实时音视频

含 UI 集成方案



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（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

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。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
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。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

- 含 UI 集成方案
  - TUIKit 概述
  - 音视频通话
    - 组件介绍（TUICallKit）
    - 快速接入（TUICallKit）
      - Android
      - iOS
      - Web & H5 (Vue2/Vue3)
      - Web（React）
      - uni-app（客户端）
      - uni-app（小程序）
      - 微信小程序
      - Flutter
    - 界面定制（TUICallKit）
      - Android
      - iOS
      - Web
      - 微信小程序
      - Flutter
    - 离线唤醒（TUICallKit）
      - Flutter
      - Android
      - iOS
        - APNs
        - VoIP (可选)
      - uni-app
    - 美颜特效（TUICallKit）
      - Android
      - iOS
    - 云端录制（TUICallKit）
    - 通话记录（TUICallKit）
      - 通话记录简介
      - 通话记录看板
      - 通话状态回调
        - 通话状态回调
        - 通话事件回调
        - 回调配置
          - 回调配置 API 列表
          - 创建回调配置
          - 查询回调配置
          - 更新回调配置
          - 删除回调配置
      - REST API
        - REST API 简介

通过 callId 获取记录
通过条件获取记录
API 参考（TUICallKit）
Android
API 概览
TUICallKit
TUICallEngine
TUICallObserver
类型定义
iOS
API 概览
TUICallKit
TUICallEngine
TUICallObserver
类型定义
Web
API 概览
TUICallKit
TUICallEngine
TUICallEvent
微信小程序
API 概览
TUICallKit
TUICallEngine
TUICallEvent
uni-app（客户端）
API 概览
TUICallKit
TUICallEngine
事件
Flutter
API 概览
TUICallKit
TUICallEngine
TUICallObserver
类型定义
错误码表（TUICallKit）
升级方案（TUICallkit）
微信小程序
Web
Android
iOS
uni-app（客户端）
Flutter
uni-app（小程序）
发布日志（TUICallKit）
uni-app（小程序）

Web
Android & iOS
Uni-App
Flutter
微信小程序
常见问题（TUICallKit）
全平台
Web
Android
iOS
uni-app（客户端）
uni-app（小程序）
微信小程序
Flutter
多人音视频
组件介绍 (TUIRoomKit)
快速接入 (TUIRoomKit)
Web & H5 (Vue2/Vue3)
Android
iOS
uni-app（小程序）
Electron
Flutter
Windows
界面定制 (TUIRoomKit)
Android
iOS
Web
Electron
Windows
Flutter
API 参考 (TUIRoomKit)
iOS&Mac
API 概览
TUIRoomKit
TUIRoomEngine
TUIRoomObserver
TUICommonDefine
TUIRoomDefine
Android
API 概览
TUIRoomKit
TUIRoomEngine
TUIRoomObserver
TUICommonDefine
TUIRoomDefine
Windows

版权所有：腾讯云计算（北京）有限责任公司

API 概览
TUIRoomEngine
TUIRoomObserver
TUICommonDefine
TUIRoomDefine
Web
API 概览
TUIRoomKit
TUIRoomEngine
TUIRoomEvents
TUIRoomEngine Defines
Electron
API 概览
TUIRoomKit
TUIRoomEngine
TUIRoomEvent
TUIRoomEngine Defines
Flutter
API 概览
TUIRoomEngine
TUIRoomObserver
类型定义
uni-app（小程序）
API 概览
TUIRoomKit
TUIRoomEngine
TUIRoomEvents
TUIRoomEngine Defines
发布日志
常见问题
iOS
Android
Web
小程序
Flutter
2.0版本升级指引
错误码
语音互动直播
集成 TUIVoiceRoom (Android)
集成 TUIVoiceRoom (iOS)
集成 TUIVoiceRoom (小程序)
TUIVoiceRoom API 查询
TRTCVoiceRoom (iOS)
TRTCVoiceRoom (Android)
视频互动直播
集成 TUILiveRoom (Android)
集成 TUILiveRoom (iOS)

版权所有：腾讯云计算（北京）有限责任公司

集成 TUIPusher&TUIPlayer (Web)
TUILiveRoom API 查询
TRTCLiveRoom API (iOS)
TRTCLiveRoom API (Android)
TRTCLiveRoom API (Flutter)
在线 K 歌
组件介绍（TUIKaraoke）
快速集成（TUIKaraoke）
Android
iOS
方案介绍（TUIKaraoke）
实现步骤
歌曲同步
Android
iOS
歌词同步
Android
iOS
人声同步
Android
iOS
混流方案
Android
iOS
API 参考（TUIKaraoke）
Android
iOS
常见问题（TUIKaraoke）
Android
iOS

含 UI 集成方案
TUIKit 概述

最近更新时间：2023-12-08 14:53:41

TUIKit 是腾讯云音视频团队在5000+客户的服务积累中，结合业内主流的音视频场景，提炼出的解决方案，包含视频通话组件、直播组件、视频房间组件等多个客户端音视频组件，可以帮助开发者快速搭建诸如通话、客服、直播、语聊、教育等场景解决方案。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，在实时音视频控制台创建应用后，我们控制台会自动帮您创建一个相同 SDKAppID 的即时通信 IM 应用，即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

集成方案的差异

实时音视频服务提供含 UI 和不含 UI 两种集成方案：

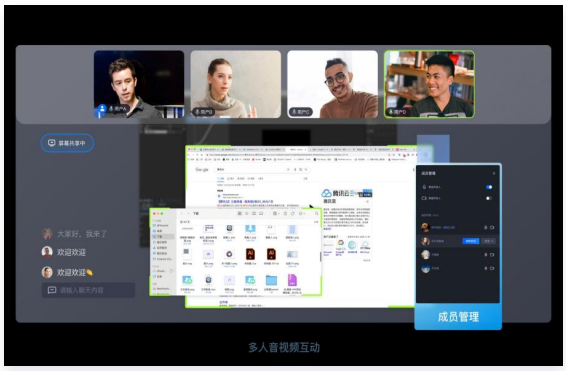
- 含 UI 集成方案代表使用 TUIKit 系列组件（如 TUICallKit、TUIRoomKit 等）进行集成；
- 无 UI 集成方案代表直接使用底层 SDK（如 TRTC SDK等）进行集成。

