：TRTCVideoResolution_640_360 + TRTCVideoResolutionModeLandscape = 640 ×

		360。
TRTCVideoResolutionModePortrait	1	竖屏分辨率，例如：TRTCVideoResolution_640_360 + TRTCVideoResolutionModePortrait = 360 × 640。

TRTCVideoStreamType

TRTCVideoStreamType

视频流类型

TRTC 内部有三种不同的视频流，分别是：
高清大画面：一般用来传输摄像头的视频数据。
低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

注意

不支持单独开启低清小画面，小画面必须依附于大画面而存在，SDK 会自动设定低清小画面的分辨率和码率。

枚举	取值	描述
TRTCVideoStreamTypeBig	0	高清大画面，一般用来传输摄像头的视频数据。
TRTCVideoStreamTypeSmall	1	低清小画面：小画面和大画面的内容相互，但是分辨率和码率都比大画面低，因此清晰度也更低。
TRTCVideoStreamTypeSub	2	辅流画面：一般用于屏幕分享，同一时间在同一个房间中只允许一个用户发布辅流视频，其他用户必须要等该用户关闭之后才能发布自己的辅流。

TRTCVideoFillMode

TRTCVideoFillMode

视频画面填充模式

如果视频显示区域的宽高比不等于视频内容的宽高比时，需要指定画面的填充模式。

枚举	取值	描述
TRTCVideoFillMode_Fill	0	填充模式：即将画面内容居中等比缩放以充满整个显示区域，超出显示区域的部分将会被裁剪掉，此模式下画面可能不完整。

TRTCVideoFillMode_Fit	1	适应模式：即按画面长边进行缩放以适应显示区域，短边部分会被填充为黑色，此模式下图像完整但可能留有黑边。

TRTCVideoRotation

TRTCVideoRotation

视频画面旋转方向

TRTC 提供了对本地和远程画面的旋转角度设置 API，下列的旋转角度都是指顺时针方向的。

枚举	取值	描述
TRTCVideoRotation0	0	不旋转。
TRTCVideoRotation90	1	顺时针旋转90度。
TRTCVideoRotation180	2	顺时针旋转180度。
TRTCVideoRotation270	3	顺时针旋转270度。

TRTCBeautyStyle

TRTCBeautyStyle

美颜（磨皮）算法

TRTC 内置多种不同的磨皮算法，您可以选择最适合您产品定位的方案。

枚举	取值	描述
TRTCBeautyStyleSmooth	0	光滑，算法比较激进，磨皮效果比较明显，适用于秀场直播。
TRTCBeautyStyleNature	1	自然，算法更多地保留了面部细节，磨皮效果更加自然，适用于绝大多数直播场景。

TRTCVideoPixelFormat

TRTCVideoPixelFormat

视频像素格式

TRTC 提供针对视频的自定义采集和自定义渲染功能：

在自定义采集功能中，您可以用下列枚举值描述您采集的视频像素格式。
在自定义渲染功能中，您可以指定您期望 SDK 回调出的视频像素格式。

枚举	取值	描述
TRTCVideoPixelFormat_Unknown	0	未定义的格式。
TRTCVideoPixelFormat_I420	1	YUV420P(I420) 格式。
TRTCVideoPixelFormat_Texture_2D	2	OpenGL 2D 纹理格式。
TRTCVideoPixelFormat_BGRA32	3	BGRA 格式
TRTCVideoPixelFormat_NV21	4	NV21 格式。
TRTCVideoPixelFormat_RGBA32	5	RGBA 格式。

TRTCVideoBufferType

TRTCVideoBufferType

视频数据传递方式

在自定义采集和自定义渲染功能，您需要用到下列枚举值来指定您希望以什么方式传递视频数据：
方案一：使用内存 Buffer 传递视频数据，该方案在 iOS 效率尚可，但在 Android 系统上效率较差，Windows 暂时仅支持内存 Buffer 的传递方式。
方案二：使用 Texture 纹理传递视频数据，该方案在 iOS 和 Android 系统下均有较高的效率，Windows 暂不支持，需要您有一定的 OpenGL 编程基础。

枚举	取值	描述
TRTCVideoBufferType_Unknown	0	未定义的传递方式。
TRTCVideoBufferType_Buffer	1	使用内存 Buffer 传递视频数据，iOS：PixelBuffer；Android：用于 JNI 层的 Direct Buffer；Win：内存数据块。
TRTCVideoBufferType_Texture	3	使用 OpengGL 纹理传递视频数据。
TRTCVideoBufferType_TextureD3D11	4	使用 D3D11 纹理传递视频数据。

TRTCVideoMirrorType

TRTCVideoMirrorType

版权所有：腾讯云计算（北京）有限责任公司

视频的镜像类型

视频的镜像是指对视频内容进行左右翻转，尤其是对本地的摄像头预览视频，开启镜像后能给主播带来熟悉的“照镜子”体验。

枚举	取值	描述
TRTCVideoMirrorType_Auto	0	自动模式：如果正使用前置摄像头则开启镜像，如果是后置摄像头则不开启镜像（仅适用于移动设备）。
TRTCVideoMirrorType_Enable	1	强制开启镜像，不论当前使用的是前置摄像头还是后置摄像头。
TRTCVideoMirrorType_Disable	2	强制关闭镜像，不论当前使用的是前置摄像头还是后置摄像头。

TRTCSnapshotSourceType

TRTCSnapshotSourceType

本地视频截图的数据源

SDK 支持从如下两种数据源中截取图片并保存成本地文件：
视频流：从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
渲染层：从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
采集层：从采集控件中截取采集到的视频内容，可以截取采集出来的高清截图。

枚举	取值	描述
TRTCSnapshotSourceTypeStream	0	从视频流中截取原生的视频内容，截取的内容不受渲染控件的显示控制。
TRTCSnapshotSourceTypeView	1	从渲染控件中截取显示的视频内容，可以做到用户所见即所得的效果，但如果显示区域过小，截取出的图片也会很小。
TRTCSnapshotSourceTypeCapture	2	从采集控件中截取采集到的视频内容，可以截取采集出来的高清截图。

TRTCAppScene

TRTCAppScene

应用场景

TRTC 针对常见的音视频应用场景都进行了定向优化，以满足各种垂直场景下的差异化要求，主要场景可以分为如下两类：

直播（LIVE）场景：包括 LIVE 和 VoiceChatRoom，前者是音频+视频，后者是纯音频。

直播场景下，用户被分成“主播”和“观众”两种角色，单个房间中同时最多支持10万人在线，适合于观众人数众多的直播场景。

实时（RTC）场景：包括 VideoCall 和 AudioCall，前者是音频+视频，后者是纯音频。

实时场景下，用户没有角色的差异，但单个房间中同时最多支持 300 人在线，适合于小范围实时通信的场景。

枚举	取值	描述
TRTCAppSceneVideoCall	0	视频通话场景，支持720P、1080P高清画质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于[1对1视频通话]、[300人视频会议]、[在线问诊]、[教育小班课]、[远程面试]等业务场景。
TRTCAppSceneLIVE	1	视频互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于300ms；支持十万级别观众同时播放，播放延时低至1000ms。 适用于[低延时互动直播]、[大班课]、[主播PK]、[视频相亲]、[在线互动课堂]、[远程培训]、[超大型会议]等业务场景。 **注意** 此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。
TRTCAppSceneAudioCall	2	语音通话场景，默认采用 SPEECH 音质，单个房间最多支持300人同时在线，最高支持50人同时发言。 适用于 [1对1语音通话]、[300人语音会议]、[语音聊天]、[语音会议]、[在线狼人杀] 等业务场景。
TRTCAppSceneVoiceChatRoom	3	语音互动直播，支持平滑上下麦，切换过程无需等待，主播延时小于 300ms；支持十万级别观众同时播放，播放延时低至 1000ms。 适用于 [语音俱乐部]、[在线K歌房]、[音乐直播间]、[FM电台] 等业务场景。 **注意** 此场景下，您必须通过 TRTCParams 中的 role 字段指定当前用户的角色。

TRTCRoleType

TRTCRoleType

角色

仅适用于直播类场景（即 TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom），把用户区分成两种不同的身份：

主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。

观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

枚举	取值	描述
TRTCRoleAnchor	20	主播：可以随时发布自己的音视频流，但人数有限制，同一个房间中最多只允许 50 个主播同时发布自己的音视频流。
TRTCRoleAudience	21	观众：只能观看其他用户的音视频流，要发布音视频流，需要先通过 switchRole 切换成主播，同一个房间中最多能容纳10万观众。

TRTCQosControlMode(Deprecated)

TRTCQosControlMode(Deprecated)

流控模式（已废弃）

枚举	取值	描述
TRTCQosControlModeClient	0	本地控制，用于 SDK 开发内部调试，客户请勿使用。
TRTCQosControlModeServer	1	云端控制，默认模式，推荐选择。

TRTCVideoQosPreference

TRTCVideoQosPreference

画质偏好

TRTC 在弱网络环境下有两种调控模式：“优先保证画面清晰”或“优先保证画面流畅”，两种模式均会优先保障声音数据的传输。

枚举	取值	描述
TRTCVideoQosPreferenceSmooth	1	流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。

TRTCVideoQosPreferenceClear	2	清晰优先（默认值）：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。

TRTCQuality

TRTCQuality

网络质量

TRTC 会每隔两秒对当前的网络质量进行评估，评估结果为六个等级：Excellent 表示最好，Down 表示最差。

枚举	取值	描述
TRTCQuality_Unknown	0	未定义。
TRTCQuality_Excellent	1	当前网络非常好。
TRTCQuality_Good	2	当前网络比较好。
TRTCQuality_Poor	3	当前网络一般。
TRTCQuality_Bad	4	当前网络较差。
TRTCQuality_Vbad	5	当前网络很差。
TRTCQuality_Down	6	当前网络不满足 TRTC 的最低要求。

TRTCAVStatusType

TRTCAVStatusType

音视频状态类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态。

枚举	取值	描述
TRTCAVStatusStopped	0	停止播放。
TRTCAVStatusPlaying	1	正在播放。
TRTCAVStatusLoading	2	正在加载。

版权所有：腾讯云计算（北京）有限责任公司

TRTCAVStatusChangeReason

TRTCAVStatusChangeReason

音视频状态变化原因类型

该枚举类型用于音频状态变化回调接口（onRemoteAudioStatusUpdated）与视频状态变化回调接口（onRemoteVideoStatusUpdated），用于指定当前的音频或视频状态变化原因。

枚举	取值	描述
TRTCAVStatusChangeReasonInternal	0	缺省值。
TRTCAVStatusChangeReasonBufferingBegin	1	网络缓冲。
TRTCAVStatusChangeReasonBufferingEnd	2	结束缓冲。
TRTCAVStatusChangeReasonLocalStarted	3	本地启动音频或视频流播放。
TRTCAVStatusChangeReasonLocalStopped	4	本地停止音频或视频流播放。
TRTCAVStatusChangeReasonRemoteStarted	5	远端音频或视频流开始（或继续）。
TRTCAVStatusChangeReasonRemoteStopped	6	远端音频或视频流停止（或中断）。

TRTCAudioQuality

TRTCAudioQuality

声音音质

TRTC 提供了三种精心校调好的模式，用来满足各种垂直场景下对音质的差异化追求：
人声模式（Speech）：适用于以人声沟通为主的应用场景，该模式下音频传输的抗性较强，TRTC 会通过各种人声处理技术保障在弱网络环境下的流畅度最佳。
音乐模式（Music）：适用于对声乐要求很苛刻的场景，该模式下音频传输的数据量很大，TRTC 会通过各项技术确保音乐信号在各频段均能获得高保真的细节还原度。
默认模式（Default）：介于 Speech 和 Music 之间的档位，对音乐的还原度比人声模式要好，但传输数据量比音乐模式要低很多，对各种场景均有不错的适应性。

枚举	取值	描述
TRTCAudioQualitySpeech	1	人声模式：采样率：16k；单声道；编码码率：16kbps；具备几个模式中最强的网络抗性，适合语音通话为主的场景，比如在线会议，语音通话等。

TRTCAudioQualityDefault	2	默认模式：采样率：48k；单声道；编码码率：50kbps；介于 Speech 和 Music 之间的档位，SDK 默认档位，推荐选择。
TRTCAudioQualityMusic	3	音乐模式：采样率：48k；全频带立体声；编码码率：128kbps；适合需要高保真传输音乐的场景，比如在线K歌、音乐直播等。

TRTCAudioFrameFormat

TRTCAudioFrameFormat

音频帧的内容格式

枚举	取值	描述
TRTCAudioFrameFormatNone	0	None
TRTCAudioFrameFormatPCM	Not Defined	PCM 格式的音频数据。

TRTCAudioFrameOperationMode

TRTCAudioFrameOperationMode

音频回调数据读写模式

TRTC 提供了两种音频回调数据的操作模式。

读写模式（ReadWrite）：可以获取并修改回调的音频数据，默认模式。

只读模式（ReadOnly）：仅从回调中获取音频数据。

枚举	取值	描述
TRTCAudioFrameOperationModeReadWrite	0	读写模式：可以获取并修改回调的音频数据。
TRTCAudioFrameOperationModeReadOnly	1	只读模式：仅从回调中获取音频数据。

TRTCLogLevel

TRTCLogLevel

Log 级别

不同的日志等级定义了不同的详实程度和日志数量，推荐一般情况下将日志等级设置为：TRTCLogLevelInfo。

枚举	取值	描述
TRTCLogLevelVerbose	0	输出所有级别的 Log。
TRTCLogLevelDebug	1	输出 DEBUG，INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelInfo	2	输出 INFO，WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelWarn	3	输出WARNING，ERROR 和 FATAL 级别的 Log。
TRTCLogLevelError	4	输出ERROR 和 FATAL 级别的 Log。
TRTCLogLevelFatal	5	仅输出 FATAL 级别的 Log。
TRTCLogLevelNone	6	不输出任何 SDK Log。

TRTCScreenCaptureSourceType

TRTCScreenCaptureSourceType

屏幕分享的目标类型（仅适用于桌面端）

枚举	取值	描述
TRTCScreenCaptureSourceTypeUnknown	-1	未定义。
TRTCScreenCaptureSourceTypeWindow	0	该分享目标是某一个应用的窗口。
TRTCScreenCaptureSourceTypeScreen	1	该分享目标是某一台显示器的屏幕。
TRTCScreenCaptureSourceTypeCustom	2	该分享目标是用户自定义的数据源。

TRTCTranscodingConfigMode

TRTCTranscodingConfigMode

云端混流的排版模式

TRTC 的云端混流服务能够将房间中的多路音视频流混合成一路，因此您需要指定画面的排版方案，我们提供了如下几种排版模式。

枚举	取值	描述
TRTCTranscodingConfigMode_Unknown	0	未定义。
TRTCTranscodingConfigMode_Manual	1	全手动排版模式。 该模式下，您需要指定每一路画面的精确排版位置。该模式的自由度最高，但易用性也最差： 您需要填写 TRTCTranscodingConfig 中的所有参数，包括每一路画面（TRTCMixUser）的位置坐标。 您需要监听 ITRTCCloudCallback 中的 onUserVideoAvailable() 和 onUserAudioAvailable() 事件回调，并根据当前房间中各个麦上用户的音视频状态不断地调整 mixUsers 参数。
TRTCTranscodingConfigMode_Template_PureAudio	2	纯音频模式。 该模式适用于语音通话（AudioCall）和语音聊天室（VoiceChatRoom）等纯音频的应用场景。 您只需要在进入房间后，通过 setMixTranscodingConfig 接口设置一次，之后 SDK 就会自动把房间内所有上麦用户的声音混流到当前用户的直播流上。 您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，只需设置 audioSampleRate、audioBitrate 和 audioChannels 等参数即可。
TRTCTranscodingConfigMode_Template_PresetLayout	3	预排版模式。 最受欢迎的排版模式，因为该模式支持您通过占位符提前对各路画面的位置进行设定，之后 SDK 会自动根据房间中画面的路数动态进行适配调整。 此模式下，您依然需要设置 mixUsers 参数，但可以将 userId 设置为“占位符”，可选的占位符有： "$PLACE_HOLDER_REMOTE$" ：指代远程用户的画面，可以设置多个。 "$PLACE_HOLDER_LOCAL_MAIN$" ：指代本地摄像头画面，只允许设置一个。

		"$PLACE_HOLDER_LOCAL_SUB$"：指代本地屏幕分享画面，只允许设置一个。 此模式下，您不需要监听 ITRTCCloudCallback 中的 onUserVideoAvailable 和 onUserAudioAvailable 回调进行实时调整，只需要在进房成功后调用一次 setMixTranscodingConfig 即可，之后 SDK 会自动将真实的 userId 补位到您设置的占位符上。
TRTCTranscodingConfigMode_Template_ScreenSharing	4	屏幕分享模式。 适用于在线教育场景等以屏幕分享为主的应用场景，仅支持 Windows 和 Mac 两个平台的 SDK。 该模式下，SDK 会先根据您通过 videoWidth 和 videoHeight 参数设置的目标分辨率构建一张画布， 当老师未开启屏幕分享时，SDK 会将老师的摄像头画面等比例拉伸绘制到该画布上； 当老师开启屏幕分享之后，SDK 会将屏幕分享画面绘制到同样的画布上。 此种排版模式的目的是为了确保混流模块的输出分辨率一致，避免课程回放和网页观看的花屏问题（网页播放器不支持可变分辨率）。 同时，连麦学生的声音也会被默认混合到老师的音视频流中。 由于教学模式下的视频内容以屏幕分享为主，因此同时传输摄像头画面和屏幕分享画面是非常浪费带宽的。 推荐的做法是直接将摄像头画面通过 setLocalVideoRenderCallback 接口自定义绘制到当前屏幕上。 在该模式下，您无需设置 TRTCTranscodingConfig 中的 mixUsers 参数，SDK 不会混合学生的画面，以免干扰屏幕分享的效果。 您可以将 TRTCTranscodingConfig 中的 width x height 设为 0px × 0px，SDK 会自动根据用户当前屏幕的宽高比计算出一个合适的分辨率： 如果老师当前屏幕宽度 <= 1920px，SDK 会使用老师当前屏幕的实际分辨

		率。 如果老师当前屏幕宽度 > 1920px，SDK 会根据当前屏幕宽高比，选择 1920x1080(16:9)、1920x1200(16:10)、1920x1440(4:3) 三种分辨率中的一种。

TRTCRecordType

TRTCRecordType

媒体录制类型

该枚举类型用于本地媒体录制接口 startLocalRecording，用于指定是录制音视频文件还是纯音频文件。

枚举	取值	描述
TRTCLocalRecordType_Audio	0	仅录制音频。
TRTCLocalRecordType_Video	1	仅录制视频。
TRTCLocalRecordType_Both	2	同时录制音频和视频。

TRTCMixInputType

TRTCMixInputType

混流输入类型

枚举	取值	描述
TRTCMixInputTypeUndefined	0	默认值，考虑到针对老版本的兼容性，如果您指定了 inputType 为 Undefined，SDK 会根据另一个参数 pureAudio 的数值决定混流输入类型。
TRTCMixInputTypeAudioVideo	1	混入音频和视频。
TRTCMixInputTypePureVideo	2	只混入视频。
TRTCMixInputTypePureAudio	3	只混入音频。
TRTCMixInputTypeWatermark	4	混入水印，此时您无需指定 userId 字段，但需要指定 image 字段，推荐使用 png 格式的图片。

版权所有：腾讯云计算（北京）有限责任公司

TRTCWaterMarkSrcType

TRTCWaterMarkSrcType

水印图片的源类型

枚举	取值	描述
TRTCWaterMarkSrcTypeFile	0	图片文件路径，支持 BMP、GIF、JPEG、PNG、TIFF、Exif、WMF 和 EMF 文件格式。
TRTCWaterMarkSrcTypeBGRA32	1	BGRA32格式内存块
TRTCWaterMarkSrcTypeRGBA32	2	RGBA32格式内存块。

TRTCAudioRecordingContent

TRTCAudioRecordingContent

音频录制内容类型

该枚举类型用于音频录制接口 startAudioRecording，用于指定录制音频的内容。

枚举	取值	描述
TRTCAudioRecordingContentAll	0	录制本地和远端所有音频。
TRTCAudioRecordingContentLocal	1	仅录制本地音频。
TRTCAudioRecordingContentRemote	2	仅录制远端音频。

TRTCPublishMode

TRTCPublishMode

媒体流发布模式

该枚举类型用于媒体流发布接口 startPublishMediaStream TRTC 的媒体流发布服务能够将房间中的多路音视频流混合成一路发布至 CDN 或者回推到房间内，也可以将您当前的这路音视频发布到腾讯或者第三方 CDN 因此您需要指定对应媒体流的发布模式，我们提供了如下几种模式。

枚举	取值	描述
TRTCPublishModeUnknown	0	未定义。

版权所有：腾讯云计算（北京）有限责任公司

TRTCPublishBigStreamToCdn	1	您可以通过设置该参数将您房间内的主路流（TRTCVideoStreamTypeBig）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishSubStreamToCdn	2	您可以通过设置该参数将您房间内的辅路流（TRTCVideoStreamTypeSub）发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishMixStreamToCdn	3	您可以通过设置该参数，配合编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到腾讯或者第三方直播 CDN 服务商（仅支持标准 RTMP 协议）。
TRTCPublishMixStreamToRoom	4	您可以通过设置该参数，配合媒体流编码输出参数(TRTCStreamEncoderParam) 和混流转码参数(TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您指定的房间中。 通过 TRTCPublishTarget 中的 TRTCUser 进行指定回推房间的机器人信息。

TRTCEncryptionAlgorithm

TRTCEncryptionAlgorithm

加密算法

该枚举类型用于媒体流私有加密算法选择。

枚举	取值	描述
TRTCEncryptionAlgorithmAes128Gcm	0	AES GCM 128。
TRTCEncryptionAlgorithmAes256Gcm	1	AES GCM 256。

TRTCSpeedTestScene

TRTCSpeedTestScene

测速场景

该枚举类型用于测速场景选择。

枚举	取值	描述

TRTCSpeedTestScene_DelayTesting	1	延迟测试。
TRTCSpeedTestScene_DelayAndBandwidthTesting	2	延迟与带宽测试。
TRTCSpeedTestScene_OnlineChorusTesting	3	在线合唱测试。