两种集成方案的具体差异见下表：

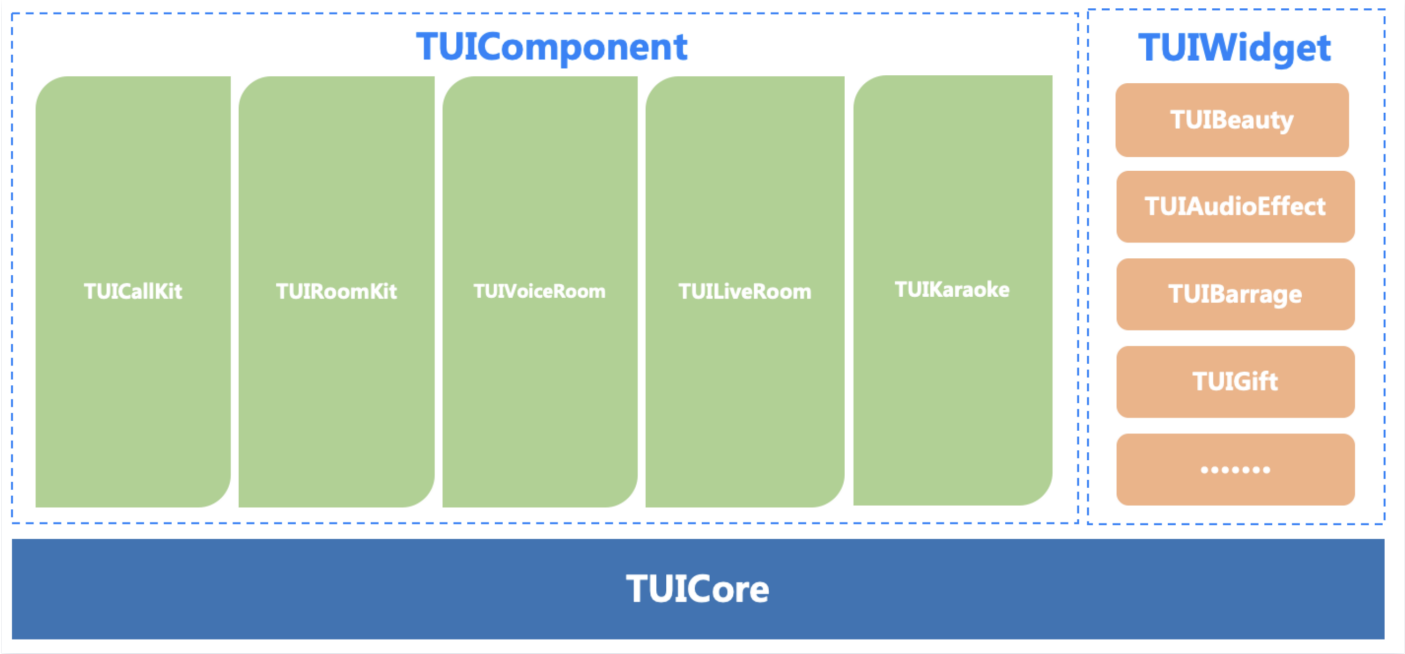
接入方式	含 UI 集成方案	不含 UI 集成方案
使用组件/SDK	音视频通话 TUICallKit、多人音视频房间 TUIRoomkit	TRTC SDK
适用场景	• 音视频通话 TUICallKit：适用于**9人及以下音视频通话场景**（参考微信语音/视频通话） • 多人音视频房间 TUIRoomKit：适用于**数十到上百人多人在线会议互动场景**（参考腾讯会议或者 Zoom） TUICallKit 和 TUIRoomKit 具体差异参见 组件差异	其他所有的音视频实时互动场景均适用（提供了底层的音视频实时通信能力，可以适配客户多样化的自定义开发需求，客户根据自有业务进行设计与开发）
上线时间要求	期望快速上线	开发时间充裕
开发资源	开发资源有限	开发资源充足
集成开发成本	接口使用更贴合业务，接入逻辑简单	提供底层通信能力，接入成本与业务的复杂程度有关系，接入相对较复杂
集成时间	**最快1小时**可完成集成	根据业务音视频需求的复杂度，时间从半个月到1个月以上不等
UI界面	**提供场景标准 UI，支持直接使用或基于 UI 组件自定义修改** • 音视频通话 TUICallKit：提供**微信视频聊天**同款 UI • 多人音视频房间 TUIRoomKit：提供**在线会议**标准UI	不包含UI，需客户结合业务来设计UI界面，自行实现页面的业务逻辑
业务灵活性	可在组件框架内自定义设计，实现部分个性化需求	可完全自定义开发
使用限制	组件能力依赖腾讯云即时通信 IM 服务，需要同时采	可以自行搭配其他产品或功能使用

	购 IM	
产品功能	• 音视频通话 TUICallKit：语音通话、视频通话、AI降噪、中途呼叫第三方等功能（参考微信语音/视频通话） • 多人音视频房间 TUIRoomKit：屏幕共享、百人开麦、会管会控、预约会议等功能（参考腾讯会议或者Zoom）	提供了多平台的音视频采集、发布、音视频互通、云端录制、审核等一系列全套功能服务，但需要与客户自己的业务进行关联和逻辑设计

组件介绍

含 UI 组件名称	音视频通话 TUICallkit	多人音视频房间 TUIRoomkit
适用场景	适用于**9人及以下**的音视频通话场景： 音视频通话、聊天应用、视频客服、在线问诊、保险咨询等	适用于**数十到上百人**多人在线会议互动场景： 企业培训、网络研讨会、在线教育、医疗会诊、在线招聘等
UI 界面	音视频通话 TUICallkit：提供「**微信视频聊天**」同款 UI 可直接使用或支持基于 UI 组件自定义修改	多人音视频房间 TUIRoomkit：提供「**在线会议**」标准 UI 可直接使用或支持基于 UI 组件自定义修改
默认 UI 样式		
产品功能	**概述：微信视频通话常见功能全部支持** 语音通话、视频通话、群组通话、AI降噪、中途呼叫第三方、小程序通话加速、离线推送、通话悬浮窗、通话记录、多端登录通话等功能	**概述：腾讯会议常见功能全部支持** 房间管理、成员管理、麦位管理、屏幕共享、百人开麦、预约会议、会议悬浮窗、会议录制、多端登录会议等功能
支持平台	• iOS • Android • Web/H5 • Flutter • Uni-App • 微信小程序（部分版本支持）	• iOS/Mac • Android • Windows • Web/H5 • Electron • Flutter • 微信小程序
Web Demo 体验	音视频通话 TUICallKit 体验	多人音视频 TUIRoomKit 体验

TUIKit 全家桶



如上图所示，TUIKit 分为 TUIComponent 和 TUIWidget 两种，同时支持可选的基础后台服务。

```
├── TUIComponent
│   ├── TUICallKit     // 通话组件（类微信通话），针对视频通话、客服、金融审核等音视频场景
│   ├── TUIRoomKit     // 多人视频房间组件，针对教育、会议、面试等音视频场景，具备禁言禁画等功能
│   ├── TUIVoiceRoom   // 语聊直播组件，针对交友语聊、排队语聊，音乐房等音频场景
│   ├── TUILiveRoom    // 视频互动直播组件，具备连麦、PK、音效等功能
│   ├── TUIKaraoke     // 创新组件，针对在线KTV、在线合唱等新玩法的音视频的组件，可以搭配腾讯云正版曲库解锁更多
玩法
├── TUIWidget
│   ├── TUIBeauty      // 美颜挂件
│   ├── TUIAudioEffect // 音效挂件
│   ├── TUIBarrage     // 弹幕挂件
│   ├── TUIGift        // 礼物挂件
```

交流和反馈

当然，一样的环境，一样的步骤，可能也会有不一样的报错，这是工程师逃不开的魔咒，您可以通过如下方式来尝试解决：

- 常见问题：TUICallKit 、TUIRoomKit
- 更多联系

音视频通话
组件介绍（TUICallKit）

最近更新时间：2024-01-01 13:30:41

组件介绍

TUICallKit 是腾讯云推出一款音视频通话 UI 组件，通过集成该组件，您只需要编写几行代码就可以为您的 App 添加音视频通话功能，并且支持离线唤起能力。TUICallKit 支持 Android、iOS、Web、微信小程序、Flutter、UniApp 等多个开发平台，基本功能如下图所示：



注意

基于开发者反馈和市场需求调研，2022年08月 TUICalling 正式升级为 TUICallKit，新的组件升级了群内多人通话和离线唤醒等功能，并且配备了更低折扣的音视频通话 SDK 套餐包以及7天的免费试用期，期待您的关注与使用！

功能优势

- 接入方便：提供带 UI 的开源组件，节省90%开发时间，快速上线音视频通话应用。
- 平台互通：各平台的 TUICallKit 组件支持相互拨打、接听、挂断等，互联互通。
- 多人通话：不仅仅支持1对1视频通话，还支持在群组内发起多人（最大9人）的视频通话。
- 离线推送：支持 Android&iOS 离线推送，被叫用户 App 不在线时也能收到新的来电消息。

适用场景

两人或多人进行音视频通话，覆盖游戏社交、在线客服、视频客服、在线问诊、保险咨询等场景。

功能与计费说明

TUICallKit 由实时音视频 TRTC 和即时通信 IM 提供底层技术支持，**您需要购买指定套餐后方可使用 TUICallKit**，其中免费体验版的开通方式可参见 开通 TUICallKit 体验版 。
下表为 TUICallKit 推荐套餐对应的功能和价格，您也可以自由组合 TRTC 包月套餐和 IM 版本以获得对应版本的 TUICallKit。TRTC 基础版配合 IM 专业版或旗舰版可使用TUICallKit 的1v1通话版；TRTC 尊享版和旗舰版配合 IM 专业版或旗舰版可使用TUICallKit 的群组通话版。TUICallKit 计费详细说明可参考 音视频通话 SDK 计费说明 。

推荐套餐		体验版	1V1通话版	群组通话版 38万分钟	群组通话版 140万分钟
价格	折扣价	0元/7天	1999元/月	4499元/月	7999元/月

			立即选购	立即选购	立即选购
	刊例价	~~0元/7天~~	~~2124元/月~~	~~4874元/月~~	~~9249元/月~~
包含套餐版本		TRTC 体验版 + IM 任意版本	TRTC 基础版 + IM 专业版	TRTC 尊享版 + IM 旗舰版	TRTC 旗舰版 + IM 旗舰版
赠送资源 (超出后付费)	通话时长	–	110,000分钟/月	380,000分钟/月	1400,000分钟/月
	日活跃用户数（DAU）	根据 IM 版本而定	10,000 DAU	10,000 DAU	10,000 DAU
	峰值群组数	根据 IM 版本而定	100000个/月	100000个/月	100000个/月
TUICallKit 专属功能	1v1 视频、音频通话	✓	✓	✓	✓
	微信同款 UI 设计	✓	✓	✓	✓
	通话状态展示	✓	✓	✓	✓
	离线推送（应用未打开时，也能通过消息推送让用户收到音视频通话的响铃呼叫）	✓	✓	✓	✓
	通话悬浮窗（通话界面收起后，可以悬浮窗样式悬浮在应用界面之上）	✓	✓	✓	✓
	自定义呼叫铃声	✓	✓	✓	✓
	呼叫/接听/拒绝/挂断	✓	✓	✓	✓
	视频通话切换语音通话	✓	✓	✓	✓
	群组通话	✓	–	✓	✓
	中途呼叫/加入三方通话	✓	–	✓	✓
	通话记录看板	✓	–	✓	✓
	小程序通话加速（支持小程序端通话与小程序进房通话加速优化，保障通话效果）	✓	–	✓	✓
	多端登录通话（任一平台接通，自动停止其他端的接入请求）	✓ （仅配合 IM 旗舰版使用）	–	✓	✓
	同平台多端登录通话	✓	–	✓	✓

版权所有：腾讯云计算（北京）有限责任公司

	（支持同一个平台，如多个 iOS设备登录）	（仅配合 IM 旗舰版使用）			
	AI 降噪（借助 AI 能力有效过滤背景噪声，还原通话效果）	✓	–	✓	✓
	弱网通话卡顿优化（针对户外等弱网环境下优化卡顿率，秒开速度更快）	✓	–	✓	✓
即时通信 IM 功能		根据 IM 版本而定 详见 IM 功能	详见 IM 专业版功能	详见 IM 旗舰版功能	详见 IM 旗舰版功能
实时音视频 TRTC 功能		详见 TRTC 体验版功能	详见 TRTC 基础版功能	详见 TRTC 尊享版功能	详见 TRTC 旗舰版功能
超出赠送资源部分后付费		通话时长后付费价格详见 音视频时长计费说明，DAU 与峰值群组数后付费价格详见 IM 计费说明			
支持平台		iOS、Android、Web、**微信小程序（仅体验版和群组通话版支持）**、uni-app、Flutter			

说明：

- **包含套餐版本：**TUICallKit 由 TRTC 和 IM 联合提供，使用 TUICallKit 需同时购买 TRTC 包月套餐和 IM 基础套餐包，不同版本套餐提供的 TUICallKit 功能有所不同。
- **DAU：**单个用户当日登录即时通信 IM 计为1个 DAU，同一用户重复登录时，DAU 不累加。
- **峰值群组数：**单个应用（SDKAppID）下所有用户创建和加入的群组数的总和，以自然月中最高峰值进行结算。

开通 TUICallKit 体验版

为了您能更好地体验音视频通话功能，我们免费为每个 SDKAppID 提供了TUICallKit 7天体验版（体验版不额外赠送通话时长），每个 SDKAppID 可免费体验2次，每次有效期均为7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。
您可以在 TRTC 控制台中开通 TUICallKit 体验版，具体操作步骤如下：

1. 访问 TRTC 控制台应用管理，点击需要使用 TUICallKit 应用（SDKAppID）后的**详情**，进入应用概览页面。
2. 找到页面右下角的**含 UI 低代码集成接入 > 音视频通话（TUICallKit）**区域，单击卡片内的 **领取体验** 即可开通 TUICallKit 的 7 天免费试用服务。



3. 体验版完成后，您可参见 TUICallKit 版本说明 确认所需功能，单击 前往购买 购买正式版 TUICallKit 。

> **说明：**
> 开通 TUICallKit 的体验版将默认开通 TRTC 包月套餐的体验版，二者体验版开通次数和规则相同。

TUICallKit 下载

Android TUICallKit

类“微信” UI，支持 1V1 通话、群组通话、悬浮窗、离线推送等特性，功能强大。

Github 快速接入 API 参考 常见问题

iOS TUICallKit

类“微信” UI，支持 1V1 通话、群组通话、悬浮窗、离线推送等特性，功能强大。

Github 快速接入 API 参考 常见问题

Web TUICallKit

类“微信” UI，支持 1V1 通话、群组通话、离线推送等特性，功能强大。

Github 快速接入 API 参考 常见问题



小程序 TUICallKit

类“微信” UI，支持 1V1 通话、群组通话等特性，功能强大。

Github 快速接入 API 参考 常见问题



uni-app（客户端） TUICallKit

类“微信” UI，支持 1V1 通话、群组通话、悬浮窗、离线推送等特性，功能强大。

Github 快速接入 API 参考



uni-app（小程序） TUICallKit

类“微信” UI，支持 1V1 通话、群组通话等特性，功能强大。

Github 快速接入 API 参考



Flutter TUICallKit

类“微信” UI，支持 1V1 通话、群组通话、悬浮窗、离线推送等特性，功能强大。

Github 快速接入 API 参考

开源建设

您可以在 这里 找到升级前的 TUICalling 开源项目。

快速接入（TUICallKit）
Android

最近更新时间：2023-12-26 17:09:42

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

环境准备

- 最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本。
- Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本）。
- Android 4.4 及以上的手机设备。

步骤一：开通服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。
您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务：

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建出的应用，进入基本配置页面，并在页面的右下角找到含 UI 低代码场景方案功能区，单击免费体验即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



> **注意**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和 **密钥(SecretKey)** 并记录下来，它们会在后续的 步骤四：登录 TUI 组件 中被用到。



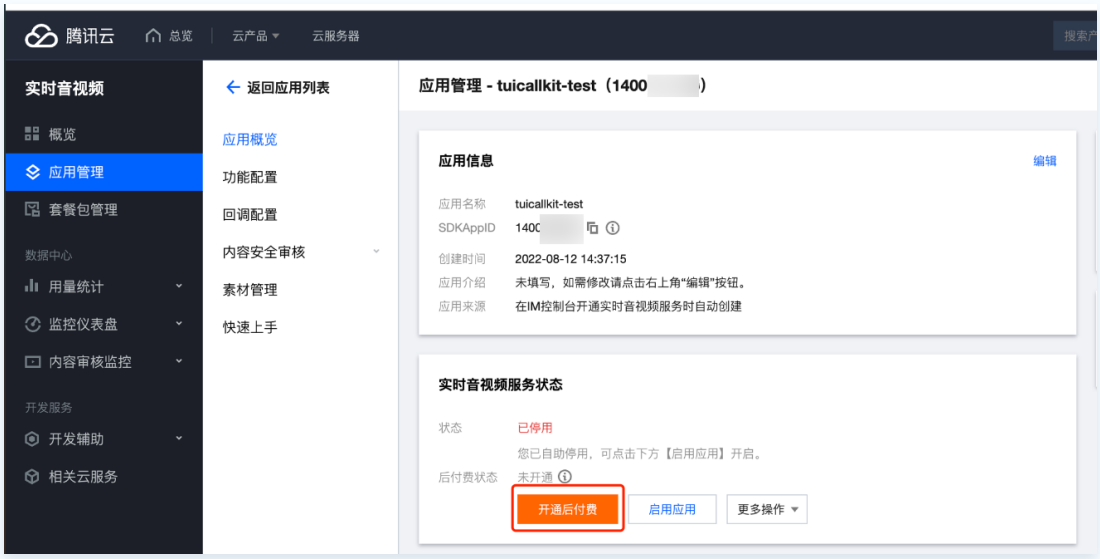
友情提示：

单击免费体验以后，部分之前使用过 实时音视频 TRTC 服务的用户会提示：

```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or the Tencent Cloud accountis in arrears
```

因为新的 IM 音视频通话能力是整合了腾讯云实时音视频 TRTC 和 即时通信 IM 两个基础的 PaaS 服务，所以当 实时音视频 TRTC 的免费额度（10000分钟）已经过期或者耗尽，就会导致开通此项服务失败，这里您可以单击 TRTC 控制台，找到对应

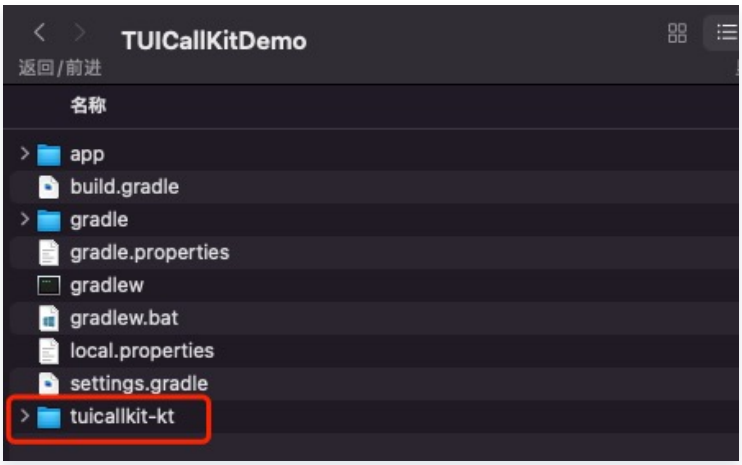
SDKAppID 的应用管理页，示例如图，开通后付费功能后，再次**启用应用**即可正常体验音视频通话能力。