TRTCGravitySensorAdaptiveMode

TRTCGravitySensorAdaptiveMode

设置重力感应的适配模式（仅适用于移动端）

枚举	取值	描述
TRTCGravitySensorAdaptiveMode_Disable	0	关闭重力感应，根据当前采集分辨率与设置的编码分辨率决策，如果两者不一致，则通过旋转90度，保证最大画幅。
TRTCGravitySensorAdaptiveMode_FillByCenterCrop	1	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用居中裁剪模式。
TRTCGravitySensorAdaptiveMode_FitWithBlackBorder	2	开启重力感应，始终保证远端画面图像为正，中间过程需要处理分辨率不一致时，采用叠加黑边模式。

TRTCParams

TRTCParams

进房参数

作为 TRTC SDK 的进房参数，只有该参数填写正确，才能顺利进入 roomId 或者 strRoomId 所指定的音视频房间。
由于历史原因，TRTC 支持数字和字符串两种类型的房间号，分别是 roomId 和 strRoomId。
请注意：不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 “123” 在 TRTC 看来是两个完全不同的房间。

枚举类型	描述
businessInfo	【字段含义】业务数据字段（选填），部分高级特性才需要用到此字段。

	【推荐取值】请不要自行设置该字段。
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
role	【字段含义】直播场景下的角色，仅适用于直播场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom），通话场景下指定该参数是无效的。 【推荐取值】默认值：主播（TRTCRoleAnchor）。
roomId	【字段含义】数字房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为 0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。
sdkAppId	【字段含义】应用标识（必填），腾讯云基于 sdkAppId 完成计费统计。 【推荐取值】在 **实时音视频控制台** 创建应用后可以在账号信息页面中得到该 ID。
strRoomId	【字段含义】字符串房间号，在同一个房间里的用户（userId）可以彼此看到对方并进行音视频通话。 【特别说明】roomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 roomId 需要填写为0。若两者都填，SDK 将优先选用 roomId。 【请您注意】不要混用 roomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 `123` 在 TRTC 看来是两个完全不同的房间。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、`!`、`#`、`$`、`%`、`&`、`(`、`)`、`+`、`-`、`:`、`;`、`<`、`=`、`.`、`>`、`?`、`@`、`[`、`]`、`^`、`_`、`{`、`}`、`|`、`~`、`,`。
streamId	【字段含义】用于指定在腾讯云直播平台上的 streamId（选填），设置之后，您可以在腾讯云直播 CDN 上通过标准拉流方案（FLV 或 HLS）播放该用户的音视频流。 【推荐取值】限制长度为64字节，可以不填写，一种推荐的方案是使用 `sdkappid_roomid_userid_main` 作为 streamid，这种命名方式容易辨认且不会在您的多个应用中发生冲突。 【特殊说明】要使用腾讯云直播 CDN，您需要先在 控制台 中的功能配置页开启“启动自动旁路直播”开关。 【参考文档】**CDN 旁路直播**。

userDefineRecordId	【字段含义】云端录制开关（选填），用于指定是否要在云端将该用户的音视频流录制下来。 【参考文档】云端录制。 【推荐取值】限制长度为64字节，只允许包含大小写英文字母（a-zA-Z）、数字（0-9）及下划线和连词符。 方案一：手动录制方案： 1. 在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `手动录制` 。 3. 设置手动录制后，在一个 TRTC 房间中只有设置了 userDefineRecordId 参数的用户才会在云端录制出视频文件，不指定该参数的用户不会产生录制行为。 4. 云端会以 “userDefineRecordId_起始时间_结束时间” 的格式命名录制下来的文件。 方案二：自动录制方案： 1. 需要在 **控制台**>应用管理>云端录制配置 中开启云端录制。 2. 设置 `录制形式` 为 `自动录制` 。 3. 设置自动录制后，在一个 TRTC 房间中的任何一个有音视频上行的用户，均会在云端录制出视频文件。 4. 文件会以 `userDefineRecordId_起始时间_结束时间` 的格式命名，如果不指定 userDefineRecordId，则文件会以 `streamId_起始时间_结束时间` 命名。
userId	【字段含义】用户标识（必填），当前用户的 userId，相当于用户名，使用 UTF-8 编码。 【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。
userSig	【字段含义】用户签名（必填），当前 userId 对应的验证签名，相当于使用云服务的登录密码。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCVideoEncParam

TRTCVideoEncParam

视频编码参数

该设置决定远端用户看到的画面质量，同时也决定了云端录制出的视频文件的画面质量。

枚举类型	描述
enableAdjustRes	【字段含义】是否允许动态调整分辨率（开启后会对云端录制产生影响）。 【推荐取值】该功能适用于不需要云端录制的场景，开启后 SDK 会根据当前网络情况，智能选择出一个合适的分辨率，避免出现“大分辨率+小码率”的低效编码模式。

	【特别说明】默认值：关闭。如有云端录制的需求，请不要开启此功能，因为如果视频分辨率发生变化后，云端录制出的 MP4 在普通的播放器上无法正常播放。
minVideoBitrate	【字段含义】最低视频码率，SDK 会在网络不佳的情况下主动降低视频码率以保持流畅度，最低会降至 minVideoBitrate 所设定的数值。 【特别说明】 默认值：0，此时最低码率由 SDK 会根据您指定的分辨率，自动计算出合适的数值。 【推荐取值】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求 弱网络下允许卡顿但要保持清晰 的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求 弱网络下允许模糊但要保持流畅 的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
resMode	【字段含义】分辨率模式（横屏分辨率 or 竖屏分辨率）。 【推荐取值】手机平台（iOS、Android）建议选择 Portrait，桌面平台（Windows、Mac）建议选择 Landscape。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如： 640 × 360 + Portrait = 360 × 640。
videoBitrate	【字段含义】目标视频码率，SDK 会按照目标码率进行编码，只有在弱网络环境下才会主动降低视频码率。 【推荐取值】请参见本 TRTCVideoResolution 在各档位注释的最佳码率，也可以在此基础上适当调高。比如：TRTCVideoResolution_1280_720 对应 1200kbps 的目标码率，您也可以设置为 1500kbps 用来获得更好的观感清晰度。 【特别说明】您可以通过同时设置 videoBitrate 和 minVideoBitrate 两个参数，用于约束 SDK 对视频码率的调整范围： 如果您追求“弱网络下允许卡顿但要保持清晰”的效果，可以设置 minVideoBitrate 为 videoBitrate 的 60%。 如果您追求“弱网络下允许模糊但要保持流畅”的效果，可以设置 minVideoBitrate 为一个较低的数值（比如 100kbps）。 如果您将 videoBitrate 和 minVideoBitrate 设置为同一个值，等价于关闭 SDK 对视频码率的自适应调节能力。
videoFps	【字段含义】视频采集帧率。 【推荐取值】15fps或20fps。5fps以下，卡顿感明显。10fps以下，会有轻微卡顿感。20fps以上，会浪费带宽（电影的帧率为24fps）。 【特别说明】部分 Android 手机的前置摄像头并不支持15fps以上的采集帧率，部分主打美颜功能的 Android 手机的前置摄像头的采集帧率可能低于10fps。
videoResolution	【字段含义】视频分辨率。 【特别说明】如需使用竖屏分辨率，请指定 resMode 为 Portrait，例如： 640 × 360 + Portrait = 360 × 640。 【推荐取值】

	手机视频通话：建议选择 360 × 640 及以下分辨率，resMode 选择 Portrait，即竖屏分辨率。 手机在线直播：建议选择 540 × 960，resMode 选择 Portrait，即竖屏分辨率。 桌面平台（Win + Mac）：建议选择 640 × 360 及以上分辨率，resMode 选择 Landscape，即横屏分辨率。

TRTCNetworkQosParam

TRTCNetworkQosParam

网络流控（Qos）参数集

网络流控相关参数，该设置决定 SDK 在弱网络环境下的调控策略（例如：“清晰优先”或“流畅优先”）。

枚举类型	描述
controlMode	【字段含义】流控模式（已废弃）。 【推荐取值】云端控制。 【特别说明】请设置为云端控制模式（TRTCQosControlModeServer）。
preference	【字段含义】清晰优先还是流畅优先。 【推荐取值】清晰优先。 【特别说明】该参数主要影响 TRTC 在较差网络环境下的音视频表现： 流畅优先：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的流畅性，代价就是画面会比较模糊且伴随有较多的马赛克。参见 TRTCVideoQosPreferenceSmooth 清晰优先（默认值）：即当前网络不足以传输既清晰又流畅的画面时，优先保证画面的清晰度，代价就是画面会比较卡顿。参见 TRTCVideoQosPreferenceClear

TRTCRenderParams

TRTCRenderParams

视频画面的渲染参数

您可以通过设置此参数来控制画面的旋转角度、填充模式和左右镜像模式。

枚举类型	描述
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTCVideoFillMode_Fill。
mirrorType	【字段含义】画面镜像模式。 【推荐取值】默认值：TRTCVideoMirrorType_Auto。

rotation	【字段含义】图像的顺时针旋转角度。 【推荐取值】支持 90、180 以及 270 旋转角度，默认值：TRTCVideoRotation0。

TRTCQualityInfo

TRTCQualityInfo

网络质量

表征网络质量的好坏，您可以通过该数值在用户界面上展示每个用户的网络质量。

枚举类型	描述
quality	网络质量
userId	用户 ID

TRTCVolumeInfo

TRTCVolumeInfo

音量大小

表征语音音量的评估值，您可以通过该数值在用户界面上展示每个用户的音量大小。

枚举类型	描述
pitch	本地用户的人声频率（单位：Hz），取值范围[0 - 4000]，对于远端用户，该值始终为0。
spectrumData	音频频谱数据是将音频数据在频率域中的分布，划分为 256 个频率段，使用 spectrumData 记录各个频率段的能量值，每个能量值的取值范围为 [-300, 0]，单位为 dBFS。 **注意** 本地频谱使用编码前的音频数据计算，会受到本地采集音量、BGM等影响；远端频谱使用接收到的音频数据计算，本地调整远端播放音量等操作不会对其产生影响。
spectrumDataLength	spectrumDataLength 记录音频频谱数据的长度，为 256。
userId	说话者的 userId, 如果 userId 为空则代表是当前用户自己。
vad	是否检测到人声，0：非人声 1：人声。
volume	说话者的音量大小, 取值范围[0 - 100]。

版权所有：腾讯云计算（北京）有限责任公司

TRTCSpeedTestParams

TRTCSpeedTestParams

测速参数

您可以在用户进入房间前通过 startSpeedTest 接口测试网速（注意：请不要在通话中调用）。

枚举类型	描述
expectedDownBandwidth	预期的下行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
expectedUpBandwidth	预期的上行带宽（kbps，取值范围： 10 ～ 5000，为 0 时不测试）。 **注意** 当参数 scene 设置为 TRTCSpeedTestScene_OnlineChorusTesting 时，为了获取更准确的 rtt / jitter 等信息，取值范围限制为 10 ～ 1000。
scene	测速场景。
sdkAppId	应用标识，请参见 TRTCParams 中的相关说明。
userId	用户标识，请参见 TRTCParams 中的相关说明。
userSig	用户签名，请参见 TRTCParams 中的相关说明。

TRTCSpeedTestResult

TRTCSpeedTestResult

网络测速结果

您可以在用户进入房间前通过 startSpeedTest 接口进行测速（注意：请不要在通话中调用）。

枚举类型	描述
availableDownBandwidth	下行带宽（kbps，-1：无效值）。
availableUpBandwidth	上行带宽（kbps，-1：无效值）。
downJitter	下行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。

downLostRate	下行丢包率，取值范围是 [0 - 1.0]，例如 0.2 表示每从服务器收取 10 个数据包可能会在中途丢失 2 个。
errMsg	带宽测试错误信息。
ip	服务器 IP 地址。
quality	内部通过评估算法测算出的网络质量，更多信息请参见 TRTCQuality。
rtt	延迟（毫秒），指当前设备到 TRTC 服务器的一次网络往返时间，该值越小越好，正常数值范围是10ms - 100ms。
success	测试是否成功。
upJitter	上行数据包抖动（ms），指用户当前网络环境下数据通信的稳定性，该值越小越好，正常数值范围是0ms - 100ms，-1 代表此次测速没有成功测得有效值，一般情况下 WiFi 网络的 Jitter 会比 4G/5G 环境稍大。
upLostRate	上行丢包率，取值范围是 [0 - 1.0]，例如 0.3 表示每向服务器发送 10 个数据包可能会在中途丢失 3 个。

TRTCTexture

TRTCTexture

视频纹理数据

枚举类型	描述
glContext	【字段含义】纹理所在的 OpenGL 上下文环境，Windows和Android平台有效
glTextureId	【字段含义】视频纹理 ID。
}	【字段含义】D3D11纹理，即ID3D11Texture2D的指针，仅Windows平台有效。

TRTCVideoFrame

TRTCVideoFrame

视频帧信息

TRTCVideoFrame 用来描述一帧视频画面的裸数据，也就是编码前或者解码后的视频画面数据。

枚举类型	描述

bufferType	【字段含义】视频数据结构类型。
data	【字段含义】bufferType 为 TRTCVideoBufferType_Buffer 时的视频数据，承载用于 C++ 层的内存数据块。
height	【字段含义】视频高度。
length	【字段含义】视频数据的长度，单位是字节。对于 i420 而言：length = width * height * 3 / 2；对于 BGRA32 而言：length = width * height * 4。
rotation	【字段含义】视频像素的顺时针旋转角度。
texture	【字段含义】bufferType 为 TRTCVideoBufferType_Texture 时的视频数据，承载用于 OpenGL 渲染的纹理数据。
timestamp	【字段含义】视频帧的时间戳，单位毫秒。 【推荐取值】自定义视频采集时可以设置为0。若该参数为0，SDK 会自定填充 timestamp 字段，但请“均匀”地控制 sendCustomVideoData 的调用间隔。
videoFormat	【字段含义】视频的像素格式。
width	【字段含义】视频宽度。

TRTCAudioFrame

TRTCAudioFrame

音频帧数据

枚举类型	描述
audioFormat	【字段含义】音频帧的格式。
channel	【字段含义】声道数。
data	【字段含义】音频数据。
extraData	【字段含义】音频额外数据，远端用户通过 `onLocalProcessedAudioFrame` 写入的数据会通过该字段回调。
extraDataLength	【字段含义】音频消息数据的长度。
length	【字段含义】音频数据的长度。
sampleRate	【字段含义】采样率。

timestamp	【字段含义】时间戳，单位ms。

TRTCMixUser

TRTCMixUser

云端混流中各路画面的描述信息

TRTCMixUser 用于指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
image	【字段含义】占位图或水印图 占位图是指当对应 userId 混流内容为纯音频时，混合后的画面中显示的是占位图片。 水印图是指一张贴在混合后画面中的半透明图片，这张图片会一直覆盖于混合后的画面上。 当指定 inputType 为 TRTCMixInputTypePureAudio 时，image 为占位图，此时需要您指定 userId。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，image 为水印图，此时不需要您指定 userId。 【推荐取值】默认值：空值，即不设置占位图或者水印图。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 “**控制台** => 应用管理 => 功能配置 => 素材管理” 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的“图片ID”，然后将“图片ID”转换成字符串类型并设置给 image 字段即可（比如假设“图片ID” 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。 image 仅在 inputType 为 TRTCMixInputTypePureAudio 或者 TRTCMixInputTypeWatermark 时才生效。
inputType	【字段含义】指定该路流的混合内容（只混音频、只混视频、混合音视频、混入水印）。 【默认取值】默认值：TRTCMixInputTypeUndefined。 【特别说明】 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 YES 时，等效于设置 inputType 为 TRTCMixInputTypePureAudio。 当指定 inputType 为 TRTCMixInputTypeUndefined 并设置 pureAudio 为 NO 时，等效于设置 inputType 为 TRTCMixInputTypeAudioVideo。 当指定 inputType 为 TRTCMixInputTypeWatermark 时，您可以不指定 userId 字段，但需要指定 image 字段。
pureAudio	【字段含义】指定该路流是不是只混合声音。 【推荐取值】默认值：false。

	【特别说明】已废弃，推荐使用8.5版本开始新引入的字段：inputType。
rect	【字段含义】指定该路画面的坐标区域（单位：像素）。
renderMode	【字段含义】该画面在输出时的显示模式。 【推荐取值】默认值：视频流默认为0。0为裁剪，1为缩放，2为缩放并显示黑底。 【特别说明】水印图和占位图暂时不支持设置 renderMode，默认强制拉伸处理。
roomId	【字段含义】该路音视频流所在的房间号（设置为空值代表当前用户所在的房间号）。
soundLevel	【字段含义】该路音频参与混音时的音量等级（取值范围：0 - 100）。 【默认取值】默认值：100。
streamType	【字段含义】指定该路画面是主路画面（TRTCVideoStreamTypeBig）还是辅路画面（TRTCVideoStreamTypeSub）。
userId	【字段含义】用户 ID
zOrder	【字段含义】指定该路画面的层级（取值范围：1 - 15，不可重复）。

TRTCTranscodingConfig

TRTCTranscodingConfig

云端混流的排版布局和转码参数

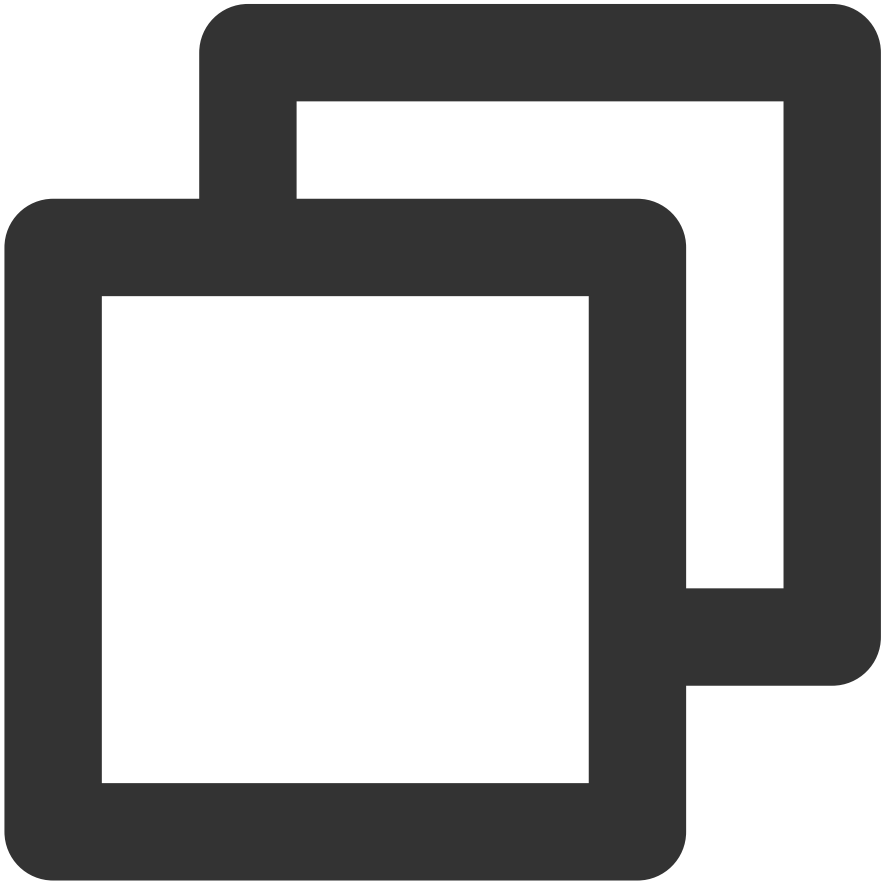
用于指定混流时各路画面的排版位置信息和云端转码的编码参数。

枚举类型	描述
appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
audioBitrate	【字段含义】指定云端转码的目标音频码率。 【推荐取值】默认值：64kbps，取值范围是 [32，192]。
audioChannels	【字段含义】指定云端转码的音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioCodec	【字段含义】指定云端转码的输出流音频编码类型。 【推荐取值】默认值：0，代表LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。

	【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。 【特别说明】HE-AAC 和 HE-AACv2 取值仅在输出流为您额外设置的 streamId 上时才生效。
audioSampleRate	【字段含义】指定云端转码的目标音频采样率。 【推荐取值】默认值：48000Hz。支持12000HZ、16000HZ、22050HZ、24000HZ、32000HZ、44100HZ、48000HZ。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为(97,158,241)。
backgroundImage	【字段含义】指定混合画面的背景图片。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 您可以将 image 设置为控制台中的某一个素材 ID，这需要您事先在 “**控制台** => 应用管理 => 功能配置 => 素材管理” 中单击 [新增图片] 按钮进行上传。 上传成功后可以获得对应的“图片ID”，然后将“图片ID”转换成字符串类型并设置给 image 字段即可（比如假设“图片ID” 为 63，可以设置 image = @"63"）。 您也可以将 image 设置为图片的 URL 地址，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
mixUsersArray	【字段含义】指定云端混流中每一路视频画面的位置、大小、图层以及流类型等信息。 【推荐取值】该字段是一个 TRTCMixUser 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
mixUsersArraySize	【字段含义】 数组 mixUsersArray 的元素个数。
mode	【字段含义】排版模式。 【推荐取值】请根据您的业务场景要求自行选择，预排版模式是适用性较好的一种模式。
streamId	【字段含义】输出到 CDN 上的直播流 ID。 【推荐取值】默认值：空值，即房间里的多路音视频流最终会混合到接口调用者的那一路音视频流上。 如不设置该参数，SDK 会执行默认逻辑，即房间里的多路音视频流会混合到该接口调用者的那一路音视频流上，也就是 A + B => A。 如您设置该参数，SDK 会将房间里的多路音视频流混合到您指定的直播流上，也就是 A + B => C（C 代表您指定的 streamId）。

版权所有：腾讯云计算（北京）有限责任公司

videoBitrate	【字段含义】指定云端转码的目标视频码率（kbps）。 【推荐取值】如果填0，TRTC 会根据 videoWidth 和 videoHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoFramerate	【字段含义】指定云端转码的目标视频帧率（FPS）。 【推荐取值】默认值：15fps，取值范围是 (0,30]。
videoGOP	【字段含义】指定云端转码的目标视频关键帧间隔（GOP）。 【推荐取值】默认值：2，单位为秒，取值范围是 [1,8]。
videoHeight	【字段含义】指定云端转码的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：



```
{
  "payLoadContent":"xxx",
  "payloadType":5,
  "payloadUuid":"1234567890abcdef1234567890abcdef",
```

	<pre>"interval":1000,
	"followIdr":false
	}</pre>
	当前支持的字段及含义：
	payloadContent: 必填。透传 sei 的 payload 内容，不能为空；
	payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）；
	payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字；
	interval: 选填，默认 1000。sei 的发送间隔，单位毫秒；
	followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。

videoWidth	【字段含义】指定云端转码的目标分辨率（宽度）。 【推荐取值】单位：像素值，推荐值：360，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。

TRTCPublishCDNParam

TRTCPublishCDNParam

向非腾讯云 CDN 上发布音视频流时需设置的转推参数

TRTC 的后台服务支持通过标准 RTMP 协议，将其中的音视频流发布到第三方直播 CDN 服务商。

如果您使用腾讯云直播 CDN 服务，可无需关注此参数，直接使用 startPublish 接口即可。

枚举类型	描述
appId	【字段含义】腾讯云直播服务的 AppID。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 appid。
bizId	【字段含义】腾讯云直播服务的 bizid。 【推荐取值】请在 **实时音视频控制台** 依次单击【应用管理】=>【应用信息】，并在【旁路直播信息】中获取 bizid。
streamId	【字段含义】需要转推的 streamId。 【推荐取值】默认值：空值。如果不填写，则默认转推调用者的旁路流。
url	【字段含义】指定该路音视频流在第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。

	【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCAudioRecordingParams

TRTCAudioRecordingParams

本地音频文件的录制参数

该参数用于在音频录制接口 startAudioRecording 中指定录制参数。

枚举类型	描述
filePath	【字段含义】录音文件的保存路径（必填）。 【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录音文件的格式，目前支持的格式有 PCM、WAV 和 AAC。 例如：假如您指定路径为 "mypath/record/audio.aac"，代表您希望 SDK 生成一个 AAC 格式的音频录制文件。 请您指定一个有读写权限的合法路径，否则录音文件无法生成。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值10000。默认值为0，表示不分片。
recordingContent	【字段含义】音频录制内容类型。 【特别说明】默认录制所有本地和远端音频。

TRTCLocalRecordingParams

TRTCLocalRecordingParams

本地媒体文件的录制参数

该参数用于在本地媒体文件的录制接口 startLocalRecording 中指定录制相关参数。

接口 startLocalRecording 是接口 startAudioRecording 的能力加强版本，前者可以录制视频文件，后者只能录制音频文件。

枚举类型	描述
filePath	【字段含义】录制的文件地址（必填），请确保路径有读写权限且合法，否则录制文件无法生成。 【特别说明】该路径需精确到文件名及格式后缀，格式后缀用于决定录制出的文件格式，目前支持的格式暂时只有 MP4。 例如：假如您指定路径为 "mypath/record/test.mp4"，代表您希望 SDK 生成一个 MP4 格式的本地视频文件。

	请您指定一个有读写权限的合法路径，否则录制文件无法生成。
interval	【字段含义】interval 录制信息更新频率，单位毫秒，有效范围：1000-10000。默认值为-1，表示不回调。
maxDurationPerFile	【字段含义】maxDurationPerFile 录制文件分片时长，单位毫秒，最小值10000。默认值为0，表示不分片。
recordType	【字段含义】媒体录制类型，默认值：TRTCRecordTypeBoth，即同时录制音频和视频。

TRTCSwitchRoomConfig

TRTCSwitchRoomConfig

房间切换参数

该参数用于切换房间接口switchRoom，可以让用户从一个房间快速切换到另一个房间。

枚举类型	描述
privateMapKey	【字段含义】用于权限控制的权限票据（选填），当您希望某个房间只能让特定的 userId 进入时，需要使用 privateMapKey 进行权限保护。 【推荐取值】仅建议有高级别安全需求的客户使用，更多详情请参见 进房权限保护。
roomId	【字段含义】数字房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
strRoomId	【字段含义】字符串房间号码 [选填]，在同一个房间内的用户可以看到彼此并能够进行音视频通话。 【特别说明】roomId 和 strRoomId 必须并且只能填一个。若两者都填，则优先选择 roomId。
userSig	【字段含义】用户签名 [选填]，当前 userId 对应的验证签名，相当于登录密码。 如果您在切换房间时不指定新计算出的 userSig，SDK 会继续使用您在进入房间时（enterRoom）时所指定的 userSig。 这就需要您必须保证旧的 userSig 在切换房间的那一刻仍在签名允许的效期内，否则会导致房间切换失败。 【推荐取值】具体计算方法请参见 如何计算UserSig。

TRTCAudioFrameDelegateFormat

TRTCAudioFrameDelegateFormat

音频自定义回调的格式参数

该参数用于在音频自定义回调相关的接口中，设置 SDK 回调出来的音频数据的相关格式（包括采样率、声道数等）。

枚举类型	描述
channel	【字段含义】声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
mode	【字段含义】回调数据读写模式 【推荐取值】TRTCAudioFrameOperationModeReadOnly：仅从回调中获取音频数据。可设定的模式有 TRTCAudioFrameOperationModeReadOnly，TRTCAudioFrameOperationModeReadWrite。
sampleRate	【字段含义】采样率。 【推荐取值】默认值：48000Hz。支持 16000, 32000, 44100, 48000。
samplesPerCall	【字段含义】采样点数。 【推荐取值】取值必须是 sampleRate/100 的整数倍。

TRTCImageBuffer

TRTCImageBuffer

TRTC 屏幕分享图标信息以及 mute image 垫片

枚举类型	描述
buffer	图像存储的内容，一般为 BGRA 结构。
height	图像的高度。
length	图像数据的大小。
width	图像的宽度。

TRTCScreenCaptureSourceInfo

TRTCScreenCaptureSourceInfo

屏幕分享的目标信息（仅适用于桌面系统）

在用户进行屏幕分享时，可以选择抓取整个桌面，也可以仅抓取某个程序的窗口。
TRTCScreenCaptureSourceInfo 用于描述待分享目标的信息，包括 ID、名称、缩略图等，该结构体中的字段信息均是只读的。

枚举类型	描述
height	【字段含义】屏幕/窗口高，单位:像素点。
iconBGRA	【字段含义】分享窗口的图标。
isMainScreen	【字段含义】是否为主显示屏（适用于多显示器的情况）。
isMinimizeWindow	【字段含义】是否为最小化窗口。
sourceId	【字段含义】采集源的ID，对于窗口，该字段代表窗口的 ID；对于屏幕，该字段代表显示器的 ID。
sourceName	【字段含义】采集源名称（采用 UTF8 编码）。
thumbBGRA	【字段含义】分享窗口的缩略图。
type	【字段含义】采集源类型（是分享整个屏幕？还是分享某个窗口？）。
width	【字段含义】屏幕/窗口宽，单位:像素点。
x	【字段含义】屏幕/窗口 x 坐标，单位:像素点。
y	【字段含义】屏幕/窗口 y 坐标，单位:像素点。

TRTCScreenCaptureProperty

TRTCScreenCaptureProperty

屏幕分享的进阶控制参数

该参数用于屏幕分享相关的接口selectScreenCaptureTarget，用于在指定分享目标时设定一系列进阶控制参数。
比如：是否采集鼠标、是否要采集子窗口、是否要在被分享目标周围绘制一个边框等。

枚举类型	描述
enableCaptureChildWindow	【字段含义】窗口采集时是否采集子窗口（需要子窗口与被采集窗口具有 Owner 或 Popup 属性），默认为 false。
enableCaptureMouse	【字段含义】是否采集目标内容的同时采集鼠标，默认为 true。

版权所有：腾讯云计算（北京）有限责任公司

enableHighLight	【字段含义】是否高亮正在共享的窗口（在被分享目标周围绘制一个边框），默认为 true。
enableHighPerformance	【字段含义】是否开启高性能模式（只会在分享屏幕时会生效），默认为 true。 【特殊说明】开启后屏幕采集性能最佳，但会丧失抗遮挡能力，如果您同时开启 enableHighLight + enableHighPerformance，远端用户可以看到高亮的边框。
highLightColor	【字段含义】指定高亮边框的颜色，RGB 格式，传入 0 时代表采用默认颜色，默认颜色为 #FFE640。
highLightWidth	【字段含义】指定高亮边框的宽度，传入0时采用默认描边宽度，默认宽度为 5px，您可以设置的最大值为 50。

TRTCAudioParallelParams

TRTCAudioParallelParams

远端音频流智能并发播放策略的参数

该参数用于设置远端音频流智能并发播放策略。

枚举类型	描述
includeUsersCount	【字段含义】指定用户必定能并发播放。 【特殊说明】指定必定并发播放的用户 ID 列表。这些用户不参与智能选择。includeUsers 的数量必须小于 maxCount，否则本次并发播放设置失效。includeUsers 仅在 maxCount > 0 时有效。当 includeUsers 生效时，参与智能并发选择的最大播放数 = maxCount - 有效 includeUsers 的数量。
maxCount	【字段含义】最大并发播放数。默认值：0。 如果 maxCount > 0，且实际人数 > maxCount，会实时智能选出 maxCount 路数据进行播放，这会极大的降低性能消耗。 如果 maxCount = 0，SDK 不限制并发播放数，在上麦人数比较多的房间可能会引发性能问题。

TRTCUser

TRTCUser

媒体流发布相关配置的用户信息

您可以通过设置该参数，配合媒体流目标发布参数 (TRTCPublishTarget) 和混流转码参数 (TRTCStreamMixingConfig)，将您指定的多路音视频流进行转码并发布到您填写的目标发布地址中

枚举类型	描述
intRoomId	【字段含义】数字房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【推荐取值】取值范围：1 - 4294967294。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 strRoomId，则 intRoomId 需要填写为0。若两者都填，SDK 将优先选用 intRoomId。
strRoomId	【字段含义】字符串房间号，需要和您的进房参数 (TRTCParams) 中的房间号类型相匹配。 【特别说明】intRoomId 与 strRoomId 是互斥的，若您进房参数中选用 roomId，则 strRoomId 无需填写。若两者都填，SDK 将优先选用 intRoomId。 【推荐取值】限制长度为 64 字节。以下为支持的字符集范围（共 89 个字符）： 大小写英文字母（a-zA-Z）； 数字（0-9）； 空格、 ! 、 # 、 $ 、 % 、 & 、 (、) 、 + 、 - 、 : 、 ; 、 < 、 = 、 . 、 > 、 ? 、 @ 、 [、] 、 ^ 、 _ 、 { 、 } 、 | 、 ~ 、 , 。
userId	【字段含义】用户标识，当前用户的 userId，相当于用户名，使用 UTF-8 编码。 【推荐取值】如果一个用户在您的账号系统中的 ID 为“mike”，则 userId 即可设置为“mike”。

TRTCPublishCdnUrl

TRTCPublishCdnUrl

向腾讯或者第三方 CDN 上发布音视频流时需设置的 url 配置

该配置用于媒体流发布接口 startPublishMediaStream 中的目标推流配置 (TRTCPublishTarget)

枚举类型	描述
isInternalLine	【字段含义】指定该路音视频流是否发布至腾讯云。 【推荐取值】默认值：true。 【特别说明】若您的目标直播服务商为腾讯，请将此参数设置为 true，此时后台计费系统不会对此计算转推服务费。
rtmpUrl	【字段含义】指定该路音视频流在腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【推荐取值】各家服务商的推流地址规则差异较大，请根据目标服务商的要求填写合法的推流 URL，TRTC 的后台服务器会按照您填写的 URL 向第三方服务商推送标准格式音视频流。

	【特别说明】推流 URL 必须为 RTMP 格式，必须符合您的目标直播服务商的规范要求，否则目标服务商会拒绝来自 TRTC 后台服务的推流请求。

TRTCPublishTarget

TRTCPublishTarget

目标推流配置

该配置用于媒体流发布接口 startPublishMediaStream。

枚举类型	描述
cdnUrlList	【字段含义】发布至腾讯或者第三方直播服务商的推流地址（RTMP 格式）。 【特别说明】若您的 mode 选择为 TRTCPublishMixStreamToRoom，此时您不需要设置该参数。
cdnUrlListSize	【字段含义】数组 cdnUrlList 的元素个数。 【特别说明】若您的 mode 选择为 TRTCPublishMixStreamToRoom，此时您不需要设置该参数。
mixStreamIdentity	【字段含义】回推房间机器人信息。 【特别说明】仅当您的 mode 选择为 TRTCPublishMixStreamToRoom 时，您需要设置该参数。 【特别说明】设置后，该路转码音视频数据将回推到您指定的房间中。建议设置为特殊的 userId，以避免难以区分回推机器人和您通过 TRTC SDK 进房的主播。 【特别说明】参与混流的用户不支持订阅该转码流。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为手动时，您需要自行管理您想要拉取的音视频流（通常当您拉取回推房间的转码流时，您应该取消订阅参与转码的对应音视频单流）。 【特别说明】当您进房前设置的订阅模式(setDefaultStreamRecvMode) 均为自动时，不参与转码的用户将自动收到后台下发的转码流并不再继续接收参与转码的音视频单流。除非您明确进行取消订阅（muteRemoteVideoStream 和 muteRemoteAudio），否则转码流数据将持续下发。
mode	【字段含义】媒体流发布模式。 【推荐取值】请根据您的业务场景要求自行选择，TRTC 支持转推、转码和回推到 RTC 房间的模式。 【特别说明】若您的业务场景需要多个发布模式，您可以通过多次调用媒体流发布接口 (startPublishMediaStream) 并分别设置不同的 TRTCPublishTarget。 【特别说明】同一个 mode 请对应一个媒体流发布接口 (startPublishMediaStream)，并在后续需要调整时，使用 updatePublishCDNStream 进行更新。

TRTCVideoLayout

TRTCVideoLayout

转码视频布局

该配置用于媒体流发布接口（startPublishMediaStream）中的转码配置（TRTCStreamMixingConfig）。
用于指定转码流中每一路视频画面的位置、大小、图层以及流类型等信息。

枚举类型	描述
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如："0x61B9F1" 代表 RGB 分别为 (97,158,241)。
fillMode	【字段含义】画面填充模式。 【推荐取值】填充（画面可能会被拉伸裁剪）或适应（画面可能会有黑边），默认值：TRTCVideoFillMode_Fill。
fixedVideoStreamType	【字段含义】指定该路画面是主路画面（TRTCVideoStreamTypeBig）还是辅路画面（TRTCVideoStreamTypeSub）。
fixedVideoUser	【字段含义】参与转码的用户信息。 【特别说明】用户信息 (TRTCUser) 支持不填写（即 userId、intRoomId 和 strRoomId 均不填写）。此时当发起混流的房间中有主播上行音视频数据时，TRTC 后台服务器将自动将对应主播音视频填充到您指定的布局中。
placeHolderImage	【字段含义】占位图 URL，即当指定用户暂时仅上行音频时，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置占位图。 【特别说明】此时需要您指定用户信息 fixedVideoUser 中的 userId。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为占位图。
rect	【字段含义】指定该路画面的坐标区域（单位：像素）。
zOrder	【字段含义】指定该路画面的层级（取值范围：0 - 15，不可重复）。

TRTCWatermark

TRTCWatermark

水印布局

该配置用于媒体流发布接口 (startPublishMediaStream) 中的转码配置 (TRTCStreamMixingConfig)

枚举类型	描述
rect	【字段含义】指定该路水印画面的坐标区域（单位：像素）。
watermarkUrl	【字段含义】水印 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为水印。
zOrder	【字段含义】指定该路水印画面的层级（取值范围：0 - 15，不可重复）。

TRTCStreamEncoderParam

TRTCStreamEncoderParam

媒体流编码输出参数

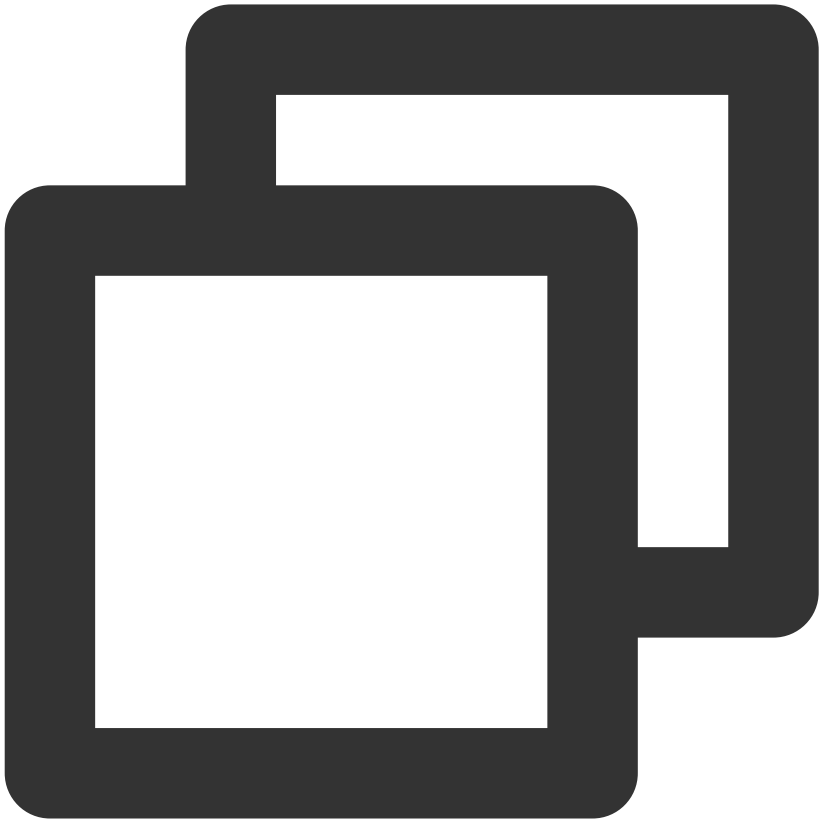
【字段含义】该配置用于媒体流发布接口（startPublishMediaStream）。

【特别说明】当您的发布目标（TRTCPublishTarget）中的 mode 配置为 TRTCPublish_MixStream_ToCdn 或者 TRTCPublish_MixStream_ToRoom 时，该参数为必填。

【特别说明】当您使用转推服务（mode 为 TRTCPublish_BigStream_ToCdn 或者 TRTCPublish_SubStream_ToCdn）时，为了更好的转推稳定性以及更好的 CDN 播放兼容性，也建议您填写该配置的具体参数。

枚举类型	描述
audioEncodedChannelNum	【字段含义】指定媒体发布流的目标音频声道数。 【推荐取值】默认值：1，代表单声道。可设定的数值只有两个数字：1-单声道，2-双声道。
audioEncodedCodecType	【字段含义】指定媒体发布流的目标音频编码类型。 【推荐取值】默认值：0，代表 LC-AAC。可设定的数值只有三个数字：0 - LC-AAC，1 - HE-AAC，2 - HE-AACv2。 【特别说明】HE-AAC 和 HE-AACv2 支持的输出流音频采样率范围为[48000, 44100, 32000, 24000, 16000]。 【特别说明】当音频编码设置为 HE-AACv2 时，只支持输出流音频声道数为双声道。
audioEncodedKbps	【字段含义】指定媒体发布流的目标音频码率（kbps）。 【推荐取值】默认值：50kbps，取值范围是 [32，192]。

audioEncodedSampleRate	【字段含义】指定媒体发布流的目标音频采样率。 【推荐取值】默认值：48000Hz。取值为 [48000, 44100, 32000, 24000, 16000, 8000]，单位是 Hz。
videoEncodedCodecType	【字段含义】指定媒体发布流的目标视频编码类型。 【推荐取值】默认值：0，代表 H264。可设定的数值只有两个数字：0 - H264，1 - H265。
videoEncodedFPS	【字段含义】指定媒体发布流的目标视频帧率（FPS）。 【推荐取值】推荐值：20fps，取值范围是 (0,30]。
videoEncodedGOP	【字段含义】指定媒体发布流的目标视频关键帧间隔（GOP）。 【推荐取值】推荐值：3，单位为秒，取值范围是 [1,5]。
videoEncodedHeight	【字段含义】指定媒体发布流的目标分辨率（高度）。 【推荐取值】单位：像素值，推荐值：640，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoEncodedKbps	【字段含义】指定媒体发布流的目标视频码率（kbps）。 【推荐取值】如果填 0，TRTC 会根据 videoEncodedWidth 和 videoEncodedHeight 估算出一个合理的码率值，您也可以参考视频分辨率枚举定义中所推荐的码率值（见注释部分）。
videoEncodedWidth	【字段含义】指定媒体发布流的目标分辨率（宽度）。 【推荐取值】单位：像素值，推荐值：368，如果你只混合音频流，请将 width 和 height 均设置为 0，否则混流转码后的直播流中会有黑色背景。
videoSeiParams	【字段含义】混流 SEI 参数，默认不填写。 【特别说明】参数以 json 字符串形式传入，示例如下：



```
{
  "payLoadContent":"xxx",
  "payloadType":5,
  "payloadUuid":"1234567890abcdef1234567890abcdef",
  "interval":1000,
  "followIdr":false
}
```

当前支持的字段及含义：
payloadContent: 必填。透传 sei 的 payload 内容，不能为空；
payloadType: 必填。sei 消息的类型，取值范围: 5 或 [100, 254] 范围内的整数（244 除外，244 是内部自定义的时间戳 sei）；
payloadUuid: 当 payloadType 为 5 时必须填写，其他情况下该值会被忽略。该值必须是长度为 32 的十六进制数字；
interval: 选填，默认 1000。sei 的发送间隔，单位毫秒；
followIdr: 选填，默认 false。该值为 true 时，发送关键帧时会确保带 sei，否则不确保。

TRTCStreamMixingConfig

TRTCStreamMixingConfig

媒体流转码配置参数

该配置用于媒体流发布接口（startPublishMediaStream）。

用于指定转码时各路画面的排版位置信息和输入的音频信息。

枚举类型	描述
audioMixUserList	【字段含义】指定转码流中的每一路输入音频的信息。 【推荐取值】该字段是一个 TRTCUser 类型的数组，数组中的每一个元素都用来代表每一路输入音频的信息。 【特别说明】用户信息支持不填写（即 audioMixUserList 为空）。此时若设置了 TRTCStreamEncoderParam 中音频相关编码输出参数，TRTC 后台服务器将自动将所有主播的音频混合输出（当前仅支持最高 16 路音视频输入）。
audioMixUserListSize	【字段含义】 数组 audioMixUserList 的元素个数。
backgroundColor	【字段含义】指定混合画面的底色颜色。 【推荐取值】默认值：0x000000 代表黑色。格式为十六进制数字，比如：“0x61B9F1” 代表 RGB 分别为 （97、158、241）。
backgroundImage	【字段含义】指定混合画面的背景图 URL，腾讯云的后台服务器会将该 URL 地址指定的图片混合到最终的画面中。 【推荐取值】默认值：空值，即不设置背景图片。 【特别说明】 URL 链接长度限制为 512 字节。图片大小限制不超过 2MB。 图片格式支持 png、jpg、jpeg、bmp 格式，推荐使用 png 格式的半透明图片作为背景图。
videoLayoutList	【字段含义】指定混合画面中的每一路视频画面的位置、大小、图层以及流类型等信息。 【推荐取值】该字段是一个 TRTCVideoLayout 类型的数组，数组中的每一个元素都用来代表每一路画面的信息。
videoLayoutListSize	【字段含义】 数组 videoLayoutList 的元素个数。
watermarkList	【字段含义】指定混合画面中的每一路水印画面的位置、大小、图层等信息。 【推荐取值】该字段是一个 TRTCWatermark 类型的数组，数组中的每一个元素都用来代表每一路水印的信息。
watermarkListSize	【字段含义】 数组 watermarkList 的元素个数。

TRTCPayloadPrivateEncryptionConfig

版权所有：腾讯云计算（北京）有限责任公司

TRTCPayloadPrivateEncryptionConfig

媒体流私有加密配置

该配置用于设置媒体流私有加密的算法和密钥。

枚举类型	描述
encryptionAlgorithm	【字段含义】加密算法，默认为 TRTCEncryptionAlgorithmAes128Gcm。
encryptionKey	【字段含义】加密用密钥，字符串类型。 【推荐取值】若加密算法为 TRTCEncryptionAlgorithmAes128Gcm，密匙长度需为 16 字节，若加密算法为 TRTCEncryptionAlgorithmAes256Gcm，密匙长度需为 32 字节。
encryptionSalt[32]	【字段含义】盐，加密用初始向量。 【推荐取值】需确保填入该参数的数组不为空、不全为 0 且数据长度为 32 字节。

TRTCAudioVolumeEvaluateParams

TRTCAudioVolumeEvaluateParams

音量评估等相关参数设置

该设置用于开启人声检测、声音频谱计算。

枚举类型	描述
enablePitchCalculation	【字段含义】是否开启本地人声频率计算
enableSpectrumCalculation	【字段含义】是否开启声音频谱计算。
enableVadDetection	【字段含义】是否开启本地人声检测。 【请您注意】在 startLocalAudio 之前调用才可以生效。
interval	【字段含义】设置 onUserVoiceVolume 回调的触发间隔，单位为毫秒，最小间隔为 100ms，如果小于等于 0 则会关闭回调。 【推荐取值】推荐值：300，单位为毫秒。 【特别说明】interval 大于 0 时，音量提示将默认开启，无需另外设置。

废弃接口

最近更新时间：2024-06-06 15:26:15

Copyright (c) 2022 Tencent. All rights reserved.