步骤二：下载并导入组件

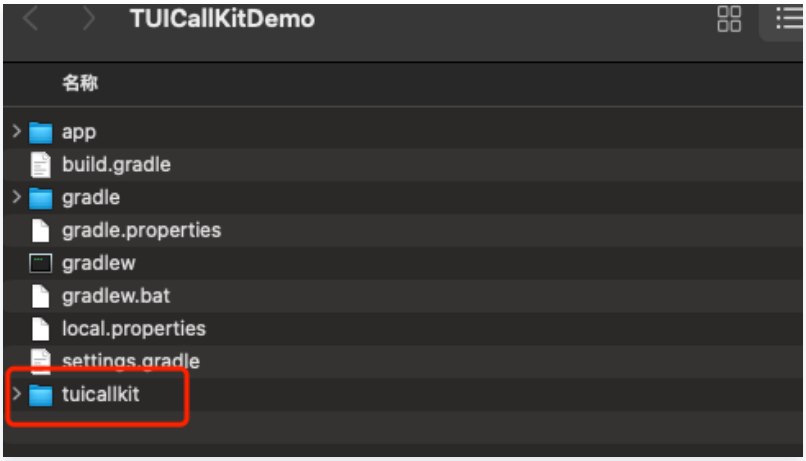
Kotlin

在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuicallkit-kt 子目录到您当前工程中的 app 同一级目录中，如下图：



Java

在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuicallkit 子目录到您当前工程中的 app 同一级目录中，如下图：



步骤三：完成工程配置

Kotlin

1. 在工程根目录下找到 settings.gradle 文件，并在其中增加如下代码，它的作用是将 步骤二 中下载的 tuicallkit-kt 组件导入到您当前的项目中：

```
include ':tuicallkit-kt'
```

2. 在 app 目录下找到 build.gradle 文件，并在其中增加如下代码，它的作用是声明当前 app 对新加入的 tuicallkit-kt 组件的依赖：

```
api project(':tuicallkit-kt')
```

> **说明**
> TUICallKit 工程内部已经默认依赖： TRTC SDK 、 IM SDK 、 tuicallengine 以及公共库 tuicore ，不需要开发者单独配置。如需进行版本升级，则修改 tuicallkit-kt/build.gradle 文件即可。

3. 由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 proguard-rules.pro 文件中添加如下代码：

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

> **注意**
> TUICallKit-kt 会在内部帮助您动态申请相机、麦克风、读取存储权限等。如果因为您的业务问题需要删减，可以在 tuicallkit-kt/src/main/AndroidManifest.xml 中进行修改。

Java

1. 在工程根目录下找到 settings.gradle 文件，并在其中增加如下代码，它的作用是将 步骤二 中下载的 tuicallkit 组件导入到您当前的项目中：

```
include ':tuicallkit'
```

2. 在 app 目录下找到 build.gradle 文件，并在其中增加如下代码，它的作用是声明当前 app 对新加入的 tuicallkit 组件的依赖：

```
api project(':tuicallkit')
```

> **说明**
> TUICallKit 工程内部已经默认依赖： TRTC SDK 、 IM SDK 、 tuicallengine 以及公共库 tuicore ，不需要开发者单独配置。如需进行版本升级，则修改 tuicallkit/build.gradle 文件即可。

3. 由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 proguard-rules.pro 文件中添加如下代码：

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

> **注意**
> TUICallKit 会在内部帮助您动态申请相机、麦克风、读取存储权限等，如果因为您的业务问题需要删减，可以请修改 tuicallkit/src/main/AndroidManifest.xml 。

步骤四：登录 TUI 组件

在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Kotlin

```
//设置对登录结果的监听器
private val listener: TUILoginListener = object : TUILoginListener() {
    override fun onKickedOffline() {
        super.onKickedOffline()
        Log.i(TAG, "You have been kicked off the line. Please login again!")
        //logout();
    }

    override fun onUserSigExpired() {
        super.onUserSigExpired()
        Log.i(TAG, "Your user signature information has expired")
        //logout();
    }
}

TUILogin.addLoginListener(listener)
```

```
// 登录
TUILogin.login(this,
    1400000001,    // 请替换为步骤一取到的 SDKAppID
    "denny",       // 请替换为您的 UserID
    "xxxxxxxxxxx",  // 您可以在控制台中计算一个 UserSig 并填在这个位置
    object : TUICallback() {
      override fun onSuccess() {
        // 呼叫视频通话,第一个参数为被呼叫者的UserID,第二参数为通话类型，包括语音通话，视频通话
        TUICallKit.createInstance(this@MainActivity).call("userID", TUICallDefine.MediaType.Video)
      }

      override fun onError(errorCode: Int, errorMessage: String) {
      }
    })
}
```

Java

```
//设置对登录结果的监听器
private final TUILoginListener mLoginListener = new TUILoginListener() {
    @Override
    public void onKickedOffline() {
        super.onKickedOffline();
        Log.i(TAG, "You have been kicked off the line. Please login again!");
        //logout();
    }
    @Override
    public void onUserSigExpired() {
        super.onUserSigExpired();
        Log.i(TAG, "Your user signature information has expired");
        //logout();
    }
};
TUILogin.addLoginListener(mLoginListener);

//登录
TUILogin.login(context,
    1400000001,    // 请替换为步骤一取到的 SDKAppID
    "denny",       // 请替换为您的 UserID
    "xxxxxxxxxxx",  // 您可以在控制台中计算一个 UserSig 并填在这个位置
    new TUICallback() {
    @Override
    public void onSuccess() {
        Log.i(TAG, "login success");
    }

    @Override
    public void onError(int errorCode, String errorMessage) {
        Log.e(TAG, "login failed, errorCode: " + errorCode + " msg:" + errorMessage);
    }
});
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在步骤一中的最后一步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 步骤一 的第3步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig。
- 更多信息请参见 如何计算及使用 UserSig 。

> ⚠ **注意**
>
> **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
>
> - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
> - UserSig 被错配成了加密密钥（SecretKey），UserSig 是用 SecretKey 把 SDKAppID、UserID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 UserSig。
> - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如“1”、“123”、“111”这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。
>
> Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 app 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤五：拨打通话

1对1视频通话

通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

Kotlin

```
// 发起1对1视频通话(假设 UserID 为 mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video)
```

Java

```
// 发起1对1视频通话(假设 UserID 为 mike)
TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video);
```

参数	类型	含义
userId	String	目标用户的 UserID：`"mike"`
callMediaType	TUICallDefine.MediaType	通话的媒体类型，示例：`TUICallDefine.MediaType.Video`

群内视频通话

通过调用 TUICallKit 的 groupCall 函数并指定通话类型和被叫方的 UserID 列表，就可以发起群内的语音或者视频通话。

Kotlin

```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mike", "tommy"), TUICallDefine.MediaType.Video)
```

Java

```
TUICallKit.createInstance(context).groupCall("12345678", Arrays.asList("jane", "mike", "tommy"),TUICallDefine.MediaType.Video);
```

参数	类型	含义
groupId	String	群组 Id，示例：`"12345678"`
userIdList	List	目标用户的 UserID 列表，示例：`{"jane", "mike", "tommy"}`
callMediaType	TUICallDefine.MediaType	通话的媒体类型，示例：`TUICallDefine.MediaType.Video`

说明

- 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。
- TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈： TUIKit 产品反馈问卷 。

步骤六：接听通话

收到来电请求后，TUICallKit 组件会自动唤起来电提醒的接听界面，不过因为 Android 系统权限的原因，分为如下几种情况：

- 您的 App 在前台时，当收到邀请时会自动弹出呼叫界面并播放来电铃音。
- 您的 App 在后台，但是有授予 悬浮窗权限 或 后台弹出应用 等权限时，仍然会自动弹出呼叫界面并播放来电铃音。
- 您的 App 在后台，且没有授予 悬浮窗权限 或 后台弹出应用 等权限时，TUICallKit 会播放来电铃音，提示用户接听或挂断。
- 您的 App 进程已经被销毁，只能通过接入 离线唤醒（Android），通过通知栏消息提示用户接听或挂断。

步骤七：更多特性

一、设置昵称&头像

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

Kotlin