Module: DeprecatedTRTCCloud @ TXLiteAVSDK

Function: TRTC 废弃接口

废弃接口

IDeprecatedTRTCCloud

函数列表	描述
enableAudioVolumeEvaluation	启用音量大小提示
enableAudioVolumeEvaluation	启用音量大小提示
startLocalAudio	设置音频质量
startRemoteView	开始显示远端视频画面
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流
setLocalViewFillMode	设置本地图像的渲染模式
setLocalViewRotation	设置本地图像的顺时针旋转角度
setLocalViewMirror	设置本地摄像头预览画面的镜像模式
setRemoteViewFillMode	设置远端图像的渲染模式
setRemoteViewRotation	设置远端图像的顺时针旋转角度
startRemoteSubStreamView	开始显示远端用户的辅路画面
stopRemoteSubStreamView	停止显示远端用户的辅路画面
setRemoteSubStreamViewFillMode	设置辅路画面的填充模式
setRemoteSubStreamViewRotation	设置辅路画面的顺时针旋转角度

setAudioQuality	设置音频质量
setPriorRemoteVideoStreamType	设定优先观看大画面还是小画面
setMicVolumeOnMixing	设置麦克风音量大小
playBGM	启动播放背景音乐
stopBGM	停止播放背景音乐
pauseBGM	停止播放背景音乐
resumeBGM	停止播放背景音乐
getBGMDuration	获取背景音乐总时长（单位：毫秒）
setBGMPosition	设置背景音乐的播放进度
setBGMVolume	设置背景音乐的音量大小
setBGMPlayoutVolume	设置背景音乐的本地播放音量
setBGMPublishVolume	设置背景音乐的远端播放音量
playAudioEffect	播放音效
setAudioEffectVolume	设置音效音量
stopAudioEffect	停止播放音效
stopAllAudioEffects	停止所有音效
setAllAudioEffectsVolume	设置所有音效音量
pauseAudioEffect	暂停音效
resumeAudioEffect	暂停音效
enableCustomVideoCapture	启用视频自定义采集模式
sendCustomVideoData	投送自己采集的视频数据
muteLocalVideo	暂停/恢复发布本地的视频流
muteRemoteVideoStream	暂停 / 恢复订阅远端用户的视频流
startSpeedTest	开始进行网络测速（进入房间前使用）
startScreenCapture	启动屏幕分享

版权所有：腾讯云计算（北京）有限责任公司

setLocalVideoProcessCallback	设置第三方美颜的视频数据回调
getCameraDevicesList	获取摄像头设备列表
setCurrentCameraDevice	选定当前要使用的摄像头
getCurrentCameraDevice	获取当前使用的摄像头
getMicDevicesList	获取麦克风设备列表
getCurrentMicDevice	获取当前的麦克风设备
setCurrentMicDevice	选定当前使用的麦克风
getCurrentMicDeviceVolume	获取当前麦克风的设备音量
setCurrentMicDeviceVolume	设置当前麦克风的设备音量
setCurrentMicDeviceMute	设置系统当前麦克风设备的静音状态
getCurrentMicDeviceMute	获取系统当前麦克风设备是否静音
getSpeakerDevicesList	获取扬声器设备列表
getCurrentSpeakerDevice	获取当前的扬声器设备
setCurrentSpeakerDevice	设置要使用的扬声器
getCurrentSpeakerVolume	获取当前扬声器的设备音量
setCurrentSpeakerVolume	设置当前扬声器的设备音量
getCurrentSpeakerDeviceMute	获取系统当前扬声器设备是否静音
setCurrentSpeakerDeviceMute	设置系统当前扬声器设备的静音状态
startCameraDeviceTest	开始进行摄像头测试
startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止进行摄像头测试
startMicDeviceTest	开始进行麦克风测试
stopMicDeviceTest	开始进行麦克风测试
startSpeakerDeviceTest	开始进行扬声器测试
stopSpeakerDeviceTest	停止进行扬声器测试

版权所有：腾讯云计算（北京）有限责任公司

selectScreenCaptureTarget	开始应用内的屏幕分享（iOS）
setVideoEncoderRotation	设置视频编码器输出的画面方向
setVideoEncoderMirror	设置编码器输出的画面镜像模式

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(uint32_t interval)

启用音量大小提示

@deprecated v10.1 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

enableAudioVolumeEvaluation

enableAudioVolumeEvaluation

void enableAudioVolumeEvaluation	(uint32_t interval
	bool enable_vad)

启用音量大小提示

@deprecated v11.2 版本开始不推荐使用，建议使用 enableAudioVolumeEvaluation(enable, params) 替代之。

startLocalAudio

startLocalAudio

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

startRemoteView

startRemoteView

void startRemoteView	(const char* userId

	TXView rendView)

开始显示远端视频画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteView

stopRemoteView

void stopRemoteView	(const char* userId)

停止显示远端视频画面，同时不再拉取该远端用户的视频数据流

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setLocalViewFillMode

setLocalViewFillMode

void setLocalViewFillMode	(TRTCVideoFillMode mode)

设置本地图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setLocalViewRotation

setLocalViewRotation

void setLocalViewRotation	(TRTCVideoRotation rotation)

设置本地图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setLocalViewMirror

setLocalViewMirror

void setLocalViewMirror	(bool mirror)

设置本地摄像头预览画面的镜像模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setLocalRenderParams 替代之。

setRemoteViewFillMode

setRemoteViewFillMode

void setRemoteViewFillMode	(const char* userId
	TRTCVideoFillMode mode)

设置远端图像的渲染模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteViewRotation

setRemoteViewRotation

void setRemoteViewRotation	(const char* userId
	TRTCVideoRotation rotation)

设置远端图像的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

startRemoteSubStreamView

startRemoteSubStreamView

void startRemoteSubStreamView	(const char* userId
	TXView rendView)

开始显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView 替代之。

stopRemoteSubStreamView

stopRemoteSubStreamView

void stopRemoteSubStreamView	(const char* userId)

停止显示远端用户的辅路画面

@deprecated v8.0 版本开始不推荐使用，建议使用 stopRemoteView 替代之。

setRemoteSubStreamViewFillMode

setRemoteSubStreamViewFillMode

void setRemoteSubStreamViewFillMode	(const char* userId
	TRTCVideoFillMode mode)

设置辅路画面的填充模式

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams 替代之。

setRemoteSubStreamViewRotation

setRemoteSubStreamViewRotation

void setRemoteSubStreamViewRotation	(const char* userId
	TRTCVideoRotation rotation)

设置辅路画面的顺时针旋转角度

@deprecated v8.0 版本开始不推荐使用，建议使用 setRemoteRenderParams:streamType:params: 替代之。

setAudioQuality

setAudioQuality

void setAudioQuality	(TRTCAudioQuality quality)

设置音频质量

@deprecated v8.0 版本开始不推荐使用，建议使用 startLocalAudio 替代之。

setPriorRemoteVideoStreamType

setPriorRemoteVideoStreamType

void setPriorRemoteVideoStreamType	(TRTCVideoStreamType type)

设定优先观看大画面还是小画面

@deprecated v8.0 版本开始不推荐使用，建议使用 startRemoteView:streamType:view: 替代之。

setMicVolumeOnMixing

setMicVolumeOnMixing

void setMicVolumeOnMixing	(uint32_t volume)

设置麦克风音量大小

playBGM

playBGM

void playBGM	(const char* path)

启动播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

stopBGM

stopBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

pauseBGM

pauseBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

resumeBGM

resumeBGM

停止播放背景音乐

@deprecated v7.3 版本开始不推荐使用，建议使用 getAudioEffectManager 替代之。

getBGMDuration

getBGMDuration

uint32_t getBGMDuration	(const char* path)

获取背景音乐总时长（单位：毫秒）

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 getMusicDurationInMS 替代之。

setBGMPosition

setBGMPosition

void setBGMPosition	(uint32_t pos)

设置背景音乐的播放进度

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 seekMusicToPosInMS 替代之。

setBGMVolume

setBGMVolume

void setBGMVolume	(uint32_t volume)

设置背景音乐的音量大小

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicVolume 替代之。

setBGMPlayoutVolume

setBGMPlayoutVolume

void setBGMPlayoutVolume	(uint32_t volume)

设置背景音乐的本地播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPlayoutVolume 替代之。

setBGMPublishVolume

setBGMPublishVolume

void setBGMPublishVolume	(uint32_t volume)

设置背景音乐的远端播放音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setBGMPublishVolume 替代之。

playAudioEffect

playAudioEffect

void playAudioEffect	(TRTCAudioEffectParam* effect)

播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 startPlayMusic 替代之。

setAudioEffectVolume

setAudioEffectVolume

void setAudioEffectVolume	(int effectId
	int volume)

设置音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

stopAudioEffect

stopAudioEffect

void stopAudioEffect	(int effectId)

停止播放音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

stopAllAudioEffects

stopAllAudioEffects

停止所有音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 stopPlayMusic 替代之。

setAllAudioEffectsVolume

setAllAudioEffectsVolume

void setAllAudioEffectsVolume	(int volume)

设置所有音效音量

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 setMusicPublishVolume 和 setMusicPlayoutVolume 替代之。

pauseAudioEffect

pauseAudioEffect

void pauseAudioEffect	(int effectId)

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 pauseAudioEffect 替代之。

resumeAudioEffect

resumeAudioEffect

void resumeAudioEffect	(int effectId)

暂停音效

@deprecated v7.3 版本开始不推荐使用，建议使用 TXAudioEffectManager 中的 resumePlayMusic 替代之。

enableCustomVideoCapture

enableCustomVideoCapture

void enableCustomVideoCapture	(bool enable)

启用视频自定义采集模式

@deprecated v8.5 版本开始不推荐使用，建议使用 enableCustomVideoCapture(streamType, enable) 接口替代之。

sendCustomVideoData

sendCustomVideoData

void sendCustomVideoData	(TRTCVideoFrame* frame)

投送自己采集的视频数据

@deprecated v8.5 版本开始不推荐使用，建议使用 sendCustomVideoData(streamType, TRTCVideoFrame) 接口替代之。

muteLocalVideo

muteLocalVideo

void muteLocalVideo	(bool mute)

暂停/恢复发布本地的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteLocalVideo(streamType, mute) 接口替代之。

muteRemoteVideoStream

muteRemoteVideoStream

void muteRemoteVideoStream	(const char* userId
	bool mute)

暂停 / 恢复订阅远端用户的视频流

@deprecated v8.9 版本开始不推荐使用，建议使用 muteRemoteVideoStream(userId, streamType, mute) 接口替代之。

startSpeedTest

startSpeedTest

void startSpeedTest	(uint32_t sdkAppId
	const char* userId
	const char* userSig)

开始进行网络测速（进入房间前使用）

@deprecated v9.2 版本开始不推荐使用，建议使用 startSpeedTest(params) 接口替代之。

startScreenCapture

startScreenCapture

void startScreenCapture	(TXView rendView)

启动屏幕分享

@deprecated v7.2 版本开始不推荐使用，建议使用 startScreenCapture 替代之。

setLocalVideoProcessCallback

setLocalVideoProcessCallback

int setLocalVideoProcessCallback	(TRTCVideoPixelFormat pixelFormat
	TRTCVideoBufferType bufferType
	ITRTCVideoFrameCallback* callback)

设置第三方美颜的视频数据回调

@deprecated v11.4 版本开始不推荐使用，建议使用 enableLocalVideoCustomProcess 和 setLocalVideoCustomProcessCallback 替代之。

getCameraDevicesList

getCameraDevicesList

获取摄像头设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

setCurrentCameraDevice

setCurrentCameraDevice

void setCurrentCameraDevice	(const char* deviceId)

选定当前要使用的摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentCameraDevice

getCurrentCameraDevice

获取当前使用的摄像头

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

getMicDevicesList

getMicDevicesList

获取麦克风设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

getCurrentMicDevice

getCurrentMicDevice

获取当前的麦克风设备

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

setCurrentMicDevice

setCurrentMicDevice

void setCurrentMicDevice	(const char* micId)

选定当前使用的麦克风

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentMicDeviceVolume

getCurrentMicDeviceVolume

获取当前麦克风的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceVolume 接口替代之。

setCurrentMicDeviceVolume

setCurrentMicDeviceVolume

void setCurrentMicDeviceVolume	(uint32_t volume)

设置当前麦克风的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceVolume 接口替代之。

setCurrentMicDeviceMute

setCurrentMicDeviceMute

void setCurrentMicDeviceMute	(bool mute)

设置系统当前麦克风设备的静音状态

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceMute 接口替代之。

getCurrentMicDeviceMute

getCurrentMicDeviceMute

获取系统当前麦克风设备是否静音

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceMute 接口替代之。

getSpeakerDevicesList

getSpeakerDevicesList

获取扬声器设备列表

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getDevicesList 接口替代之。

getCurrentSpeakerDevice

getCurrentSpeakerDevice

获取当前的扬声器设备

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDevice 接口替代之。

版权所有：腾讯云计算（北京）有限责任公司

setCurrentSpeakerDevice

setCurrentSpeakerDevice

void setCurrentSpeakerDevice	(const char* speakerId)

设置要使用的扬声器

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDevice 接口替代之。

getCurrentSpeakerVolume

getCurrentSpeakerVolume

获取当前扬声器的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceVolume 接口替代之。

setCurrentSpeakerVolume

setCurrentSpeakerVolume

void setCurrentSpeakerVolume	(uint32_t volume)

设置当前扬声器的设备音量

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceVolume 接口替代之。

getCurrentSpeakerDeviceMute

getCurrentSpeakerDeviceMute

获取系统当前扬声器设备是否静音

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 getCurrentDeviceMute 接口替代之。

setCurrentSpeakerDeviceMute

setCurrentSpeakerDeviceMute

void setCurrentSpeakerDeviceMute	(bool mute)

设置系统当前扬声器设备的静音状态

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 setCurrentDeviceMute 接口替代之。

startCameraDeviceTest

startCameraDeviceTest

void startCameraDeviceTest	(TXView renderView)

开始进行摄像头测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startCameraDeviceTest 接口替代之。

startCameraDeviceTest

startCameraDeviceTest

void startCameraDeviceTest	(ITRTCVideoRenderCallback* callback)

开始进行摄像头测试

stopCameraDeviceTest

stopCameraDeviceTest

停止进行摄像头测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopCameraDeviceTest 接口替代之。

startMicDeviceTest

startMicDeviceTest

void startMicDeviceTest	(uint32_t interval)

开始进行麦克风测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startMicDeviceTest 接口替代之。

stopMicDeviceTest

stopMicDeviceTest

开始进行麦克风测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopMicDeviceTest 接口替代之。

startSpeakerDeviceTest

startSpeakerDeviceTest

void startSpeakerDeviceTest	(const char* testAudioFilePath)

开始进行扬声器测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 startSpeakerDeviceTest 接口替代之。

stopSpeakerDeviceTest

stopSpeakerDeviceTest

停止进行扬声器测试

@deprecated v8.0 版本开始不推荐使用，建议使用 TXDeviceManager 中的 stopSpeakerDeviceTest 接口替代之。

selectScreenCaptureTarget

selectScreenCaptureTarget

void selectScreenCaptureTarget	(const TRTCScreenCaptureSourceInfo& source
	const RECT& captureRect
	bool captureMouse = true
	bool highlightWindow = true)

开始应用内的屏幕分享（iOS）

版权所有：腾讯云计算（北京）有限责任公司

@deprecated v8.6 版本开始不推荐使用，建议使用 startScreenCaptureInApp 接口替代之。

setVideoEncoderRotation

setVideoEncoderRotation

void setVideoEncoderRotation	(TRTCVideoRotation rotation)

设置视频编码器输出的画面方向

@deprecated v11.7 版本开始不推荐使用。

setVideoEncoderMirror

setVideoEncoderMirror

void setVideoEncoderMirror	(bool mirror)

设置编码器输出的画面镜像模式

@deprecated v11.7 版本开始不推荐使用。

版权所有：腾讯云计算（北京）有限责任公司

错误码表

最近更新时间：2024-06-06 15:26:14

Copyright (c) 2021 Tencent. All rights reserved.