```
TUICallKit.createInstance(context).setSelfInfo("jack", "https:/****/user_avatar.png", callback)
```

Java

```
TUICallKit.createInstance(context).setSelfInfo("jack", "https:/****/user_avatar.png", callback);
```

注意

因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

二、离线唤醒

完成以上步骤，就可以实现音视频通话的拨打和接通，但如果您的业务场景需要在 `应用的进程被杀死后` 或者 `应用退到后台后` ，还可以正常接收到音视频通话请求，就需要增加离线唤醒功能，详情见 离线唤醒（Android）。

三、悬浮窗功能

如果您的业务需要开启悬浮窗功能，您可以在 TUICallKit 组件初始化时调用以下接口开启该功能：

Kotlin

```
TUICallKit.createInstance(context).enableFloatWindow(true)
```

Java

```
TUICallKit.createInstance(context).enableFloatWindow(true);
```

四、通话状态监听

如果您的业务需要 **监听通话的状态**，例如通话开始、结束，以及通话过程中的网络质量等等（详见 TUICallObserver），可以监听以下事件：

Kotlin

```
private val observer: TUICallObserver = object : TUICallObserver() {
    override fun onCallBegin(roomId: TUICommonDefine.RoomId?, callMediaType: TUICallDefine.MediaType?,
callRole: TUICallDefine.Role?) {
    }
    override fun onCallEnd(roomId: TUICommonDefine.RoomId?, callMediaType: TUICallDefine.MediaType?,
callRole: TUICallDefine.Role?, totalTime: Long) {
    }
    override fun onUserNetworkQualityChanged(networkQualityList:
MutableList<TUICommonDefine.NetworkQualityInfo>?) {
    }
}
private fun initData() {
    TUICallEngine.createInstance(context).addObserver(observer)
}
```

Java

```
private TUICallObserver observer = new TUICallObserver() {
    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole) {
    }
    public void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType
callMediaType,TUICallDefine.Role callRole, long totalTime) {
    }
```

```
    public void onUserNetworkQualityChanged(List<TUICommonDefine.NetworkQualityInfo>
networkQualityList) {
    }
};

private void initData(){
    TUICallEngine.createInstance(context).addObserver(observer);
}
```

> ⚠ **注意：**
> 设置 TUICallObserver 监听回调，需保证回调所在类不会被清除，举例：不建议在 LoginActivity 中添加监听，LoginActivity 销毁的时候，回调也会被清除；建议在应用的 Application 中或者应用主界面进行监听。

五、自定义铃音

如果您需要自定义来电铃音，可以通过如下接口进行设置：

Kotlin

```
TUICallKit.createInstance(context).setCallingBell(filePath)
```

Java

```
TUICallKit.createInstance(context).setCallingBell(filePath);
```

六、自定义通话超时时间

如果您需要自定义通话的超时时间，可以在调用 call 、groupCall 、inviteUser 方法的时候修改 CallParams 参数进行设置：

Kotlin

```
val params = CallParams()
params.offlinePushInfo = createOfflinePushInfo(this)
params.timeout = 30
params.userData = "user data"

TUICallEngine.createInstance(context).call(roomId, userId, mediaType, params, null)
```

Java

```
TUICallDefine.CallParams params = new TUICallDefine.CallParams();
params.offlinePushInfo = OfflinePushInfoConfig.createOfflinePushInfo(mContext);
params.timeout = 30;
params.userData = "user data";

TUICallEngine.createInstance(context).call(roomId, userId, mediaType, params, null);
```

常见问题

1、错误提示“The package you purchased does not support this ability”？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2、如何购买套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请点击页面右侧，进行售前套餐咨询，或者加入 QQ 群：**605115878**，进行咨询和反馈。

3、TUICallKit 崩溃，崩溃日志："No implementation found for xxxx"?

堆栈信息如下：

```
java.lang.UnsatisfiedLinkError: No implementation found for void
com.tencent.liteav.base.Log.nativeWriteLogToNative(int, java.lang.String, java.lang.String) (tried
Java_com_tencent_liteav_base_Log_nativeWriteLogToNative and
Java_com_tencent_liteav_base_Log_nativeWriteLogToNative__ILjava_lang_String_2Ljava_lang_String_2)
    at com.tencent.liteav.base.Log.nativeWriteLogToNative(Native Method)
    at com.tencent.liteav.base.Log.i(SourceFile:177)
    at com.tencent.liteav.basic.log.TXCLog.i(SourceFile:36)
    at com.tencent.liteav.trtccalling.model.impl.base.TRTCLogger.i(TRTCLogger.java:15)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.init(ServiceInitializer.java:36)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.onCreate(ServiceInitializer.java:101)
    at android.content.ContentProvider.attachInfo(ContentProvider.java:2097)
    at android.content.ContentProvider.attachInfo(ContentProvider.java:2070)
    at android.app.ActivityThread.installProvider(ActivityThread.java:8168)
    at android.app.ActivityThread.installContentProviders(ActivityThread.java:7709)
    at android.app.ActivityThread.handleBindApplication(ActivityThread.java:7573)
    at android.app.ActivityThread.access$2600(ActivityThread.java:260)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2435)
    at android.os.Handler.dispatchMessage(Handler.java:110)
    at android.os.Looper.loop(Looper.java:219)
    at android.app.ActivityThread.main(ActivityThread.java:8668)
    at java.lang.reflect.Method.invoke(Native Method)
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:513)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1109)
```

TUICallkit不支持虚拟机，请用真机进行体验；如在真机上遇到上述异常，是因为 TUICallKit 依赖的 TRTC 等 SDK 部分接口使用了 JNI 实现，Android Studio 在某些条件进行编译 APK 的时候可能并不会把 Native 的 .so 库打包进去，导致出现这类报错，重新 Clean 一下即可。

> **说明：**
> 更多帮助信息，详情请参见 Android 常见问题 。

交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈： TUICallKit 产品反馈问卷 ，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群：**605115878**，进行技术交流和产品沟通。

iOS

最近更新时间：2023-12-26 17:09:43

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，跟随本文档，您将在一个小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

环境准备

iOS 9.0（API level 16）及更高。

步骤一：开通服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。
您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务：

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建出的应用，进入基本配置页面，并在页面的右下角找到含 UI 低代码场景方案功能区，单击免费体验即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



注意

IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和 **密钥(SecretKey)** 并记录下来，它们会在后续的 步骤四：登录 TUI 组件 中被用到。



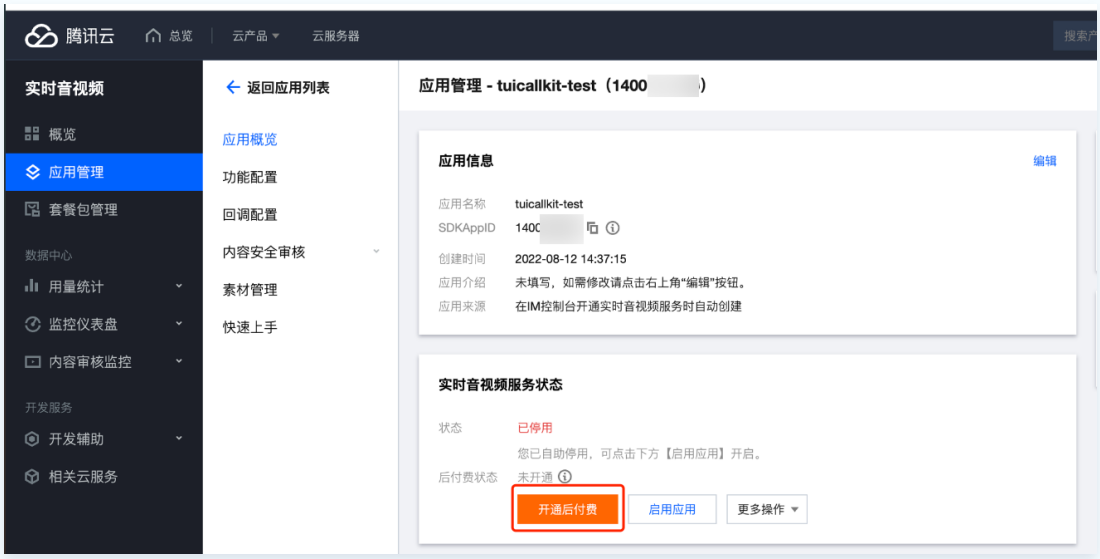
友情提示：

单击免费体验以后，部分之前使用过 实时音视频 TRTC 服务的用户会提示：