Module: TRTC 错误码表

Function: 用于通知客户 TRTC 在使用过程中出现的警告和错误

错误码表

枚举类型

枚举类型	描述
TXLiteAVError	错误码
TXLiteAVWarning	警告码

TXLiteAVError

TXLiteAVError

错误码

枚举	取值	描述
ERR_NULL	0	无错误
ERR_FAILED	-1	暂未归类的通用错误
ERR_INVALID_PARAMETER	-2	调用 API 时，传入的参数不合法
ERR_REFUSED	-3	API 调用被拒绝
ERR_NOT_SUPPORTED	-4	当前 API 不支持调用
ERR_INVALID_LICENSE	-5	license 不合法，调用失败

ERR_REQUEST_SERVER_TIMEOUT	-6	请求服务器超时
ERR_SERVER_PROCESS_FAILED	-7	服务器无法处理您的请求
ERR_DISCONNECTED	-8	断开连接
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_SCREEN_CAPTURE_START_FAIL	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了，如果在 Windows 或 Mac 系统的设备出现，请检查录屏接口的参数是否符合要求
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	录屏失败，在 Android 平台，需要5.0以上的系统，在 iOS 平台，需要11.0以上的系统
ERR_SCREEN_CAPTURE_STOPPED	-7001	录屏被系统中止
ERR_SCREEN_SHARE_NOT_AUTHORIZED	-102015	没有权限上行辅路
ERR_SCREEN_SHRAE_OCCUPIED_BY_OTHER	-102016	其他用户正在上行辅路
ERR_VIDEO_ENCODE_FAIL	-1303	视频帧编码失败，例如 iOS 设备切换到其他应用时，硬编码器可能被系统释放，再切换回来时，硬编码器重启前，可能会抛出
ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率

版权所有：腾讯云计算（北京）有限责任公司

ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	自定视频采集：设置的 pixel format 不支持
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	自定视频采集：设置的 buffer type 不支持
ERR_NO_AVAILABLE_HEVC_DECODERS	-2304	找不到可用的 HEVC 解码器
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败，例如在 Windows 或 Mac 设备，扬声器的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败
ERR_AUDIO_PLUGIN_START_FAIL	-1330	开启系统声音录制失败，例如音频驱动插件不可用
ERR_AUDIO_PLUGIN_INSTALL_NOT_AUTHORIZED	-1331	安装音频驱动插件未授权
ERR_AUDIO_PLUGIN_INSTALL_FAILED	-1332	安装音频驱动插件失败
ERR_AUDIO_PLUGIN_INSTALLED_BUT_NEED_RESTART	-1333	安装虚拟声卡插件成功，但首次安装后功能暂时不可用，此为 Mac 系统限制，请

版权所有：腾讯云计算（北京）有限责任公司

		在收到此错误码后提示用户重启当前 APP
ERR_AUDIO_ENCODE_FAIL	-1304	音频帧编码失败，例如传入自定义音频数据，SDK 无法处理
ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率
ERR_TRTC_ENTER_ROOM_FAILED	-3301	进入房间失败，请查看 onError 中的 -3301 对应的 msg 提示确认失败原因
ERR_TRTC_REQUEST_IP_TIMEOUT	-3307	请求 IP 和 sig 超时，请检查网络是否正常，或网络防火墙是否放行 UDP。 可尝试访问下列 IP： 162.14.22.165:8000 162.14.6.105:8000 和域名： default-query.trtc.tencent-cloud.com:8000
ERR_TRTC_CONNECT_SERVER_TIMEOUT	-3308	请求进房超时，请检查是否断网或者是否开启vpn，您也可以切换4G进行测试确认
ERR_TRTC_ROOM_PARAM_NULL	-3316	进房参数为空，请检查：enterRoom:appScene: 接口调用是否传入有效的 param
ERR_TRTC_INVALID_SDK_APPID	-3317	进房参数 sdkAppId 错误，请检查 TRTCParams.sdkAppId 是否为空
ERR_TRTC_INVALID_ROOM_ID	-3318	进房参数 roomId 错误，请检查 TRTCParams.roomId 或 TRTCParams.strRoomId 是否为空，注意 roomId 和 strRoomId 不可混用
ERR_TRTC_INVALID_USER_ID	-3319	进房参数 userId 不正确，请检查 TRTCParams.userId 是否为空
ERR_TRTC_INVALID_USER_SIG	-3320	进房参数 userSig 不正确，请检查 TRTCParams.userSig 是否为空

ERR_TRTC_ENTER_ROOM_REFUSED	-3340	进房请求被拒绝，请检查是否连续调用 enterRoom 进入相同 Id 的房间
ERR_TRTC_INVALID_PRIVATE_MAPKEY	-100006	您开启了高级权限控制，但参数 TRTCParams.privateMapKey 校验失败， 您可参考 高级权限控制 进行检查
ERR_TRTC_SERVICE_SUSPENDED	-100013	服务不可用。请检查：套餐包剩余分钟数是否大于0，腾讯云账号是否欠费。 您可参考 **套餐包管理** 进行查看与配置
ERR_TRTC_USER_SIG_CHECK_FAILED	-100018	UserSig 校验失败，请检查参数 TRTCParams.userSig 是否填写正确，或是否已经过期。 您可参考 **UserSig 生成与校验** 进行校验
ERR_TRTC_PUSH_THIRD_PARTY_CLOUD_TIMEOUT	-3321	旁路转推请求超时
ERR_TRTC_MIX_TRANSCODING_TIMEOUT	-3322	云端混流请求超时
ERR_TRTC_PUSH_THIRD_PARTY_CLOUD_FAILED	-3323	旁路转推回包异常
ERR_TRTC_MIX_TRANSCODING_FAILED	-3324	云端混流回包异常
ERR_TRTC_START_PUBLISHING_TIMEOUT	-3333	开始向腾讯云的直播 CDN 推流信令超时
ERR_TRTC_START_PUBLISHING_FAILED	-3334	开始向腾讯云的直播 CDN 推流信令异常
ERR_TRTC_STOP_PUBLISHING_TIMEOUT	-3335	停止向腾讯云的直播 CDN 推流信令超时
ERR_TRTC_STOP_PUBLISHING_FAILED	-3336	停止向腾讯云的直播 CDN 推流信令异常
ERR_TRTC_CONNECT_OTHER_ROOM_TIMEOUT	-3326	请求连麦超时
ERR_TRTC_DISCONNECT_OTHER_ROOM_TIMEOUT	-3327	请求退出连麦超时

版权所有：腾讯云计算（北京）有限责任公司

ERR_TRTC_CONNECT_OTHER_ROOM_INVALID_PARAMETER	-3328	无效参数
ERR_TRTC_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	当前是观众角色，不能请求或断开跨房连麦，需要先 `switchRole` 到主播
ERR_BGM_OPEN_FAILED	-4001	打开文件失败，如音频数据无效，FFMPEG 协议未找到等
ERR_BGM_DECODE_FAILED	-4002	音频文件解码失败
ERR_BGM_OVER_LIMIT	-4003	数量超过限定值，如同时预加载两首背景音乐
ERR_BGM_INVALID_OPERATION	-4004	无效操作，如开始播放后再调用预加载操作
ERR_BGM_INVALID_PATH	-4005	非法路径导致打开文件失败，请检查您传入的路径参数是否指向一个合法的音乐文件
ERR_BGM_INVALID_URL	-4006	非法URL导致打开文件失败，请用浏览器检查您传入的 URL 地址是否可以下载到期望的音乐文件
ERR_BGM_NO_AUDIO_STREAM	-4007	无音频流导致打开文件失败，请确认您传入的文件是否是合法的音频文件，以及文件是否被损坏
ERR_BGM_FORMAT_NOT_SUPPORTED	-4008	格式不支持导致打开文件失败，请确认您传入的文件格式是否是支持的文件格式，移动端支持【mp3，aac，m4a，wav，ogg，mp4，mkv】，桌面端支持【mp3，aac，m4a，wav，mp4，mkv】

TXLiteAVWarning

TXLiteAVWarning

警告码

枚举	取值	描述
WARNING_HW_ENCODER_START_FAIL	1103	硬编码启动出现问题，自动切换到软编码
WARNING_CURRENT_ENCODE_TYPE_CHANGED	1104	表示编码器发生改变，可以通过 onWarning 函数的扩展信息中的相关字段来获取当前的编码格式和类型。 type 字段值为 0 代表 H.264 编码，1 代表 H.265 编码 hardware 字段值为 0 代表软件编码，1 代表硬件编码 stream 字段值为 0 代表大流，1 代表小流，2 代表辅流
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	当前 CPU 使用率太高，无法满足软件编码需求，自动切换到硬件编码
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	摄像头采集帧率不足，部分自带美颜算法的 Android 手机上会出现
WARNING_SW_ENCODER_START_FAIL	1109	软编码启动失败
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	摄像头采集分辨率被降低，以满足当前帧率和性能最优解。
WARNING_CAMERA_DEVICE_EMPTY	1111	没有检测到可用的摄像头设备
WARNING_CAMERA_NOT_AUTHORIZED	1112	用户未授权当前应用使用摄像头
WARNING_OUT_OF_MEMORY	1113	内存不足，部分功能

版权所有：腾讯云计算（北京）有限责任公司

		可能不正常。
WARNING_CAMERA_IS_OCCUPIED	1114	摄像头被占用.
WARNING_CAMERA_DEVICE_ERROR	1115	摄像头设备异常.
WARNING_CAMERA_DISCONNECTED	1116	摄像头无法连接.
WARNING_CAMERA_START_FAILED	1117	摄像头启动失败.
WARNING_CAMERA_SERVER_DIED	1118	系统异常.
WARNING_SCREEN_CAPTURE_NOT_AUTHORIZED	1206	用户未授权当前应用使用屏幕录制
WARNING_CURRENT_DECODE_TYPE_CHANGED	2008	表示解码器发生改变，可以通过 onWarning 函数的扩展信息中的 type 字段来获取当前的解码格式。 其中 1 代表 265 解码，0 代表 264 解码。注意 Windows 端不支持此错误码的扩展信息。
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	当前视频帧解码失败
WARNING_HW_DECODER_START_FAIL	2106	硬解启动失败，采用软解码
WARNING_VIDEO_DECODER_HW_TO_SW	2108	当前流硬解第一个 I 帧失败，SDK 自动切软解
WARNING_SW_DECODER_START_FAIL	2109	软解码器启动失败
WARNING_VIDEO_RENDER_FAIL	2110	视频渲染失败
WARNING_VIRTUAL_BACKGROUND_DEVICE_UNSURPORTED	8001	虚拟背景设备不支持
WARNING_VIRTUAL_BACKGROUND_NOT_AUTHORIZED	8002	虚拟背景未授权
WARNING_VIRTUAL_BACKGROUND_INVALID_PARAMETER	8003	虚拟背景参数异常
WARNING_VIRTUAL_BACKGROUND_PERFORMANCE_INSUFFICIENT	8004	虚拟背景性能不足
WARNING_MICROPHONE_DEVICE_EMPTY	1201	没有检测到可用的麦

版权所有：腾讯云计算（北京）有限责任公司

		克风设备
WARNING_SPEAKER_DEVICE_EMPTY	1202	没有检测到可用的扬声器设备
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	用户未授权当前应用使用麦克风
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	音频采集设备不可用（例如被占用或者PC判定无效设备）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	音频播放设备不可用（例如被占用或者PC判定无效设备）
WARNING_BLUETOOTH_DEVICE_CONNECT_FAIL	1207	蓝牙设备连接失败（例如其他应用通过设置通话音量占用音频通道）
WARNING_MICROPHONE_IS_OCCUPIED	1208	音频采集设备被占用
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	当前音频帧解码失败
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	音频录制写入文件失败
WARNING_MICROPHONE_HOWLING_DETECTED	7002	录制音频时监测到啸叫
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	当前是观众角色，不支持发布音视频，需要先切换成主播角色
WARNING_UPSTREAM_AUDIO_AND_VIDEO_OUT_OF_SYNC	6006	音视频发送时间戳异常，可能引起音画不同步问题

版权所有：腾讯云计算（北京）有限责任公司

Web
API 概览

最近更新时间：2024-05-29 15:21:54

API 介绍

TRTC

注意：

本文适用于 5.x.x 版本的 TRTC Web SDK。

TRTC 对象通过调用 TRTC.create() 方法创建，提供实时音视频的核心能力：

进入一个音视频房间，调用 `trtc.enterRoom()` 方法。

退出当前音视频房间，调用 `trtc.exitRoom()` 方法。

预览/发布本地视频，调用 `trtc.startLocalVideo()` 方法。

采集/发布本地音频，调用 `trtc.startLocalAudio()` 方法。

取消预览/发布本地视频，调用 `trtc.stopLocalVideo()` 方法。

取消采集/发布本地音频，调用 `trtc.stopLocalAudio()` 方法。

观看远端视频，调用 `trtc.startRemoteVideo()` 方法。

取消观看远端视频，调用 `trtc.stopRemoteVideo()` 方法。

静默/取消静默远端音频，调用 `trtc.muteRemoteAudio()` 方法。

静态方法

API	描述
create	创建一个 TRTC 对象，用于实现进房、预览、推流、拉流等功能。
setLogLevel	设置日志输出等级建议在开发测试阶段设置为 DEBUG 等级，该日志等级包含详细的提示信息。默认输出 INFO 日志等级，该日志等级包含 SDK 主要功能的日志信息。
isSupported	检测 TRTC Web SDK 是否支持当前浏览器
getCameraList	返回摄像头设备列表
getMicrophoneList	返回麦克风设备列表
getSpeakerList	返回扬声器设备列表，出于安全的考虑，在用户未授权摄像头或麦克风访问权限

版权所有：腾讯云计算（北京）有限责任公司

	前，label 及 deviceId 字段可能都是空的。因此建议在用户授权访问后再调用该接口获取设备详情。
setCurrentSpeaker	设置当前音频播放的扬声器

成员方法

创建实例后调用。

API	描述
enterRoom	进入一个音视频通话房间（以下简称"进房"）。
exitRoom	退出当前音视频通话房间。
switchRole	切换用户角色，仅在 TRTC.TYPE.SCENE_LIVE 互动直播模式下生效。
destroy	销毁 TRTC 实例
startLocalAudio	开启本地麦克风采集，并发布到当前的房间中。
updateLocalAudio	更新本地麦克风配置。
stopLocalAudio	停止本地麦克风的采集及发布。
startLocalVideo	开启本地摄像头采集，在您指定的 HTMLElement 标签下播放摄像头画面，并将摄像头画面发布到当前所在房间中。
updateLocalVideo	更新本地摄像头配置。
stopLocalVideo	停止本地摄像头的采集、预览及发布。
startScreenShare	开启屏幕分享。
updateScreenShare	更新屏幕分享配置
stopScreenShare	停止屏幕分享。
startRemoteVideo	播放远端视频
updateRemoteVideo	更新远端视频播放配置
stopRemoteVideo	用于停止远端视频播放。
muteRemoteAudio	静音某个远端用户，并且不再拉取该用户的音频数据。仅对当前用户有效，房间内的其他用户依然可以听到被静音用户的声音。

版权所有：腾讯云计算（北京）有限责任公司

setRemoteAudioVolume	用于控制远端音频的播放音量。
enableAudioVolumeEvaluation	开启或关闭音量大小回调
on	监听 TRTC 事件详细事件列表请参见：TRTC.EVENT
off	取消事件监听
getVideoSnapshot	获取视频截图
getVideoTrack	获取视频轨道
getAudioTrack	获取音频轨道
sendSEIMessage	发送 SEI 消息
sendCustomMessage	发送自定义消息
startPlugin	开启插件
updatePlugin	更新插件
stopPlugin	停止插件

教程

您可以参考如下教程实现各种功能：

功能	Sample Code 指引
基础音视频通话	指引链接
实现互动直播连麦	指引链接
切换摄像头和麦克风	指引链接
设置本地视频属性	指引链接
开关摄像头、麦克风	指引链接
屏幕分享	指引链接
音量大小检测	指引链接
自定义采集与自定义播放渲染	指引链接
房间内上行用户个数限制	指引链接

版权所有：腾讯云计算（北京）有限责任公司

通话前环境与设备检测	指引链接
通话前的网络质量检测	指引链接
设备插拔检测及采集异常处理	指引链接
开启水印	指引链接

最佳实践

最佳实践	指引链接
多人视频通话	指引链接
自动播放处理建议	指引链接
应对防火墙受限	指引链接

说明：
常见问题参见 Web 端相关。

支持的平台

TRTC Web SDK 基于 WebRTC 实现，目前支持桌面端和移动端的主流浏览器，详细支持度参见下方表格。理论上来说， SDK 支持所有基于 Chromium 56+ 版本的浏览器。

如果您的应用场景不在支持的表格里，可以使用 TRTC.isSupported() 接口 或者打开 TRTC Web SDK 能力检测页面 检测当前环境是否支持 WebRTC 所有能力。

操作系统	浏览器类型	浏览器最低版本要求	接收（播放）	发送（推流）	屏幕分享
Windows	Chrome	56+	支持	支持	Chrome 72+
	Firefox	56+	支持	支持	Firefox 66+
	Edge	80+	支持	支持	支持
	Opera	46+	支持	支持	Opera 60+

MacOS	Chrome	56+	支持	支持	Chrome 72+
	Safari	11+	支持	支持	Safari 13+
	Edge	80+	支持	支持	支持
	Firefox	56+	支持	支持	Firefox 66+
Android	Chrome	69+	支持	支持	不支持
	Edge	80+	支持	支持	不支持
	Firefox	56+	支持	支持	不支持
	Opera	46+	支持	支持	不支持
iOS	Chrome	iOS 11+	支持	iOS 14.3+	不支持
	Safari	iOS 11+	支持	支持	不支持
	Edge	iOS 11+	支持	iOS 14.3+	不支持
	Firefox	iOS 11+	支持	iOS 14.3+	不支持

注意

1. 由于 H.264 版权限制，华为 Chrome 88 以下版本以及部分厂商的 Android 设备，不具备 H264 编码能力（即无法推流）。如果您希望在这些设备的浏览器中，使用 TRTC Web SDK 推流，请 联系我们 开通 VP8 编解码能力。
2. Mac OS 下的
Firefox
屏幕分享效果比较差且暂无解决方案，建议使用 Chrome 或者 Safari 进行屏幕分享。
3. 建议您及时将 TRTC Web SDK 更新至最新版本，以便获得更好的产品稳定性及在线支持。版本升级注意事项请参见：升级指引。

URL 域名协议限制

由于浏览器安全策略的限制，使用 WebRTC 能力对页面的访问协议有严格的要求，请参照以下表格进行开发和部署应用。

应用场景	协议	接收（播放）	发送（上麦）	屏幕分享	备注
生产环境	HTTPS 协议	支持	支持	支持	**推荐**
生产环境	HTTP 协议	支持	不支持	不支持	-

版权所有：腾讯云计算（北京）有限责任公司

本地开发环境	http://localhost	支持	支持	支持	**推荐**
本地开发环境	http://127.0.0.1	支持	支持	支持	-
本地开发环境	http://[本机IP]	支持	不支持	不支持	-
本地开发环境	file:///	支持	支持	支持	-

联系我们

在 telegram 或 github 联系我们。

错误码

最近更新时间：2024-03-26 10:55:04

注意：

本文适用于 5.x.x 版本的 TRTC Web SDK。

TRTC SDK v5.0 定义了 8 种错误码类型，通过 RtcError 对象来获取 RtcError.code 并做相应的处理。

错误码定义

Key	Code	描述
INVALID_PARAMETER	5000	说明：调用接口时传入了不满足 API 要求的参数 处理建议：请检查传入参数是否符合 API 的规范，例如参数类型是否正确。
INVALID_OPERATION	5100	说明：调用接口时，不满足 API 的前提要求。 处理建议：请根据对应 API 文档检查调用逻辑是否符合 API 的前提要求。例如：1.未进房成功就进行切换角色，2.播放的远端用户和流不存在。
ENV_NOT_SUPPORTED	5200	说明：当前环境不支持该功能，表明当前浏览器不支持调用对应 API 处理建议：通常使用 TRTC.isSupported 可感知当前浏览器支持哪些能力。如果浏览器不支持，需要引导用户使用支持该能力的浏览器，参考：检测浏览器支持性
DEVICE_ERROR	5300	说明：获取设备或者采集音视频出现异常 以下接口出现异常时会抛出该错误码：startLocalVideo、updateLocalVideo、startLocalAudio、updateLocalAudio、startScreenShare、updateScreenShare 处理建议：引导用户检查设备是否有摄像头及麦克风、系统是否给浏览器授权以及浏览器是否给页面授权。建议增加进房前的设备检测流程，确认麦克风和摄像头是否存在，并且能正常采集，再进行下一步通话操作。通常经过设备检查后都能避免该异常。 实现方式请参考：通话前环境与设备检测 如果需要区分更详细的异常类别，可以参考 extraCode
SERVER_ERROR	5400	说明：收到服务端返回的异常数据时抛出该错误码 以下接口出现异常时会抛出该错误码：enterRoom、startLocalVideo、startLocalAudio、startScreenShare、startRemoteVideo、switchRole 处理建议：服务端异常通常在开发阶段处理，常见的异常有：传入的 userSig 过期，腾讯云账号欠费，未开通TRTC服务等，服务端返回异常数据有以下原因。

OPERATION_FAILED	5500	说明：在满足 API 调用要求的情况下，SDK 经过多次重试仍然无法解决的异常，通常是由于浏览器、网络的问题造成。 以下接口出现异常时会抛出该错误码：enterRoom、startLocalVideo、startLocalAudio、startScreenShare、startRemoteVideo、switchRole 处理建议： 确认通信必需的域名和端口是否满足您的网络环境要求，参考文档应对防火墙限制及设置代理 其他问题需要联系工程师处理，在 Telegram 联系我们
OPERATION_ABORT	5998	说明：中止 API 执行时抛出该错误码。在不满足 API 生命周期的调用或重复调用时 API 会中止执行，避免无意义的操作。 例如：连续调用 enterRoom，startLocalXxx等接口，在没有进房就调用退房。 以下接口出现异常时会抛出该错误码：enterRoom、startLocalVideo、startLocalAudio、startScreenShare、startRemoteVideo、switchRole 处理建议：捕获并识别该错误码，然后在业务逻辑规避不必要的调用，或者也可以不做任何处理，因为 SDK 做了无副作用处理，您只需在 catch 时识别该错误码并忽略。
UNKNOWN_ERROR	5999	说明：未知错误或者未被定义的错误 处理建议：联系工程师处理在 Telegram 联系我们

Electron
API 概览

最近更新时间：2022-08-08 15:25:08

TRTCCloud @ TXLiteAVSDK

腾讯云视频通话功能的主要接口类。

- **主要文档地址**：TRTC Electron SDK
- **示例代码地址**：TRTC Electron Demo

创建 TRTC 对象

```
const TRTCCloud = require('trtc-electron-sdk').default;
// import TRTCCloud from 'trtc-electron-sdk';
this.rtcCloud = new TRTCCloud();
```

从v7.9.348起，TRTC Electron SDK 增加了 trtc.d.ts 文件，方便使用 typescript 的开发者

```
import TRTCCloud from 'trtc-electron-sdk';
const rtcCloud: TRTCCloud = new TRTCCloud();
// 获取 SDK 版本号
rtcCloud.getSDKVersion();
```

设置回调

```
subscribeEvents = (rtcCloud) => {
rtcCloud.on('onError', (errcode, errmsg) => {
console.info('trtc_demo: onError :' + errcode + " msg" + errmsg);
});
rtcCloud.on('onEnterRoom', (elapsed) => {
console.info('trtc_demo: onEnterRoom elapsed:' + elapsed);
});
rtcCloud.on('onExitRoom', (reason) => {
console.info('onExitRoom: userenter reason:' + reason);
});
};

subscribeEvents(this.rtcCloud);
```

版权所有：腾讯云计算（北京）有限责任公司

创建与销毁 TRTCCloud 单例

API	描述
getTRTCShareInstance	用于动态加载 dll 时，创建 TRTCCloud 对象单例。
destroyTRTCShareInstance	释放 TRTCCloud 单例对象并清理资源。

房间相关接口函数

API	描述
enterRoom	进入房间，若房间不存在，系统将自动创建一个新房间。
exitRoom	退出房间。
switchRoom	切换房间。
switchRole	切换角色，仅适用于直播场景（TRTCAppSceneLIVE 和 TRTCAppSceneVoiceChatRoom）。
connectOtherRoom	请求跨房连麦（主播跨房 PK）。
disconnectOtherRoom	关闭跨房连麦（主播跨房 PK）。
setDefaultStreamRecvMode	设置音视频数据接收模式（需要在进房前设置才能生效）。

CDN 相关接口函数

API	描述
startPublishing	开始向腾讯云的直播 CDN 推流。
stopPublishing	停止向腾讯云的直播 CDN 推流。
startPublishCDNStream	开始向非腾讯云的直播 CDN 转推。
stopPublishCDNStream	停止向非腾讯云的直播 CDN 推流。
setMixTranscodingConfig	设置云端的混流转码参数。