```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or the Tencent Cloud accountis in arrears
```

因为新的 IM 音视频通话能力是整合了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础的 PaaS 服务，所以当 实时音视频 TRTC 的免费额度（10000分钟）已经过期或者耗尽，就会导致开通此项服务失败，这里您可以单击 TRTC 控制台，找到对应

SDKAppID 的应用管理页，示例如图，开通后付费功能后，再次**启用应用**即可正常体验音视频通话能力。



步骤二：导入组件

从1.6版本开始，TUICallKit SDK同时支持 Objective 和 Swift，使用 CocoaPods 导入组件，具体步骤如下：

1. 在您的 `Podfile` 文件中添加以下依赖。

Objective-C

```
pod 'TUICallKit'
```

Swift

```
pod 'TUICallKit_Swift'
```

2. 执行以下命令，安装组件。

```
pod install
```

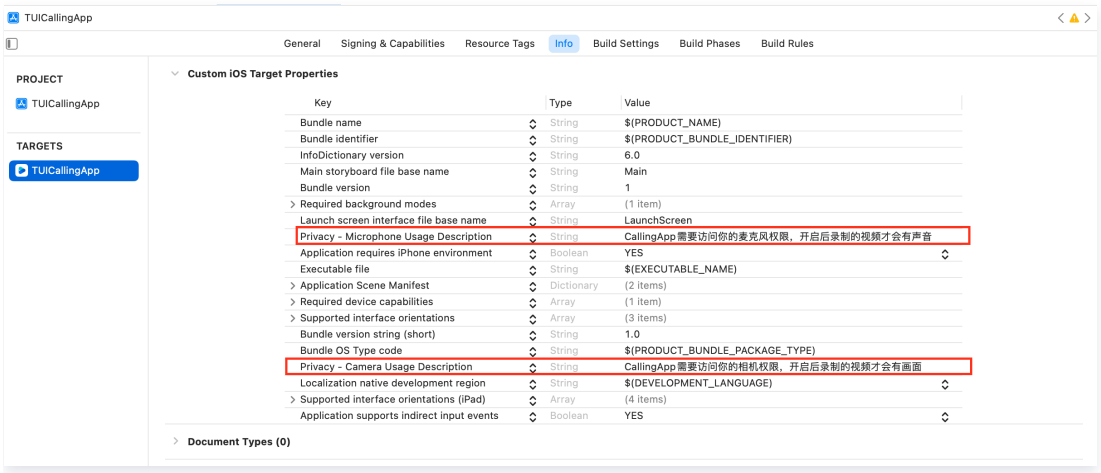
如果无法安装 TUICallKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

步骤三：完成工程配置

使用音视频功能，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。

```
<key>NSCameraUsageDescription</key>
<string>CallingApp需要访问您的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>CallingApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```



步骤四：登录 TUI 组件

在您的项目中添加如下代码，它的作用是通过调用 TUICore 中的相关接口完成 TUI 组件的登录。这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：

Objective-C

```
#import <TUICore/TUILogin.h>

// 组件登录
[TUILogin login:1400000001          // 请替换为步骤一取到的 SDKAppID
        userID:@"denny"             // 请替换为您的 UserID
        userSig:@"xxxxxxxxxx"       // 您可以在控制台中计算一个 UserSig 并填在这个位置
          succ:^{
    NSLog(@"login success");
} fail:^(int code, NSString *msg) {
    NSLog(@"login failed, code: %d, error: %@", code, msg);
}
```

Swift

```
import TUICore

// 组件登录
TUILogin.login(1400000001,                  // 请替换为步骤一取到的 SDKAppID
          userID: "denny",                  // 请替换为您的 UserID
          userSig: "xxxxxxxxxx") {          // 您可以在控制台中计算一个 UserSig 并填在这个位置
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \(code), error: \(message ?? "nil")")
}
```

参数说明：

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- sdkAppId：在步骤一中的最后一步中您已经获取到，这里不再赘述。
- userId：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- userSig：使用步骤三中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig。
- 更多信息请参见 如何计算及使用 UserSig 。

> ⚠ **注意：**
>
> - **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
>   - sdkAppId 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
>   - userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 sdkAppId、userId 以及过期时间等信息加密得来的，而不是直接把 Secretkey 配置成 userSig。
>   - userId 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userId 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userId。
> - Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 userSig。

步骤五：拨打通话

1对1视频通话

通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

Objective-C

```
#import <TUICallKit/TUICallKit.h>

// 发起1对1视频通话(假设 userId 为 mike)
[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```

Swift

```
import TUICallKit

// 发起1对1视频通话(假设 userId 为 mike)
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video)
```

参数	类型	含义
userId	String	目标用户的 UserID：`"mike"`

callMediaType	TUICallMediaType	通话的媒体类型，示例：TUICallMediaTypeVideo

群内视频通话

通过调用 TUICallKit 的 groupCall 函数并指定通话类型和被叫方的 UserID 列表，就可以发起群内的语音或者视频通话。

Objective-C

```
#import <TUICallKit/TUICallKit.h>

[[TUICallKit createInstance] groupCall:@"12345678" userIdList:@[@"denny", @"mike", @"tommy"]
callMediaType:TUICallMediaTypeVideo];
```

Swift

```
import TUICallKit

TUICallKit.createInstance().groupCall(groupId: "12345678", userIdList: ["denny", "mike", "tommy"],
callMediaType: .video)
```

参数	类型	含义
groupId	String	群组 ID，示例：@"12345678"
userIdList	Array	目标用户的userId 列表，示例：@[@"denny", @"mike", @"tommy"]
callMediaType	TUICallMediaType	通话的媒体类型，示例：TUICallMediaTypeVideo

说明

- 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。
- TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈： TUIKit 需求收集表 。

步骤六：接听通话

在 步骤四 完成后，收到来电请求后，TUICallKit 组件会自动启动相应的接听界面。

步骤七：更多特性

一、设置昵称&头像

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

Objective-C

```
[[TUICallKit createInstance] setSelfInfo:@"昵称" avatar:@"头像url" succ:^{

} fail:^(int code, NSString *errMsg) {

}];
```

Swift

```
TUICallKit.createInstance().setSelfInfo(nickname: "昵称", avatar: "头像url", succ: {

}) { code, desc in

}
```

> **注意**
> 因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

二、离线唤醒

完成以上步骤，就可以实现音视频通话的拨打和接通，但如果您的业务场景需要在 App 的进程被杀死后 或者 APP 退到后台后 ，还可以正常接收到音视频通话请求，就需要增加离线唤醒功能，详情见 离线唤醒（iOS）。

三、悬浮窗功能

如果您的业务需要开启悬浮窗功能，您可以在 TUICallKit 组件初始化时调用以下接口开启该功能：

Objective-C

```
[[TUICallKit createInstance] enableFloatWindow:YES];
```

Swift

```
TUICallKit.createInstance().enableFloatWindow(true)
```

四、通话状态监听

如果您的业务需要 **监听通话的状态**，例如通话开始、结束，以及通话过程中的网络质量等等，可以监听以下事件：

Objective-C

```
#import <TUICallEngine/TUICallEngine.h>

[[TUICallEngine createInstance] addObserver:self];

- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole {

}
```

```
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole totalTime:(float)totalTime {

}
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQualityList {

}
```

Swift

```
import TUICallEngine

TUICallEngine.createInstance().addObserver(self)

public func onCallBegin(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRole: TUICallRole) {

}
public func onCallEnd(roomId: TUIRoomId, callMediaType: TUICallMediaType, callRole: TUICallRole, totalTime:
Float) {

}
public func onUserNetworkQualityChanged(networkQualityList: [TUINetworkQualityInfo]) {

}
```

五、自定义铃音

如果您需要自定义来电铃音，可以通过如下接口进行设置：常见问题。

Objective-C

```
[[TUICallKit createInstance] setCallingBell:filePath];
```

Swift

```
TUICallKit.createInstance().setCallingBell(filePath: filePath)
```

六、自定义通话超时时间

如果您需要自定义通话的超时时间，可以在调用 call 、groupCall 、joinInGroupCall 方法的时候修改 TUICallParams 参数进行设置：

Objective-C

```
TUICallParams *callParams = [TUICallParams new];
callParams.offlinePushInfo = [TUIOfflinePushInfo new];
/// 自定义超时时间
callParams.timeout = 30;
callParams.userData = @"user data";

[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo params:callParams succ:nil
fail:nil];
```

Swift