视频相关接口函数

API	描述
startLocalPreview	启动本地摄像头采集和预览。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
stopLocalPreview	停止本地摄像头采集和预览。
muteLocalVideo	是否屏蔽自己的视频画面。
startRemoteView	开始显示远端视频画面。
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流。
stopAllRemoteView	停止显示所有远端视频画面，同时不再拉取远端用户的视频数据流。
muteRemoteVideoStream	暂停接收指定的远端视频流。
muteAllRemoteVideoStreams	停止接收所有远端视频流。
setVideoEncoderParam	设置视频编码器相关参数。
setNetworkQosParam	设置网络流控相关参数。
setLocalRenderParams	设置本地图像（主流）的渲染参数。
setLocalViewFillMode	废弃接口：设置本地图像的渲染模式。
setRemoteRenderParams	设置远端图像的渲染参数。
setRemoteViewFillMode	废弃接口：设置远端图像的渲染模式。
setLocalViewRotation	废弃接口：设置本地图像的顺时针旋转角度。
setRemoteViewRotation	废弃接口：设置远端图像的顺时针旋转角度。
setVideoEncoderRotation	设置视频编码输出的（即远端用户观看到的以及服务器录制下来的）画面方向。
setLocalViewMirror	废弃接口：设置本地摄像头预览画面的镜像模式。
setVideoEncoderMirror	设置编码器输出的画面镜像模式。
enableSmallVideoStream	开启大小画面双路编码模式。
setRemoteVideoStreamType	选定观看指定 userId 的大画面或小画面。
setPriorRemoteVideoStreamType	废弃接口：设定观看方优先选择的视频质量。
snapshotVideo	视频画面截图。

音频相关接口函数

版权所有：腾讯云计算（北京）有限责任公司

API	描述
startLocalAudio	开启本地音频的采集和上行。
stopLocalAudio	关闭本地音频的采集和上行。
muteLocalAudio	静音本地的音频。
muteRemoteAudio	静音某一个用户的声音，同时不再拉取该远端用户的音频数据流。
muteAllRemoteAudio	静音所有用户的声音，同时不再拉取远端用户的音频数据流。
setAudioCaptureVolume	设置 SDK 采集音量。
getAudioCaptureVolume	获取 SDK 采集音量。
setAudioPlayoutVolume	设置 SDK 播放音量。
getAudioPlayoutVolume	获取 SDK 播放音量。
enableAudioVolumeEvaluation	启用或关闭音量大小提示。
startAudioRecording	开始录音。
stopAudioRecording	停止录音。
setAudioQuality	废弃接口：设置音频质量。
setRemoteAudioVolume	设置远程用户播放音量。

摄像头相关接口函数

API	描述
getCameraDevicesList	获取摄像头设备列表。
setCurrentCameraDevice	设置要使用的摄像头。
getCurrentCameraDevice	获取当前使用的摄像头。

音频设备相关接口函数

API	描述
getMicDevicesList	获取麦克风设备列表。
getCurrentMicDevice	获取当前选择的麦克风。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setCurrentMicDevice	设置要使用的麦克风。
getCurrentMicDeviceVolume	获取系统当前麦克风设备音量。
setCurrentMicDeviceVolume	设置系统当前麦克风设备的音量。
setCurrentMicDeviceMute	设置系统当前麦克风设备的静音状态。
getCurrentMicDeviceMute	获取系统当前麦克风设备是否静音。
getSpeakerDevicesList	获取扬声器设备列表。
getCurrentSpeakerDevice	获取当前的扬声器设备。
setCurrentSpeakerDevice	设置要使用的扬声器。
getCurrentSpeakerVolume	获取系统当前扬声器设备音量。
setCurrentSpeakerVolume	设置系统当前扬声器设备音量。
setCurrentSpeakerDeviceMute	设置系统当前扬声器设备的静音状态。
getCurrentSpeakerDeviceMute	获取系统当前扬声器设备是否静音。

美颜相关接口函数

API	描述
setBeautyStyle	设置美颜、美白以红润效果级别。
setWaterMark	设置水印。

辅流相关接口函数

API	描述
startRemoteSubStreamView	废弃接口：开始渲染远端用户的辅流（屏幕分享）画面。
stopRemoteSubStreamView	废弃接口：停止渲染远端用户的辅流（屏幕分享）画面。
setRemoteSubStreamViewFillMode	废弃接口：设置辅流（屏幕分享）画面的渲染模式。
setRemoteSubStreamViewRotation	废弃接口：设置辅流（屏幕分享）画面的顺时针旋转角度。
getScreenCaptureSources	枚举可共享的窗口列表。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
selectScreenCaptureTarget	设置屏幕共享参数，该方法在屏幕共享过程中也可以调用。
startScreenCapture	启动屏幕分享。
pauseScreenCapture	暂停屏幕分享。
resumeScreenCapture	恢复屏幕分享。
stopScreenCapture	停止屏幕分享。
setSubStreamEncoderParam	设置辅流（屏幕分享）的编码器参数。
setSubStreamMixVolume	设置辅流（屏幕分享）的混音音量大小。
addExcludedShareWindow	将指定窗口加入屏幕分享的排除列表中，加入排除列表中的窗口不会被分享出去。
removeExcludedShareWindow	将指定窗口从屏幕分享的排除列表中移除。
removeAllExcludedShareWindow	将所有窗口从屏幕分享的排除列表中移除。

自定义消息发送

API	描述
sendCustomCmdMsg	发送自定义消息给房间内所有用户。
sendSEIMsg	将小数据量的自定义数据嵌入视频帧中。

背景混音相关接口函数

API	描述
playBGM	废弃接口：启动播放背景音乐。
stopBGM	废弃接口：停止播放背景音乐。
pauseBGM	废弃接口：暂停播放背景音乐。
resumeBGM	废弃接口：继续播放背景音乐。
getBGMDuration	废弃接口：获取背景音乐文件总时长，单位毫秒。
setBGMPosition	废弃接口：设置背景音乐播放进度。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setBGMVolume	废弃接口：设置背景音乐播放音量的大小。
setBGMPlayoutVolume	废弃接口：设置背景音乐本地播放音量的大小。
setBGMPublishVolume	废弃接口：设置背景音乐远端播放音量的大小。
startSystemAudioLoopback	打开系统声音采集。
stopSystemAudioLoopback	关闭系统声音采集。
setSystemAudioLoopbackVolume	设置系统声音采集的音量。
startPlayMusic	启动播放背景音乐。
stopPlayMusic	停止播放背景音乐。
pausePlayMusic	暂停播放背景音乐。
resumePlayMusic	恢复播放背景音乐。
getMusicDurationInMS	获取背景音乐文件总时长，单位毫秒。
seekMusicToPosInTime	设置背景音乐播放进度。
setAllMusicVolume	设置背景音乐的音量大小，播放背景音乐混音时使用，用来控制背景音音量大小。
setMusicPlayoutVolume	设置背景音乐本地播放音量的大小。
setMusicPublishVolume	设置背景音乐远端播放音量的大小。

音效相关接口函数

API	描述
playAudioEffect	废弃接口：播放音效。
setAudioEffectVolume	废弃接口：设置音效音量。
stopAudioEffect	废弃接口：停止音效。
stopAllAudioEffects	废弃接口：停止所有音效。
setAllAudioEffectsVolume	废弃接口：设置所有音效的音量。
pauseAudioEffect	废弃接口：暂停音效。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
resumeAudioEffect	废弃接口：恢复音效。

设备和网络测试

API	描述
startSpeedTest	开始进行网络测速（视频通话期间请勿测试，以免影响通话质量）。
stopSpeedTest	停止网络测速。
startCameraDeviceTest	开始进行摄像头测试。
stopCameraDeviceTest	停止摄像头测试。
startMicDeviceTest	开始进行麦克风测试。
stopMicDeviceTest	停止麦克风测试。
startSpeakerDeviceTest	开始进行扬声器测试。
stopSpeakerDeviceTest	停止扬声器测试。

LOG 相关接口函数

API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setConsoleEnabled	启用或禁用控制台日志打印。
setLogCompressEnabled	启用或禁用 Log 的本地压缩。
setLogDirPath	设置日志保存路径。
setLogCallback	设置日志回调。
callExperimentalAPI	调用实验性 API 接口。

弃用接口函数

API	描述

版权所有：腾讯云计算（北京）有限责任公司

API	描述
setMicVolumeOnMixing	从 v6.9 版本开始废弃。

TRTCCallback @ TXLiteAVSDK

腾讯云视频通话功能的回调接口类。

错误事件和警告事件

API	描述
onError	错误回调：SDK 不可恢复的错误，必须监听，并分情况给用户适当的界面提示。
onWarning	警告回调：用于告知您一些非严重性问题，例如出现了卡顿或可恢复的解码失败。

房间事件回调

API	描述
onEnterRoom	已加入房间的回调。
onExitRoom	退出房间的事件回调。
onSwitchRole	切换角色的事件回调。
onConnectOtherRoom	请求跨房连麦（主播跨房 PK）的结果回调。
onDisconnectOtherRoom	关闭跨房连麦（主播跨房 PK）的结果回调。
onSwitchRoom	切换房间。

成员事件回调

API	描述
onRemoteUserEnterRoom	有用户加入当前房间。
onRemoteUserLeaveRoom	有用户离开当前房间。
onUserVideoAvailable	用户是否开启摄像头视频。
onUserSubStreamAvailable	用户是否开启屏幕分享。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onUserAudioAvailable	用户是否开启音频上行。
onFirstVideoFrame	开始渲染本地或远程用户的首帧画面。
onFirstAudioFrame	开始播放远程用户的首帧音频（本地声音暂不通知）。
onSendFirstLocalVideoFrame	首帧本地视频数据已经被送出。
onSendFirstLocalAudioFrame	首帧本地音频数据已经被送出。
onUserEnter	废弃接口：有主播加入当前房间。
onUserExit	废弃接口： 有主播离开当前房间。

统计和质量回调

API	描述
onNetworkQuality	网络质量：该回调每2秒触发一次，统计当前网络的上行和下行质量。
onStatistics	技术指标统计回调。

服务器事件回调

API	描述
onConnectionLost	SDK 与服务器的连接断开。
onTryToReconnect	SDK 尝试重新连接到服务器。
onConnectionRecovery	SDK 与服务器的连接恢复。
onSpeedTest	废弃接口：服务器测速的回调，SDK 对多个服务器 IP 进行测速，每个 IP 的测速结果通过这个回调通知。
onSpeedTestResult	网速测试的结果回调。

硬件设备事件回调

API	描述
onCameraDidReady	摄像头准备就绪。
onMicDidReady	麦克风准备就绪。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onUserVoiceVolume	用于提示音量大小的回调,包括每个 userId 的音量和远端总音量；本地用户 userid 为''。
onDeviceChange	本地设备通断回调。
onTestMicVolume	麦克风测试音量回调。
onTestSpeakerVolume	扬声器测试音量回调。
onAudioDeviceCaptureVolumeChanged	当前音频采集设备音量变化回调。
onAudioDevicePlayoutVolumeChanged	当前音频播放设备音量变化回调。

自定义消息的接收回调

API	描述
onRecvCustomCmdMsg	收到自定义消息回调。
onMissCustomCmdMsg	自定义消息丢失回调。
onRecvSEIMsg	收到 SEI 消息的回调。

CDN 旁路转推回调

API	描述
onStartPublishing	开始向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 startPublishing() 接口。
onStopPublishing	停止向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 stopPublishing() 接口。
onStartPublishCDNStream	启动旁路推流到 CDN 完成的回调。
onStopPublishCDNStream	停止旁路推流到 CDN 完成的回调。
onSetMixTranscodingConfig	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 setMixTranscodingConfig() 接口。

系统音量采集回调

API	描述

API	描述
onSystemAudioLoopbackError	系统音量采集状态的回调（仅在 Mac 上有效）。

音效回调

API	描述
onAudioEffectFinished	废弃接口：播放音效结束回调。

屏幕分享回调

API	描述
onScreenCaptureCovered	当屏幕分享窗口被遮挡无法正常捕获时，SDK 会通过此回调通知，可在此回调里通知用户移开遮挡窗口。
onScreenCaptureStarted	当屏幕分享开始时，SDK 会通过此回调通知。
onScreenCapturePaused	当屏幕分享暂停时，SDK 会通过此回调通知。
onScreenCaptureResumed	当屏幕分享恢复时，SDK 会通过此回调通知。
onScreenCaptureStopped	当屏幕分享停止时，SDK 会通过此回调通知。

截图回调

API	描述
onSnapshotComplete	截图完成时，SDK 会通过此回调通知。

背景混音事件回调

API	描述
onPlayBGMBegin	废弃接口：开始播放背景音乐。
onPlayBGMProgress	废弃接口：播放背景音乐的进度。
onPlayBGMComplete	废弃接口：播放背景音乐结束。

关键类型定义

关键类型

类名	描述
TRTCParams	进房相关参数。
TRTCVideoEncParam	视频编码参数。
TRTCNetworkQosParam	网络流控相关参数。
TRTCQualityInfo	视频质量。
TRTCVolumeInfo	音量大小。
TRTCSpeedTestResult	网络测速结果。
TRTCMixUser	云端混流中每一路子画面的位置信息。
TRTCTranscodingConfig	云端混流（转码）配置。
TRTCPublishCDNParam	CDN 旁路推流参数。
TRTCAudioRecordingParams	录音参数。
TRTCLocalStatistics	自己本地的音视频统计信息。
TRTCRemoteStatistics	远端成员的音视频统计信息。
TRTCStatistics	统计数据。

枚举值

枚举	描述
TRTCVideoResolution	视频分辨率。
TRTCVideoResolutionMode	视频分辨率模式。
TRTCVideoStreamType	视频流类型。
TRTCQuality	画质级别。
TRTCVideoFillMode	视频画面填充模式。
TRTCBeautyStyle	美颜（磨皮）算法。
TRTCAppScene	应用场景。

版权所有：腾讯云计算（北京）有限责任公司

枚举	描述
TRTCRoleType	角色，仅适用于直播场景（TRTCAppSceneLIVE）。
TRTCQosControlMode	流控模式。
TRTCVideoQosPreference	画质偏好。
TRTCDeviceState	设备操作。
TRTCDeviceType	设备类型。
TRTCWaterMarkSrcType	水印图片的源类型。
TRTCTranscodingConfigMode	混流参数配置模式。

版权所有：腾讯云计算（北京）有限责任公司

错误码

最近更新时间：2022-06-15 09:40:45

错误码表

基础错误码

符号	值	含义
ERR_NULL	0	无错误

进房相关错误码

TRTCCloud.enterRoom() 在进房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onEnterRoom() 和 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_ROOM_ENTER_FAIL	-3301	进入房间失败
ERR_ENTER_ROOM_PARAM_NULL	-3316	进房参数为空，请检查 TRTCCloud.enterRoom(): 接口调用是否传入有效的 param
ERR_SDK_APPID_INVALID	-3317	进房参数 sdkAppId 错误
ERR_ROOM_ID_INVALID	-3318	进房参数 roomId 错误
ERR_USER_ID_INVALID	-3319	进房参数 userID 不正确
ERR_USER_SIG_INVALID	-3320	进房参数 userSig 不正确
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	请求进房超时，请检查网络
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	权限位校验失败，请检查 privateMapKey 是否正确
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	服务不可用。请检查：套餐包剩余分钟数是否大于0，腾讯云账号是否欠费
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSig 校验失败，请检查 userSig 是否正确

版权所有：腾讯云计算（北京）有限责任公司

退房相关错误码

TRTCCloud.exitRoom() 在退房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	请求退房超时

设备（摄像头、麦克风、扬声器）相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败，例如在 Windows 或 Mac 设备，扬声器的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败

版权所有：腾讯云计算（北京）有限责任公司

屏幕分享相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_SCREEN_CAPTURE_START_FAIL	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了，如果在 Windows 或 Mac 系统的设备出现，请检查录屏接口的参数是否符合要求
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	录屏失败，在 Android 平台，需要5.0以上的系统，在 iOS 平台，需要11.0以上的系统
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	没有权限上行辅路
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	其他用户正在上行辅路
ERR_SCREEN_CAPTURE_STOPPED	-7001	录屏被系统中止

编解码相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_VIDEO_ENCODE_FAIL	-1303	视频帧编码失败，例如 iOS 设备切换到其他应用时，硬编码器可能被系统释放，再切换回来时，硬编码器重启前，可能会抛出
ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率
ERR_AUDIO_ENCODE_FAIL	-1304	音频帧编码失败，例如传入自定义音频数据，SDK 无法处理
ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率

自定义采集相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义

符号	值	含义
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	设置的 pixel format 不支持
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	设置的 buffer type 不支持

CDN 绑定和混流相关错误码

您可以通过回调函数 TRTCCloudDelegate.onStartPublishing() 和 TRTCCloudDelegate.onSetMixTranscodingConfig() 捕获相关通知。

符号	值	含义
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	旁路转推请求超时
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	云端混流请求超时
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	旁路转推回包异常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	云端混流回包异常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	开始向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	开始向腾讯云的直播 CDN 推流信令异常
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	停止向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	停止向腾讯云的直播 CDN 推流信令异常

跨房连麦相关错误码

TRTCCloud.ConnectOtherRoom() 在跨房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onConnectOtherRoom() 捕获相关通知。

符号	值	含义
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	请求连麦超时
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	请求退出连麦超时
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	无效参数

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	当前是观众角色，不能请求或断开跨房连麦，需要先 switchRole() 到主播
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	不支持跨房间连麦
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	达到跨房间连麦上限
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	跨房间连麦重试次数耗尽
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	跨房间连麦请求超时
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	跨房间连麦请求格式错误
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	跨房间连麦无签名
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	跨房间连麦签名解密失败
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	未找到跨房间连麦签名解密密钥
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	跨房间连麦签名解析错误
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	跨房间连麦签名时间戳错误
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	跨房间连麦签名内房间号不匹配
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	跨房间连麦签名内用户名不匹配
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	本用户未发起连麦

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	取消跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	被连麦房间不存在
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	被连麦房间达到连麦上限
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	被连麦用户不存在
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	被连麦用户已被删除
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	被连麦用户达到资源上限
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	连麦请求序号错乱

警告码表

警告码无需特别关注，您可以按需选择是否对当前用户进行相应的提示。

符号	值	含义
WARNING_HW_ENCODER_START_FAIL	1103	硬编码启动出现问题，自动切换到软编码
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	当前 CPU 使用率太高，无法满足软件编码需求，自动切换到硬件编码
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	摄像头采集帧率不足，部分自带美颜算法的 Android 手机上会出现
WARNING_SW_ENCODER_START_FAIL	1109	软编码启动失败
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	摄像头采集分辨率被降低，以满足当前帧率和性能最优解。

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_CAMERA_DEVICE_EMPTY	1111	没有检测到可用的摄像头设备
WARNING_CAMERA_NOT_AUTHORIZED	1112	用户未授权当前应用使用摄像头
WARNING_MICROPHONE_DEVICE_EMPTY	1201	没有检测到可用的麦克风设备
WARNING_SPEAKER_DEVICE_EMPTY	1202	没有检测到可用的扬声器设备
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	用户未授权当前应用使用麦克风
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	音频采集设备不可用（例如被占用）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	音频播放设备不可用（例如被占用）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	当前视频帧解码失败
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	当前音频帧解码失败
WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿
WARNING_HW_DECODER_START_FAIL	2106	硬解启动失败，采用软解码
WARNING_VIDEO_DECODER_HW_TO_SW	2108	当前流硬解第一个 I 帧失败，SDK 自动切软解
WARNING_SW_DECODER_START_FAIL	2109	软解码器启动失败
WARNING_VIDEO_RENDER_FAIL	2110	视频渲染失败
WARNING_START_CAPTURE_IGNORED	4000	已经在采集，启动采集被忽略
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	音频录制写入文件失败
WARNING_ROOM_DISCONNECT	5101	网络断开连接
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	当前是观众角色，忽略上行音视频数据
WARNING_NET_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
WARNING_RTMP_SERVER_RECONNECT	1102	直播推流，网络断连，已启动自动重连（自动重连连续失败超过三次会放弃）

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	直播拉流，网络断连, 已启动自动重连（自动重连连续失败超过三次会放弃）
WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
WARNING_RTMP_DNS_FAIL	3001	直播，DNS 解析失败
WARNING_RTMP_SEVER_CONN_FAIL	3002	直播，服务器连接失败
WARNING_RTMP_SHAKE_FAIL	3003	直播，与 RTMP 服务器握手失败
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	直播，服务器主动断开
WARNING_RTMP_READ_WRITE_FAIL	3005	直播，RTMP 读/写失败，将会断开连接
WARNING_RTMP_WRITE_FAIL	3006	直播，RTMP 写失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_READ_FAIL	3007	直播，RTMP 读失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_NO_DATA	3008	直播，超过30s 没有数据发送，主动断开连接
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	直播，connect 服务器调用失败（SDK 内部错误码，不会对外抛出）
WARNING_NO_STEAM_SOURCE_FAIL	3010	直播，连接失败，该流地址无视频（SDK 内部错误码，不会对外抛出）
WARNING_ROOM_RECONNECT	5102	网络断连，已启动自动重连
WARNING_ROOM_NET_BUSY	5103	网络状况不佳：上行带宽太小，上传数据受阻

版权所有：腾讯云计算（北京）有限责任公司

Flutter
API 概览

最近更新时间：2024-06-24 15:22:32

TRTCCloud

基础方法

API	描述
sharedInstance	创建 TRTCCloud 单例。
destroySharedInstance	销毁 TRTCCloud单例。
registerListener	设置事件监听
unRegisterListener	移除事件监听

房间相关接口函数

API	描述
enterRoom	进入房间，若房间不存在，系统将自动创建一个新房间。
exitRoom	离开房间。
switchRole	切换角色，仅适用于直播场景（TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM）。
setDefaultStreamRecvMode	设置音视频数据接收模式，需要在进房前设置才能生效。
connectOtherRoom	请求跨房通话（主播 PK）。
disconnectOtherRoom	退出跨房通话。
switchRoom	切换房间。
createSubCloud	创建房间子实例（用于并发多房间收听/观看）
destroySubCloud	销毁房间子实例

CDN 相关接口函数

API	描述
startPublishing	开始向腾讯云的直播 CDN 推流。
stopPublishing	停止向腾讯云的直播 CDN 推流。
startPublishCDNStream	开始向友商云的直播 CDN 转推。
stopPublishCDNStream	停止向非腾讯云地址转推。
setMixTranscodingConfig	设置云端的混流转码参数。
startPublishMediaStream	开始发布媒体流。
updatePublishMediaStream	更新发布媒体流。
stopPublishMediaStream	停止发布媒体流。

视频相关接口函数

API	描述
startLocalPreview	开启本地视频的预览画面。
updateLocalView	更新本地视频预览画面。
updateRemoteView	更新远端用户的视频渲染控件。
stopLocalPreview	停止本地视频采集及预览。
muteLocalVideo	暂停/恢复推送本地的视频数据。
startRemoteView	开始显示远端视频画面。
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流。
stopAllRemoteView	停止显示所有远端视频画面，同时不再拉取远端用户的视频数据流。
setVideoMuteImage	设置暂停推送本地视频时要推送的图片。
muteRemoteVideoStream	暂停/恢复接收指定的远端视频流。
muteAllRemoteVideoStreams	暂停/恢复接收所有远端视频流。
setVideoEncoderParam	设置视频编码器相关参数。
setNetworkQosParam	设置网络流控相关参数。
setLocalRenderParams	设置本地图像的渲染模式。

版权所有：腾讯云计算（北京）有限责任公司

setRemoteRenderParams	设置远端图像相关参数。
setVideoEncoderRotation	设置视频编码输出的画面方向，即设置远端用户观看到的和服务器录制的画面方向。
setVideoEncoderMirror	设置编码器输出的画面镜像模式。
setGSensorMode	设置重力感应的适应模式。
enableEncSmallVideoStream	开启大小画面双路编码模式。
setRemoteVideoStreamType	选定观看指定 uid 的大画面或小画面。
snapshotVideo	视频画面截图。
startLocalRecording	开始本地媒体录制，同时包含音频与视频数据。
stopLocalRecording	停止本地媒体录制。

音频相关接口函数

API	描述
startLocalAudio	开启本地音频的采集和上行。
stopLocalAudio	关闭本地音频的采集和上行。
muteLocalAudio	静音/取消静音本地的音频。
setAudioRoute	设置音频路由。
muteRemoteAudio	静音/取消静音指定的远端用户的声音。
muteAllRemoteAudio	静音/取消静音所有用户的声音。
setRemoteAudioVolume	设定某一个远端用户的声音播放音量
setAudioCaptureVolume	设置 SDK 采集音量。
getAudioCaptureVolume	获取 SDK 采集音量。
setAudioPlayoutVolume	设置 SDK 播放音量。
getAudioPlayoutVolume	获取 SDK 播放音量。
enableAudioVolumeEvaluation	启用音量大小提示。
startAudioRecording	开始录音。

版权所有：腾讯云计算（北京）有限责任公司

stopAudioRecording	停止录音。
setSystemVolumeType	设置通话时使用的系统音量类型。
startSystemAudioLoopback	启用系统音频采集。
stopSystemAudioLoopback	停止系统音频采集。
setSystemAudioLoopbackVolume	设置系统音频采集的音量。

设备管理接口

API	描述
getDeviceManager	获取设备管理模块。接口详情见设备管理接口文档

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象。接口详情见美颜管理文档
setWatermark	添加水印。

自定义采集和自定义渲染

API	描述
setLocalVideoRenderListener	设置本地视频的自定义渲染回调
setRemoteVideoRenderListener	设置远端视频自定义渲染回调
unregisterTexture	取消自定义的渲染回调
enableCustomVideoProcess	启用/禁用自定义视频处理
setAudioFrameListener	设置自定义音频数据回调

音乐特效和人声特效

API	描述
getAudioEffectManager	获取音效管理类 TXAudioEffectManager，用于管理BGM，短音效和人声特效。接口详情见音效管理文档

版权所有：腾讯云计算（北京）有限责任公司

辅流相关接口函数

API	描述
startScreenCapture	启动屏幕分享
stopScreenCapture	停止屏幕采集
pauseScreenCapture	暂停屏幕分享
resumeScreenCapture	恢复屏幕分享
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持 Windows ）
selectScreenCaptureTarget	选取要分享的屏幕或窗口（该接口仅支持 Windows）

自定义消息发送

API	描述
sendCustomCmdMsg	发送自定义消息给房间内所有用户。
sendSEIMsg	将小数据量的自定义数据嵌入视频帧中。

网络测试

API	描述
startSpeedTest	开始进行网络测速（视频通话期间请勿测试，以免影响通话质量）。
stopSpeedTest	停止服务器测速。

Log 相关接口函数

API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setLogDirPath	修改日志保存路径。
setLogCompressEnabled	启用或禁用 Log 的本地压缩。
setConsoleEnabled	启用或禁用控制台日志打印。
showDebugView	显示调试信息浮层(可以展示音视频信息和事件信息)。

版权所有：腾讯云计算（北京）有限责任公司

callExperimentalAPI	调用实验性 API。

TRTCCloudListener

腾讯云视频通话功能的事件回调接口。

错误事件和警告事件

API	描述
onError	错误回调，表示 SDK 不可恢复的错误，一定要监听并分情况给用户适当的界面提示。
onWarning	警告回调，用于告知您一些非严重性问题，例如出现卡顿或者可恢复的解码失败。

房间事件回调

API	描述
onEnterRoom	已加入房间的回调。
onExitRoom	离开房间的事件回调。
onSwitchRole	切换角色的事件回调。
onConnectOtherRoom	请求跨房通话（主播 PK）的结果回调。
onDisConnectOtherRoom	结束跨房通话（主播 PK）的结果回调。
onSwitchRoom	切换房间 (switchRoom) 的结果回调。

成员事件回调

API	描述
onRemoteUserEnterRoom	有用户加入当前房间。
onRemoteUserLeaveRoom	有用户离开当前房间。
onUserVideoAvailable	远端用户是否存在可播放的主路画面（一般用于摄像头）。
onUserSubStreamAvailable	远端用户是否存在可播放的辅路画面（一般用于屏幕分享）。
onUserAudioAvailable	远端用户是否存在可播放的音频数据。

版权所有：腾讯云计算（北京）有限责任公司

onFirstVideoFrame	开始渲染本地或远程用户的首帧画面。
onFirstAudioFrame	开始播放远程用户的首帧音频（本地声音暂不通知）。
onSendFirstLocalVideoFrame	首帧本地视频数据已经被送出。
onSendFirstLocalAudioFrame	首帧本地音频数据已经被送出。

录制任务回调

API	描述
onLocalRecordBegin	当本地录制任务已经开始时，SDK会通过这个回调通知。
onLocalRecording	当本地录制任务正在进行时，SDK会定期通过这个回调通知用户。
onLocalRecordFragment	当你启用分段录制时，SDK会在每次完成一个片段时通过这个回调通知你。
onLocalRecordComplete	当本地录制任务已经结束时，SDK会通过这个回调通知你。

播放背景音乐的回调接口

API	描述
onMusicObserverStart	音乐播放开始的回调通知。
onMusicObserverPlayProgress	音乐播放进度的回调通知。
onMusicObserverComplete	音乐播放结束的回调通知。

统计和质量回调

API	描述
onNetworkQuality	网络质量，该回调每2秒触发一次，统计当前网络的上行和下行质量。
onStatistics	技术指标统计回调。

服务器事件回调

API	描述
onConnectionLost	SDK 跟服务器的连接断开。
onTryToReconnect	SDK 尝试重新连接到服务器。

版权所有：腾讯云计算（北京）有限责任公司

onConnectionRecovery	SDK 跟服务器的连接恢复。
onSpeedTest	服务器测速的回调，SDK 对多个服务器 IP 做测速，每个 IP 的测速结果通过这个回调通知。

硬件设备事件回调

API	描述
onCameraDidReady	摄像头准备就绪。
onMicDidReady	麦克风准备就绪。
onUserVoiceVolume	用于提示音量大小的回调，包括每个 userId 的音量和远端总音量。
onDeviceChange	本地设备连接和断开的回调
onTestMicVolume	麦克风测试音量的回调
onTestSpeakerVolume	扬声器测试音量的回调
onAudioRouteChanged	音频路由改变的回调（仅限移动设备）

自定义消息的接收回调

API	描述
onRecvCustomCmdMsg	收到自定义消息回调。
onMissCustomCmdMsg	自定义消息丢失回调。
onRecvSEIMsg	收到 SEI 消息的回调。

CDN 旁路转推回调

API	描述
onStartPublishing	开始向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 startPublishing() 接口。
onStopPublishing	停止向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 stopPublishing() 接口。
onStartPublishCDNStream	启动旁路推流到 CDN 完成的回调。
onStopPublishCDNStream	停止旁路推流到 CDN 完成的回调。

版权所有：腾讯云计算（北京）有限责任公司

onSetMixTranscodingConfig	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 setMixTranscodingConfig() 接口。
onStartPublishMediaStream	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 startPublishMediaStream() 接口。
onUpdatePublishMediaStream	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 updatePublishMediaStream() 接口。
onStopPublishMediaStream	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 stopPublishMediaStream() 接口。

屏幕分享回调

API	描述
onScreenCaptureStarted	当屏幕分享开始时，SDK 会通过此回调通知
onScreenCapturePaused	当屏幕分享调用 pauseScreenCapture() 暂停时，SDK 会通过此回调通知。
onScreenCaptureResumed	当屏幕分享调用 resumeScreenCapture() 恢复时，SDK 会通过此回调通知。
onScreenCaptureStopped	当屏幕分享停止时，SDK 会通过此回调通知。

截图回调

API	描述
onSnapshotComplete	截图完成时回调。

TXAudioEffectManager

API	描述
enableVoiceEarMonitor	启用耳返，使本地用户可以听到自己的声音。
setVoiceEarMonitorVolume	设置耳返音量
setVoiceReverbType	设置语音混响效果（KTV、小房间、大会堂、低沉、洪亮等效果）
setVoiceChangerType	设置语音变声效果（萝莉、大叔、重金属、朋克等效果）
setVoiceCaptureVolume	设置麦克风语音音量
startPlayMusic	开始播放背景音乐

stopPlayMusic	停止播放背景音乐
pausePlayMusic	暂停播放背景音乐
resumePlayMusic	恢复播放背景音乐
setMusicPublishVolume	设置背景音乐的远程音量。主播可以使用此 API 设置远程观众听到的背景音乐的音量。
setMusicPlayoutVolume	设置背景音乐的本地音量。主播可以使用此 API 设置本地背景音乐的音量。
setAllMusicVolume	设置全局背景音乐的本地和远程音量
setMusicPitch	调整背景音乐的音调
setMusicSpeedRate	调整背景音乐的速度
getMusicCurrentPosInMS	获取背景音乐的当前播放进度（毫秒）
seekMusicToPosInMS	设置背景音乐的播放进度（毫秒）
getMusicDurationInMS	获取背景音乐文件的总时长（毫秒）
setVoicePitch	设置语音音调。

TXBeautyManager

API	描述
setBeautyStyle	设置美颜滤镜类型
setFilter	指定素材滤镜效果
setFilterStrength	设置滤镜强度
setBeautyLevel	设置美颜滤镜强度
setWhitenessLevel	设置亮白滤镜强度
enableSharpnessEnhancement	启用清晰度增强
setRuddyLevel	设置红润滤镜强度

TXDeviceManager

API	描述
isFrontCamera	设置是否使用前置摄像头
switchCamera	切换摄像头（前置/后置）
getCameraZoomMaxRatio	获取摄像头的最大缩放倍数
setCameraZoomRatio	设置摄像头的缩放因子（焦距）
enableCameraAutoFocus	设置是否启用面部位置自动识别
isAutoFocusEnabled	查询设备是否支持自动识别人脸位置
setCameraFocusPosition	设置摄像头对焦位置
enableCameraTorch	启用/禁用闪光灯，也就是手电筒模式
setSystemVolumeType	设置通话中使用的系统音量类型
setAudioRoute	设置音频路由，即使用听筒或者扬声器
getDevicesList	获取指定类型的设备列表
setCurrentDevice	设置当前要使用的设备
getCurrentDevice	获取当前正在使用的设备
setCurrentDeviceVolume	设置当前设备的音量
getCurrentDeviceVolume	获取当前设备的音量
setCurrentDeviceMute	设置当前设备的静音状态
getCurrentDeviceMute	查询当前设备的静音状态
startMicDeviceTest	开始麦克风测试
stopMicDeviceTest	停止麦克风测试
startSpeakerDeviceTest	开始扬声器测试
stopSpeakerDeviceTest	停止扬声器测试
setApplicationPlayVolume	设置 Windows 系统音量混合器中当前进程的音量
getApplicationPlayVolume	获取 Windows 系统音量混合器中当前进程的音量
setApplicationMuteState	设置 Windows 系统音量混合器中当前进程的静音状态

版权所有：腾讯云计算（北京）有限责任公司

getApplicationMuteState	获取 Windows 系统音量混合器中当前进程的静音状态

关键类型定义

类名	描述
TRTCCloudDef	关键类型定义变量。
TRTCParams	进房参数。
TRTCSwitchRoomConfig	切换房间参数参数。
TRTCVideoEncParam	视频编码参数。
TRTCNetworkQosParam	网络流控相关参数。
TRTCRenderParams	远端图像参数。
TRTCMixUser	云端混流中每一路子画面的位置信息。
TRTCTranscodingConfig	云端混流（转码）配置。
TXVoiceChangerType	变声类型定义（萝莉、大叔、重金属、外国人...）。
TXVoiceReverbType	变声类型定义（KTV、小房间、大会堂、低沉、洪亮...）。
AudioMusicParam	音乐和人声设置接口参数。
TRTCAudioRecordingParams	录音参数。
TRTCLocalRecordingParams	录制参数
TRTCPublishCDNParam	CDN转推参数。
CustomLocalRender	本地视频使用外部纹理渲染的参数
CustomRemoteRender	远程视频使用外部纹理渲染的参数
CustomRender	视频使用外部纹理渲染的参数
TRTCPublishMode	媒体流发布模式
TRTCPublishCdnUrl	配置将实时音视频（TRTC）流发布到腾讯云或第三方 CDN
TRTCUser	TRTC 用户的信息，主要包含用户ID和用户所在的房间号
TRTCPublishTarget	配置 TRTC 流的发布目标

版权所有：腾讯云计算（北京）有限责任公司

TRTCStreamEncoderParam	发布流的相关编码设置，包含分辨率、帧率、关键帧间隔等等
Rect	用于描述一些视图的坐标
TRTCVideoFillMode	TRTC 视频视图显示模式的枚举，包括填充模式和适应模式
TRTCVideoStreamType	TRTC 所提供的不同的视频流类型
TRTCVideoLayout	TRTC 混流转码时的视频布局属性配置，包括位置、大小、图层等
TRTCWatermark	TRTC 水印功能的属性配置
TRTCStreamMixingConfig	TRTC 混流转码的相关设置，包括背景颜色、背景图像、所有要混合的视频和音频流的信息以及水印设置等。
TRTCAudioFrame	音视频帧数据类，用于处理和传输音频数据。
TRTCScreenCaptureSourceList	屏幕窗口列表。
TRTCScreenCaptureSourceInfo	屏幕分享的目标信息（仅适用于桌面系统）
TRTCImageBuffer	TRTC 屏幕分享图标信息以及 mute image 垫片
TRTCScreenCaptureProperty	屏幕分享的进阶控制参数
TRTCScreenCaptureSourceType	屏幕共享目标类型（仅桌面）

视频渲染组件

API	描述
TRTCCloudVideoView	视频视图窗口，用于显示本地视频、远程视频或辅流。

错误码

最近更新时间：2022-06-15 09:40:45

错误码表

基础错误码

符号	值	含义
ERR_NULL	0	无错误

进房相关错误码

TRTCCloud.enterRoom() 在进房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onEnterRoom() 和 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_ROOM_ENTER_FAIL	-3301	进入房间失败
ERR_ENTER_ROOM_PARAM_NULL	-3316	进房参数为空，请检查 TRTCCloud.enterRoom(): 接口调用是否传入有效的 param
ERR_SDK_APPID_INVALID	-3317	进房参数 sdkAppId 错误
ERR_ROOM_ID_INVALID	-3318	进房参数 roomId 错误
ERR_USER_ID_INVALID	-3319	进房参数 userID 不正确
ERR_USER_SIG_INVALID	-3320	进房参数 userSig 不正确
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	请求进房超时，请检查网络
ERR_SERVER_INFO_PRIVILEGE_FLAG_ERROR	-100006	权限位校验失败，请检查 privateMapKey 是否正确
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	服务不可用。请检查：套餐包剩余分钟数是否大于0，腾讯云账号是否欠费
ERR_SERVER_INFO_ECDH_GET_TINYID	-100018	userSig 校验失败，请检查 userSig 是否正确

版权所有：腾讯云计算（北京）有限责任公司

退房相关错误码

TRTCCloud.exitRoom() 在退房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	请求退房超时

设备（摄像头、麦克风、扬声器）相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败，例如在 Windows 或 Mac 设备，扬声器的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败

版权所有：腾讯云计算（北京）有限责任公司

屏幕分享相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_SCREEN_CAPTURE_START_FAIL	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了，如果在 Windows 或 Mac 系统的设备出现，请检查录屏接口的参数是否符合要求
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	录屏失败，在 Android 平台，需要5.0以上的系统，在 iOS 平台，需要11.0以上的系统
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	没有权限上行辅路
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	其他用户正在上行辅路
ERR_SCREEN_CAPTURE_STOPPED	-7001	录屏被系统中止

编解码相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_VIDEO_ENCODE_FAIL	-1303	视频帧编码失败，例如 iOS 设备切换到其他应用时，硬编码器可能被系统释放，再切换回来时，硬编码器重启前，可能会抛出
ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率
ERR_AUDIO_ENCODE_FAIL	-1304	音频帧编码失败，例如传入自定义音频数据，SDK 无法处理
ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率

自定义采集相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义

符号	值	含义
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	设置的 pixel format 不支持
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	设置的 buffer type 不支持

CDN 绑定和混流相关错误码

您可以通过回调函数 TRTCCloudDelegate.onStartPublishing() 和 TRTCCloudDelegate.onSetMixTranscodingConfig() 捕获相关通知。

符号	值	含义
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	旁路转推请求超时
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	云端混流请求超时
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	旁路转推回包异常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	云端混流回包异常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	开始向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	开始向腾讯云的直播 CDN 推流信令异常
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	停止向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	停止向腾讯云的直播 CDN 推流信令异常

跨房连麦相关错误码

TRTCCloud.ConnectOtherRoom() 在跨房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onConnectOtherRoom() 捕获相关通知。

符号	值	含义
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	请求连麦超时
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	请求退出连麦超时
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	无效参数

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	当前是观众角色，不能请求或断开跨房连麦，需要先 switchRole() 到主播
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	不支持跨房间连麦
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	达到跨房间连麦上限
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	跨房间连麦重试次数耗尽
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	跨房间连麦请求超时
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	跨房间连麦请求格式错误
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	跨房间连麦无签名
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	跨房间连麦签名解密失败
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	未找到跨房间连麦签名解密密钥
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	跨房间连麦签名解析错误
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	跨房间连麦签名时间戳错误
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	跨房间连麦签名内房间号不匹配
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	跨房间连麦签名内用户名不匹配
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	本用户未发起连麦

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	取消跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	被连麦房间不存在
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	被连麦房间达到连麦上限
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	被连麦用户不存在
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	被连麦用户已被删除
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	被连麦用户达到资源上限
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	连麦请求序号错乱

警告码表

警告码无需特别关注，您可以按需选择是否对当前用户进行相应的提示。

符号	值	含义
WARNING_HW_ENCODER_START_FAIL	1103	硬编码启动出现问题，自动切换到软编码
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	当前 CPU 使用率太高，无法满足软件编码需求，自动切换到硬件编码
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	摄像头采集帧率不足，部分自带美颜算法的 Android 手机上会出现
WARNING_SW_ENCODER_START_FAIL	1109	软编码启动失败
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	摄像头采集分辨率被降低，以满足当前帧率和性能最优解。

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_CAMERA_DEVICE_EMPTY	1111	没有检测到可用的摄像头设备
WARNING_CAMERA_NOT_AUTHORIZED	1112	用户未授权当前应用使用摄像头
WARNING_MICROPHONE_DEVICE_EMPTY	1201	没有检测到可用的麦克风设备
WARNING_SPEAKER_DEVICE_EMPTY	1202	没有检测到可用的扬声器设备
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	用户未授权当前应用使用麦克风
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	音频采集设备不可用（例如被占用）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	音频播放设备不可用（例如被占用）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	当前视频帧解码失败
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	当前音频帧解码失败
WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿
WARNING_HW_DECODER_START_FAIL	2106	硬解启动失败，采用软解码
WARNING_VIDEO_DECODER_HW_TO_SW	2108	当前流硬解第一个 I 帧失败，SDK 自动切软解
WARNING_SW_DECODER_START_FAIL	2109	软解码器启动失败
WARNING_VIDEO_RENDER_FAIL	2110	视频渲染失败
WARNING_START_CAPTURE_IGNORED	4000	已经在采集，启动采集被忽略
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	音频录制写入文件失败
WARNING_ROOM_DISCONNECT	5101	网络断开连接
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	当前是观众角色，忽略上行音视频数据
WARNING_NET_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
WARNING_RTMP_SERVER_RECONNECT	1102	直播推流，网络断连，已启动自动重连（自动重连连续失败超过三次会放弃）

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	直播拉流，网络断连, 已启动自动重连（自动重连连续失败超过三次会放弃）
WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
WARNING_RTMP_DNS_FAIL	3001	直播，DNS 解析失败
WARNING_RTMP_SEVER_CONN_FAIL	3002	直播，服务器连接失败
WARNING_RTMP_SHAKE_FAIL	3003	直播，与 RTMP 服务器握手失败
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	直播，服务器主动断开
WARNING_RTMP_READ_WRITE_FAIL	3005	直播，RTMP 读/写失败，将会断开连接
WARNING_RTMP_WRITE_FAIL	3006	直播，RTMP 写失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_READ_FAIL	3007	直播，RTMP 读失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_NO_DATA	3008	直播，超过30s 没有数据发送，主动断开连接
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	直播，connect 服务器调用失败（SDK 内部错误码，不会对外抛出）
WARNING_NO_STEAM_SOURCE_FAIL	3010	直播，连接失败，该流地址无视频（SDK 内部错误码，不会对外抛出）
WARNING_ROOM_RECONNECT	5102	网络断连，已启动自动重连
WARNING_ROOM_NET_BUSY	5103	网络状况不佳：上行带宽太小，上传数据受阻