```
let callParams = TUICallParams()
let offlinePushInfo = TUIOfflinePushInfo()
callParams.offlinePushInfo = offlinePushInfo
callParams.timeout = 30
callParams.userData = "user data"
TUICallKit.createInstance().call(userId: "mike", callMediaType: .video, params: callParams) {

}
```

常见问题

错误提示“The package you purchased does not support this ability”？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通音视频通话服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

如何购买套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请点击页面右侧，进行售前套餐咨询，或者加入 QQ 群：**592465424**，进行咨询和反馈。

交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈： TUICallKit 产品反馈问卷，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群：**605115878**，进行技术交流和产品沟通。

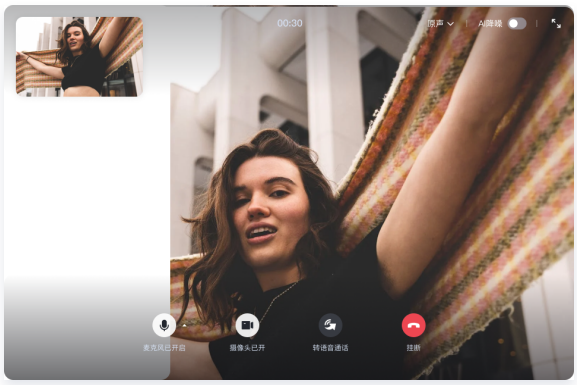
Web & H5 (Vue2/Vue3)

最近更新时间：2024-02-21 17:54:31

说明：
我们团队即将在 Web 端丰富更多不同的组件，以满足您的开发需求。
希望您能抽出几分钟时间，为您自己的需求投上一票，我们将优先开发。
问卷地址：TUICallKit Web 问卷调查 。

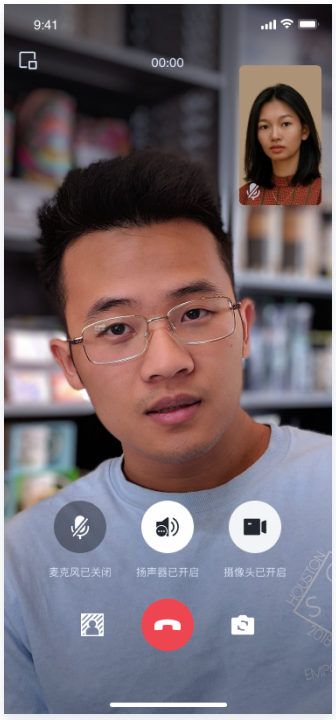
本文将介绍如何快速完成 TUICallKit 组件的接入，您将在10分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

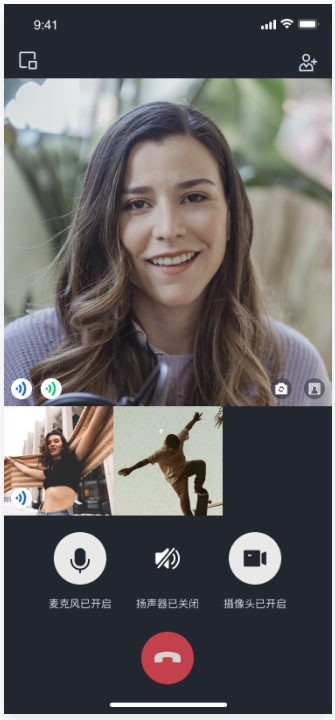
桌面端

视频通话	语音通话
	

移动端

单人通话	群组通话





TUICallKit Demo 体验

- 如果您想要直接体验音视频通话，进入 Demo 体验页面，点击 Web 卡片中的 通话 。
- 如果您想要直接跑通一个新工程，请直接阅读 Web Demo 快速跑通 。

开发环境要求

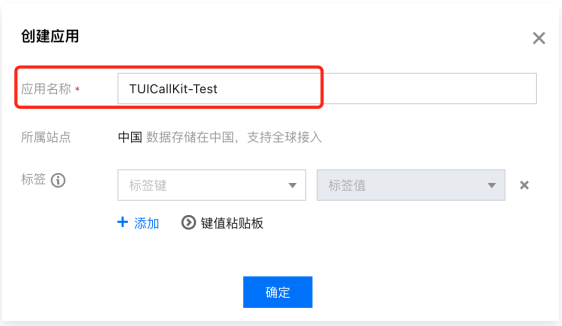
- **npm 包集成**
  - Vue3 开发环境，集成 @tencentcloud/call-uikit-vue NPM 包。
  - Vue2.7 开发环境：集成 @tencentcloud/call-uikit-vue2 NPM 包。
  - Vue2.6 开发环境：集成 @tencentcloud/call-uikit-vue2.6 NPM 包。
    - 需要：Vue2.6 + @vue/composition-api 。
- **源码集成**
  - Vue3 + TypeScript 开发环境：从 @tencentcloud/call-uikit-vue NPM 包拷贝源码。
  - Vue2.7 + TypeScript 开发环境：从 @tencentcloud/call-uikit-vue2 NPM 包拷贝源码。
  - Vue2.6 + TypeScript 开发环境：从 @tencentcloud/call-uikit-vue2.6 NPM 包拷贝源码。
    - 需要：Vue2.6 + unplugin-vue2-script-setup + @vue/composition-api 。
- **node（node.js ≥ 16.0.0）。**
- **npm（版本请与 node 版本匹配）。**

> ⚠ **注意：**
> 1. HBuilderX 中选择 vue2 时，使用的是 vue2.6，因此需要使用：@tencentcloud/call-uikit-vue2.6 。
> 2. HBuilderX 中选择 vue3 时，使用的是 vue3，因此需要使用：@tencentcloud/call-uikit-vue 。

步骤一：开通 TRTC 服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务。

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建的应用，进入**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**标题，单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。



> **注意：**
> - 如果需要**正式应用上线**，可以单击 购买正式版 即可进入购买页面。
> - IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和**密钥（SecretKey）**并记录下来，它们会在后续的步骤中被用到。



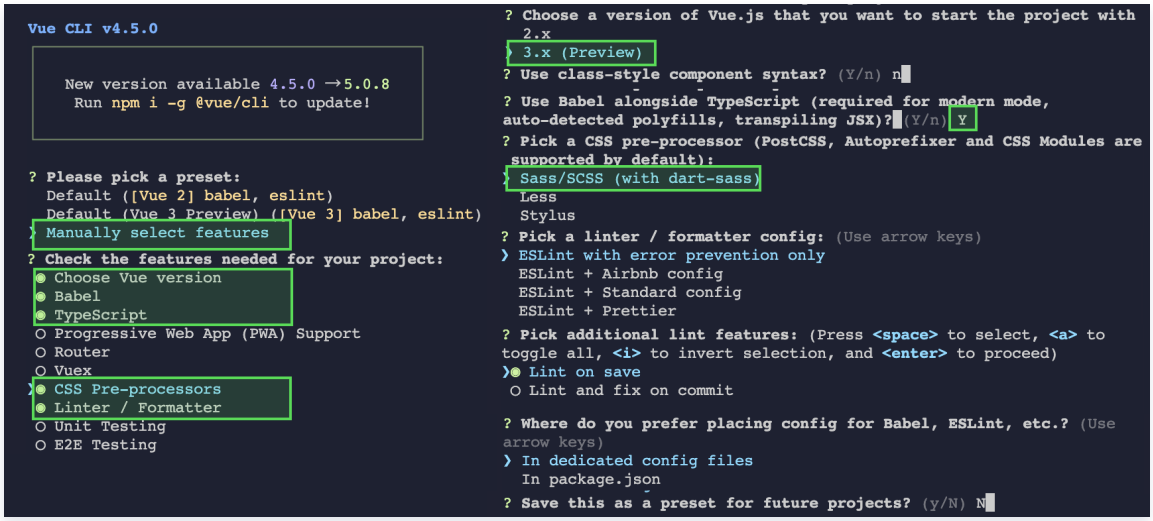
步骤二：创建 Vue3 项目

1. 如果您尚未安装 vue-cli ，可以在 terminal 或 cmd 中采用如下方式进行安装：

```
npm install -g @vue/cli
```

2. 通过 vue-cli 创建项目，并选择下图中所选配置项。

```
vue create callkit-demo
```



3. 创建完成项目，切换到项目所在目录。

```
cd callkit-demo
```

步骤三：下载 TUICallKit 组件

推荐使用 NPM 集成，如果您需要采用源码集成，请参考 TUICallKit 源码集成 。

> **说明：**
> 如果您的工程同时使用了 TUIChat UIKit，请直接移步 TUIKit 接入音视频通话 进行接入。

1. 通过 npm 方式下载 TUICallKit 组件。

Vue3