版权所有：腾讯云计算（北京）有限责任公司

Unity
API 概览

最近更新时间：2021-06-15 19:46:51

TRTC Unity Api概览

基础方法

API	描述
getTRTCShareInstance	创建 TRTCCloud 单例。
destroyTRTCShareInstance	释放 ITRTCCloud 单例对象。
addCallback	设置回调接口 TRTCCloudCallback。
removeCallback	移除事件回调。

房间相关接口函数

API	描述
enterRoom	进入房间，若房间不存在，系统将自动创建一个新房间。
exitRoom	离开房间。
switchRole	切换角色，仅适用于直播场景（TRTC_APP_SCENE_LIVE 和 TRTC_APP_SCENE_VOICE_CHATROOM）。
setDefaultStreamRecvMode	设置音视频数据接收模式，需要在进房前设置才能生效。
connectOtherRoom	请求跨房通话（主播 PK）。
disconnectOtherRoom	退出跨房通话。
switchRoom	切换房间。

CDN 相关接口函数

API	描述
startPublishing	开始向腾讯云的直播 CDN 推流。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
stopPublishing	停止向腾讯云的直播 CDN 推流。
startPublishCDNStream	开始向友商云的直播 CDN 转推。
stopPublishCDNStream	停止向非腾讯云地址转推。
setMixTranscodingConfig	设置云端的混流转码参数。

视频相关接口函数

API	描述
startLocalPreview	开启本地视频的预览画面，目前仅支持自定义渲染。
stopLocalPreview	停止本地视频采集及预览。
muteLocalVideo	暂停/恢复推送本地的视频数据。
startRemoteView	开始拉取并显示指定用户的远端画面，目前仅支持自定义渲染。
stopRemoteView	停止显示远端视频画面，同时不再拉取该远端用户的视频数据流。
stopAllRemoteView	停止显示所有远端视频画面，同时不再拉取远端用户的视频数据流。
muteRemoteVideoStream	暂停/恢复接收指定的远端视频流。
muteAllRemoteVideoStreams	暂停/恢复接收所有远端视频流。
setVideoEncoderParam	设置视频编码器相关参数。
setNetworkQosParam	设置网络流控相关参数。
setVideoEncoderMirror	设置编码器输出的画面镜像模式。

音频相关接口函数

API	描述
startLocalAudio	开启本地音频的采集和上行。
stopLocalAudio	关闭本地音频的采集和上行。
muteLocalAudio	静音/取消静音本地的音频。
muteRemoteAudio	静音/取消静音指定的远端用户的声音。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
muteAllRemoteAudio	静音/取消静音所有用户的声音。
setRemoteAudioVolume	设置某个远程用户的播放音量。
setAudioCaptureVolume	设置 SDK 采集音量。
getAudioCaptureVolume	获取 SDK 采集音量。
setAudioPlayoutVolume	设置 SDK 播放音量。
getAudioPlayoutVolume	获取 SDK 播放音量。
enableAudioVolumeEvaluation	启用音量大小提示。
startAudioRecording	开始录音。
stopAudioRecording	停止录音。

设备管理接口

API	描述
getDeviceManager	获取设备管理模块。接口详情请参见 设备管理详细接口。

音乐特效和人声特效

API	描述
getAudioEffectManager	获取音效管理类 TXAudioEffectManager，用于管理BGM，短音效和人声特效。接口详情见 音效管理详细接口。

自定义视频渲染

API	描述
setLocalVideoRenderCallback	设置本地视频自定义渲染。
setRemoteVideoRenderCallback	设置远端视频自定义渲染。

自定义消息发送

API	描述

版权所有：腾讯云计算（北京）有限责任公司

API	描述
sendSEIMsg	将小数据量的自定义数据嵌入视频帧中。

网络测试

API	描述
startSpeedTest	开始进行网络测速（视频通话期间请勿测试，以免影响通话质量）。
stopSpeedTest	停止服务器测速。

Log 相关接口函数

API	描述
getSDKVersion	获取 SDK 版本信息。
setLogLevel	设置 Log 输出级别。
setLogDirPath	修改日志保存路径。
setLogCompressEnabled	启用或禁用 Log 的本地压缩。
callExperimentalAPI	启用或禁用 Log 的本地压缩。

ITRTCCloudCallback

腾讯云音频通话功能的事件回调接口。

错误事件和警告事件

API	描述
onError	错误回调，表示 SDK 不可恢复的错误，一定要监听并分情况给用户适当的界面提示。
onWarning	警告回调，用于告知您一些非严重性问题，例如出现卡顿或者可恢复的解码失败。

房间事件回调

API	描述
onEnterRoom	已加入房间的回调。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onExitRoom	离开房间的事件回调。
onSwitchRole	切换角色的事件回调。
onConnectOtherRoom	请求跨房通话（主播 PK）的结果回调。
onDisConnectOtherRoom	结束跨房通话（主播 PK）的结果回调。
onSwitchRoom	切换房间 (switchRoom) 的结果回调。

成员事件回调

API	描述
onRemoteUserEnterRoom	有用户加入当前房间。
onRemoteUserLeaveRoom	有用户离开当前房间。
onUserVideoAvailable	用户是否开启摄像头视频。
onUserAudioAvailable	远端用户是否存在可播放的音频数据。
onFirstVideoFrame	开始渲染本地或远程用户的首帧画面。
onFirstAudioFrame	开始播放远程用户的首帧音频（本地声音暂不通知）。
onSendFirstLocalVideoFrame	首帧本地视频数据已经被送出。
onSendFirstLocalAudioFrame	首帧本地音频数据已经被送出。

统计和质量回调

API	描述
onNetworkQuality	网络质量，该回调每2秒触发一次，统计当前网络的上行和下行质量。
onStatistics	技术指标统计回调。

服务器事件回调

API	描述
onConnectionLost	SDK 跟服务器的连接断开。

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onTryToReconnect	SDK 尝试重新连接到服务器。
onConnectionRecovery	SDK 跟服务器的连接恢复。
onSpeedTest	服务器测速的回调，SDK 对多个服务器 IP 做测速，每个 IP 的测速结果通过这个回调通知。

硬件设备事件回调

API	描述
onCameraDidReady	摄像头准备就绪。
onMicDidReady	麦克风准备就绪。
onUserVoiceVolume	用于提示音量大小的回调，包括每个 userId 的音量和远端总音量。
onDeviceChange	本地设备通断回调。

自定义消息的接收回调

API	描述
onRecvSEIMsg	收到 SEI 消息的回调。

CDN 旁路转推回调

API	描述
onStartPublishing	开始向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 startPublishing() 接口。
onStopPublishing	停止向腾讯云的直播 CDN 推流的回调，对应于 TRTCCloud 中的 stopPublishing() 接口。
onStartPublishCDNStream	启动旁路推流到 CDN 完成的回调。
onStopPublishCDNStream	停止旁路推流到 CDN 完成的回调。
onSetMixTranscodingConfig	设置云端的混流转码参数的回调，对应于 TRTCCloud 中的 setMixTranscodingConfig() 接口。

版权所有：腾讯云计算（北京）有限责任公司

关键类型定义

类名	描述
TRTCParams	进房参数。
TRTCVideoEncParam	视频编码参数。
TRTCTranscodingConfig	云端混流（转码）配置。
TRTCSwitchRoomConfig	切换房间参数参数。
TRTCNetworkQosParam	网络流控相关参数。
TXVoiceReverbType	混响类型定义（KTV、小房间、大会堂、低沉、洪亮...）。
AudioMusicParam	音乐和人声设置接口参数。
TRTCAudioRecordingParams	录音参数。

设备管理详细接口

API	描述
isFrontCamera	判断当前是否为前置摄像头。
switchCamera	切换摄像头。
getCameraZoomMaxRatio	查询当前摄像头支持的最大缩放比例。
setCameraZoomRatio	设置当前摄像头的缩放比例。
isAutoFocusEnabled	查询是否支持自动识别人脸位置。
enableCameraAutoFocus	设置人脸自动识别。
setCameraFocusPosition	设置摄像头焦点。
enableCameraTorch	开关闪光灯。
setSystemVolumeType	设置通话时使用的系统音量类型。
setAudioRoute	设置音频路由。

版权所有：腾讯云计算（北京）有限责任公司

音乐特效和人声特效详细接口

API	描述
setVoiceReverbType	设置人声的混响效果（KTV、小房间、大会堂、低沉、洪亮...）。
setMusicObserver	设置背景音乐的播放进度回调接口。
startPlayMusic	开始播放背景音乐。
stopPlayMusic	停止播放背景音乐。
pausePlayMusic	暂停播放背景音乐。
resumePlayMusic	恢复播放背景音乐。
setMusicPublishVolume	设置背景音乐的远端音量大小，即主播可以通过此接口设置远端观众能听到的背景音乐的音量大小。
setMusicPlayoutVolume	设置背景音乐的本地音量大小，即主播可以通过此接口设置主播自己本地的背景音乐的音量大小。
setAllMusicVolume	设置全局背景音乐的本地和远端音量的大小。
setMusicPitch	调整背景音乐的音调高低。
setMusicSpeedRate	调整背景音乐的变速效果。
getMusicCurrentPosInMS	获取背景音乐当前的播放进度（单位：毫秒）
seekMusicToPosInMS	设置背景音乐的播放进度（单位：毫秒）
getMusicDurationInMS	获取景音乐文件的总时长（单位：毫秒）

版权所有：腾讯云计算（北京）有限责任公司

错误码

最近更新时间：2022-05-19 10:26:58

错误码表

基础错误码

符号	值	含义
ERR_NULL	0	无错误

进房相关错误码

TRTCCloud.enterRoom() 在进房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onEnterRoom() 和 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_ROOM_ENTER_FAIL	-3301	进入房间失败
ERR_ENTER_ROOM_PARAM_NULL	-3316	进房参数为空，请检查 TRTCCloud.enterRoom(): 接口调用是否传入有效的 param
ERR_SDK_APPID_INVALID	-3317	进房参数 sdkAppId 错误
ERR_ROOM_ID_INVALID	-3318	进房参数 roomId 错误
ERR_USER_ID_INVALID	-3319	进房参数 userID 不正确
ERR_USER_SIG_INVALID	-3320	进房参数 userSig 不正确
ERR_ROOM_REQUEST_ENTER_ROOM_TIMEOUT	-3308	请求进房超时，请检查网络
ERR_SERVER_INFO_SERVICE_SUSPENDED	-100013	服务不可用。请检查：套餐包剩余分钟数是否大于0，腾讯云账号是否欠费

退房相关错误码

TRTCCloud.exitRoom() 在退房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义

符号	值	含义
ERR_ROOM_REQUEST_QUIT_ROOM_TIMEOUT	-3325	请求退房超时

设备（摄像头、麦克风、扬声器）相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_CAMERA_START_FAIL	-1301	打开摄像头失败，例如在 Windows 或 Mac 设备，摄像头的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_CAMERA_NOT_AUTHORIZED	-1314	摄像头设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_CAMERA_SET_PARAM_FAIL	-1315	摄像头参数设置出错（参数不支持或其它）
ERR_CAMERA_OCCUPY	-1316	摄像头正在被占用中，可尝试打开其他摄像头
ERR_MIC_START_FAIL	-1302	打开麦克风失败，例如在 Windows 或 Mac 设备，麦克风的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_MIC_NOT_AUTHORIZED	-1317	麦克风设备未授权，通常在移动设备出现，可能是权限被用户拒绝了
ERR_MIC_SET_PARAM_FAIL	-1318	麦克风设置参数失败
ERR_MIC_OCCUPY	-1319	麦克风正在被占用中，例如移动设备正在通话时，打开麦克风会失败
ERR_MIC_STOP_FAIL	-1320	停止麦克风失败
ERR_SPEAKER_START_FAIL	-1321	打开扬声器失败，例如在 Windows 或 Mac 设备，扬声器的配置程序（驱动程序）异常，禁用后重新启用设备，或者重启机器，或者更新配置程序
ERR_SPEAKER_SET_PARAM_FAIL	-1322	扬声器设置参数失败
ERR_SPEAKER_STOP_FAIL	-1323	停止扬声器失败

屏幕分享相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_SCREEN_CAPTURE_START_FAIL	-1308	开始录屏失败，如果在移动设备出现，可能是权限被用户拒绝了，如果在 Windows 或 Mac 系统的设备出现，请检查录屏接口的参数是否符合要求
ERR_SCREEN_CAPTURE_UNSURPORT	-1309	录屏失败，在 Android 平台，需要5.0以上的系统，在 iOS 平台，需要11.0以上的系统
ERR_SERVER_CENTER_NO_PRIVILEDGE_PUSH_SUB_VIDEO	-102015	没有权限上行辅路
ERR_SERVER_CENTER_ANOTHER_USER_PUSH_SUB_VIDEO	-102016	其他用户正在上行辅路
ERR_SCREEN_CAPTURE_STOPPED	-7001	录屏被系统中止

编解码相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_VIDEO_ENCODE_FAIL	-1303	视频帧编码失败，例如 iOS 设备切换到其他应用时，硬编码器可能被系统释放，再切换回来时，硬编码器重启前，可能会抛出
ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率
ERR_AUDIO_ENCODE_FAIL	-1304	音频帧编码失败，例如传入自定义音频数据，SDK 无法处理
ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率

自定义采集相关错误码

您可以通过回调函数 TRTCCloudDelegate.OnError() 捕获相关通知。

符号	值	含义
ERR_PIXEL_FORMAT_UNSUPPORTED	-1327	设置的 pixel format 不支持
ERR_BUFFER_TYPE_UNSUPPORTED	-1328	设置的 buffer type 不支持

版权所有：腾讯云计算（北京）有限责任公司

CDN 绑定和混流相关错误码

您可以通过回调函数 TRTCCloudDelegate.onStartPublishing() 和 TRTCCloudDelegate.onSetMixTranscodingConfig() 捕获相关通知。

符号	值	含义
ERR_PUBLISH_CDN_STREAM_REQUEST_TIME_OUT	-3321	旁路转推请求超时
ERR_CLOUD_MIX_TRANSCODING_REQUEST_TIME_OUT	-3322	云端混流请求超时
ERR_PUBLISH_CDN_STREAM_SERVER_FAILED	-3323	旁路转推回包异常
ERR_CLOUD_MIX_TRANSCODING_SERVER_FAILED	-3324	云端混流回包异常
ERR_ROOM_REQUEST_START_PUBLISHING_TIMEOUT	-3333	开始向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_START_PUBLISHING_ERROR	-3334	开始向腾讯云的直播 CDN 推流信令异常
ERR_ROOM_REQUEST_STOP_PUBLISHING_TIMEOUT	-3335	停止向腾讯云的直播 CDN 推流信令超时
ERR_ROOM_REQUEST_STOP_PUBLISHING_ERROR	-3336	停止向腾讯云的直播 CDN 推流信令异常

跨房连麦相关错误码

TRTCCloud.ConnectOtherRoom() 在跨房失败时会触发此类错误码，您可以通过回调函数 TRTCCloudDelegate.onConnectOtherRoom() 捕获相关通知。

符号	值	含义
ERR_ROOM_REQUEST_CONN_ROOM_TIMEOUT	-3326	请求连麦超时
ERR_ROOM_REQUEST_DISCONN_ROOM_TIMEOUT	-3327	请求退出连麦超时
ERR_ROOM_REQUEST_CONN_ROOM_INVALID_PARAM	-3328	无效参数
ERR_CONNECT_OTHER_ROOM_AS_AUDIENCE	-3330	当前是观众角色，不能请求或断开跨房连麦，需要先 switchRole() 到主播

符号	值	含义
ERR_SERVER_CENTER_CONN_ROOM_NOT_SUPPORT	-102031	不支持跨房间连麦
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_NUM	-102032	达到跨房间连麦上限
ERR_SERVER_CENTER_CONN_ROOM_REACH_MAX_RETRY_TIMES	-102033	跨房间连麦重试次数耗尽
ERR_SERVER_CENTER_CONN_ROOM_REQ_TIMEOUT	-102034	跨房间连麦请求超时
ERR_SERVER_CENTER_CONN_ROOM_REQ	-102035	跨房间连麦请求格式错误
ERR_SERVER_CENTER_CONN_ROOM_NO_SIG	-102036	跨房间连麦无签名
ERR_SERVER_CENTER_CONN_ROOM_DECRYPT_SIG	-102037	跨房间连麦签名解密失败
ERR_SERVER_CENTER_CONN_ROOM_NO_KEY	-102038	未找到跨房间连麦签名解密密钥
ERR_SERVER_CENTER_CONN_ROOM_PARSE_SIG	-102039	跨房间连麦签名解析错误
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SIG_TIME	-102040	跨房间连麦签名时间戳错误
ERR_SERVER_CENTER_CONN_ROOM_SIG_GROUPID	-102041	跨房间连麦签名内房间号不匹配
ERR_SERVER_CENTER_CONN_ROOM_NOT_CONNED	-102042	跨房间连麦签名内用户名不匹配
ERR_SERVER_CENTER_CONN_ROOM_USER_NOT_CONNED	-102043	本用户未发起连麦
ERR_SERVER_CENTER_CONN_ROOM_FAILED	-102044	跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CANCEL_FAILED	-102045	取消跨房间连麦失败
ERR_SERVER_CENTER_CONN_ROOM_CONNED_ROOM_NOT_EXIST	-102046	被连麦房间不存在

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
ERR_SERVER_CENTER_CONN_ROOM_CONNED_REACH_MAX_ROOM	-102047	被连麦房间达到连麦上限
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_NOT_EXIST	-102048	被连麦用户不存在
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_DELETED	-102049	被连麦用户已被删除
ERR_SERVER_CENTER_CONN_ROOM_CONNED_USER_FULL	-102050	被连麦用户达到资源上限
ERR_SERVER_CENTER_CONN_ROOM_INVALID_SEQ	-102051	连麦请求序号错乱

警告码表

警告码无需特别关注，您可以按需选择是否对当前用户进行相应的提示。

符号	值	含义
WARNING_HW_ENCODER_START_FAIL	1103	硬编码启动出现问题，自动切换到软编码
WARNING_VIDEO_ENCODER_SW_TO_HW	1107	当前 CPU 使用率太高，无法满足软件编码需求，自动切换到硬件编码
WARNING_INSUFFICIENT_CAPTURE_FPS	1108	摄像头采集帧率不足，部分自带美颜算法的 Android 手机上会出现
WARNING_SW_ENCODER_START_FAIL	1109	软编码启动失败
WARNING_REDUCE_CAPTURE_RESOLUTION	1110	摄像头采集分辨率被降低，以满足当前帧率和性能最优解。
WARNING_CAMERA_DEVICE_EMPTY	1111	没有检测到可用的摄像头设备
WARNING_CAMERA_NOT_AUTHORIZED	1112	用户未授权当前应用使用摄像头
WARNING_MICROPHONE_DEVICE_EMPTY	1201	没有检测到可用的麦克风设备
WARNING_SPEAKER_DEVICE_EMPTY	1202	没有检测到可用的扬声器设备

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_MICROPHONE_NOT_AUTHORIZED	1203	用户未授权当前应用使用麦克风
WARNING_MICROPHONE_DEVICE_ABNORMAL	1204	音频采集设备不可用（例如被占用）
WARNING_SPEAKER_DEVICE_ABNORMAL	1205	音频播放设备不可用（例如被占用）
WARNING_VIDEO_FRAME_DECODE_FAIL	2101	当前视频帧解码失败
WARNING_AUDIO_FRAME_DECODE_FAIL	2102	当前音频帧解码失败
WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿
WARNING_HW_DECODER_START_FAIL	2106	硬解启动失败，采用软解码
WARNING_VIDEO_DECODER_HW_TO_SW	2108	当前流硬解第一个 I 帧失败，SDK 自动切软解
WARNING_SW_DECODER_START_FAIL	2109	软解码器启动失败
WARNING_VIDEO_RENDER_FAIL	2110	视频渲染失败
WARNING_START_CAPTURE_IGNORED	4000	已经在采集，启动采集被忽略
WARNING_AUDIO_RECORDING_WRITE_FAIL	7001	音频录制写入文件失败
WARNING_ROOM_DISCONNECT	5101	网络断开连接
WARNING_IGNORE_UPSTREAM_FOR_AUDIENCE	6001	当前是观众角色，忽略上行音视频数据
WARNING_NET_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
WARNING_RTMP_SERVER_RECONNECT	1102	直播推流，网络断连，已启动自动重连（自动重连连续失败超过三次会放弃）
WARNING_LIVE_STREAM_SERVER_RECONNECT	2103	直播拉流，网络断连，已启动自动重连（自动重连连续失败超过三次会放弃）
WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
WARNING_RTMP_DNS_FAIL	3001	直播，DNS 解析失败

版权所有：腾讯云计算（北京）有限责任公司

符号	值	含义
WARNING_RTMP_SEVER_CONN_FAIL	3002	直播，服务器连接失败
WARNING_RTMP_SHAKE_FAIL	3003	直播，与 RTMP 服务器握手失败
WARNING_RTMP_SERVER_BREAK_CONNECT	3004	直播，服务器主动断开
WARNING_RTMP_READ_WRITE_FAIL	3005	直播，RTMP 读/写失败，将会断开连接
WARNING_RTMP_WRITE_FAIL	3006	直播，RTMP 写失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_READ_FAIL	3007	直播，RTMP 读失败（SDK 内部错误码，不会对外抛出）
WARNING_RTMP_NO_DATA	3008	直播，超过30s 没有数据发送，主动断开连接
WARNING_PLAY_LIVE_STREAM_INFO_CONNECT_FAIL	3009	直播，connect 服务器调用失败（SDK 内部错误码，不会对外抛出）
WARNING_NO_STEAM_SOURCE_FAIL	3010	直播，连接失败，该流地址无视频（SDK 内部错误码，不会对外抛出）
WARNING_ROOM_RECONNECT	5102	网络断连，已启动自动重连
WARNING_ROOM_NET_BUSY	5103	网络状况不佳：上行带宽太小，上传数据受阻

版权所有：腾讯云计算（北京）有限责任公司