```
npm install @tencentcloud/call-uikit-vue
```

Vue2.7

```
npm install @tencentcloud/call-uikit-vue2
```

Vue2.6

```
npm install @tencentcloud/call-uikit-vue2.6 @vue/composition-api
```

如果是源码集成还需要安装 unplugin-vue2-script-setup 。

```
npm i -D unplugin-vue2-script-setup
```

2. 将 debug 目录复制到您的项目。

Vue3

MacOS 端

```
cp -r node_modules/@tencentcloud/call-uikit-vue/debug ./src
```

Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-vue\debug .\src /i /e
```

Vue2.7

MacOS 端

```
cp -r node_modules/@tencentcloud/call-uikit-vue2/debug ./src
```

Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-vue2\debug .\src /i /e
```

Vue2.6

MacOS 端

```
cp -r node_modules/@tencentcloud/call-uikit-vue2.6/debug ./src
```

Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-vue2.6\debug .\src /i /e
```

步骤四：引入 TUICallKit 组件

将下面示例代码中的 SDKAppID 、SecretKey 、userID 、callUserID 等填写。

- SDKAppID：在腾讯云创建的音视频应用 SDKAppID，参考 步骤一：开通 TRTC 服务 。
- SecretKey：用户签名，参考 步骤一：开通 TRTC 服务 。
- userID：主叫的 userID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- callUserID：被叫的 userID，需要已初始化存在。(demo 中的 callUserID，当有被叫时输入，无被叫可以不输入)

> **注意：**
> TUICallKit 组件需要放到一个 dom 节点里，用于显示并控制 TUICallKit 的位置、宽高等样式。

Vue3

将下面代码直接复制引用在 App.vue 文件中。

<template>

```
<template>
  <div style="padding: 20px 0 0 20px">
    <div><span>userID：</span><input type="text" v-model="userID" /></div>
    <div style="margin-top: 6px;"><span>callUserID：</span><input type="text" v-
model="callUserID" /></div>
    <button @click="init()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 1: Initialization
</button>
```

```
    <button @click="call()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 2: Start 1v1 video
call </button>
    <div style="width: 50rem; height: 35rem; border: 1px solid salmon;">
      <TUICallKit />
    </div>
  </div>
</template>
<script lang="ts" setup>
import { ref } from 'vue';
import { TUICallKit, TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-vue";
import * as GenerateTestUserSig from "./debug/GenerateTestUserSig-es";
// SDKAppID、SecretKey 的获取参考下面步骤
const userID = ref('');     // 主叫的 userID
const calluserID = ref(''); // 被叫的 userID
const SDKAppID = 0;    // Replace with your SDKAppID
const SecretKey = '';  // Replace with your SecretKey

async function init() {
  try {
    const { userSig } = GenerateTestUserSig.genTestUserSig({
      userID: userID.value,
      SDKAppID: SDKAppID,
      SecretKey,
    });
    // 初始化
    await TUICallKitServer.init({
      SDKAppID,
      userID: userID.value,
      userSig,
    });
    alert("[TUICallKit] Initialization success.");
  } catch (error: any) {
    alert(`[TUICallKit] Initialization failed. Reason: ${error}`);
  }
}
async function call() {
  try {
    await TUICallKitServer.call({ userID: calluserID.value, type: TUICallType.VIDEO_CALL });
  } catch (error: any) {
    alert(`[TUICallKit] Call failed. Reason: ${error}`);
  }
}
</script>
```

Vue2.7

将下面代码直接复制引用在 `App.vue` **文件中。**

<template>

```
<template>
  <div style="padding: 20px 0 0 20px">
```

```
    <div><span>userID：</span><input type="text" v-model="userID" /></div>
    <div style="margin-top: 6px;"><span>callUserID：</span><input type="text" v-
model="callUserID" /></div>
    <button @click="init()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 1: Initialization
</button>
    <button @click="call()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 2: Start 1v1 video
call </button>
    <div style="width: 50rem; height: 35rem; border: 1px solid salmon;">
      <TUICallKit />
    </div>
  </div>
</template>
<script>
import { TUICallKit, TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-vue2";

export default {
  name: 'App',
  data() {
    return {
      // SDKAppID、userSig 的获取参考下面步骤
      userID: '',    // 主叫的 userID
      callUserID: '', // 被叫的 userID
      SDKAppID: 0,    // Replace with your SDKAppID
      SecretKey: '',  // Replace with your SecretKey
    };
  },
  components: {
    TUICallKit
  },
  methods: {
    async init() {
      try {
        const { userSig } = GenerateTestUserSig.genTestUserSig({
          userID: this.userID,
          SDKAppID: Number(this.SDKAppID),
          SecretKey: this.SecretKey,
        });
        // 初始化
        await TUICallKitServer.init({
          SDKAppID: Number(this.SDKAppID),
          userID: this.userID,
          userSig,
          // tim: this.tim     // 如果工程中已有 tim 实例，需在此处传入
        });
        alert("[TUICallKit] Initialization succeeds.");
      } catch (error) {
        alert(`[TUICallKit] Initialization failed. Reason: ${error}`);
      }
    },
    async call() {
      try {
        // 1v1 video call
        await TUICallKitServer.call({ userID: this.callUserID, type: TUICallType.VIDEO_CALL });
      } catch (error) {
        alert(`[TUICallKit] Call failed. Reason: ${error}`);
      }
```

```
    }
  },
}
</script>
```

Vue2.6

1. main.ts 文件 注册 @vue/composition-api 。

main.ts

```
import Vue from 'vue'
import VueCompositionAPI from '@vue/composition-api'
Vue.use(VueCompositionAPI)
```

2. 将下面代码直接复制引用在 App.vue 文件中。

<template>

```
<template>
  <div style="padding: 20px 0 0 20px">
    <div><span>userID：</span><input type="text" v-model="userID" /></div>
    <div style="margin-top: 6px;"><span>callUserID：</span><input type="text" v-
model="callUserID" /></div>
    <button @click="init()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 1: Initialization
</button>
    <button @click="call()" style="margin: 10px 10px 10px 0; height: 28px;"> Step 2: Start 1v1 video
call </button>
    <div style="width: 50rem; height: 35rem; border: 1px solid salmon;">
      <TUICallKit />
    </div>
  </div>
</template>
<script>
import { TUICallKit, TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-vue2.6";

export default {
  name: 'App',
  data() {
    return {
      // SDKAppID、userSig 的获取参考下面步骤
      userID: '',     // 主叫的 userID
      callUserID: '', // 被叫的 userID
      SDKAppID: 0,    // Replace with your SDKAppID
      SecretKey: '',  // Replace with your SecretKey
    };
  },
  components: {
    TUICallKit
  },
```

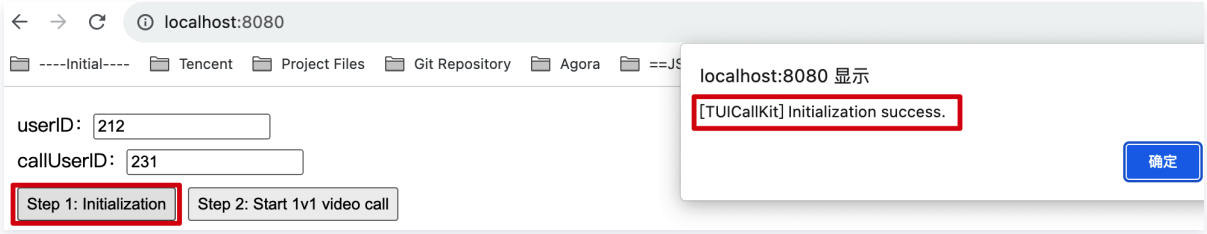
```
  methods: {
    async init() {
      try {
        const { userSig } = GenerateTestUserSig.genTestUserSig({
          userID: this.userID,
          SDKAppID: Number(this.SDKAppID),
          SecretKey: this.SecretKey,
        });
        await TUICallKitServer.init({
          SDKAppID: Number(this.SDKAppID),
          userID: this.userID,
          userSig,
          // tim: this.tim     // 如果工程中已有 tim 实例，需在此处传入
        });
        alert("[TUICallKit] Initialization succeeds.");
      } catch (error) {
        alert(`[TUICallKit] Initialization failed. Reason: ${error}`);
      }
    },
    async call() {
      try {
        // 1v1 video call
        await TUICallKitServer.call({ userID: this.callUserID, type: TUICallType.VIDEO_CALL });
      } catch (error) {
        alert(`[TUICallKit] Call failed. Reason: ${error}`);
      }
    }
  },
}
</script>
```

步骤五：拨打您的第一通电话

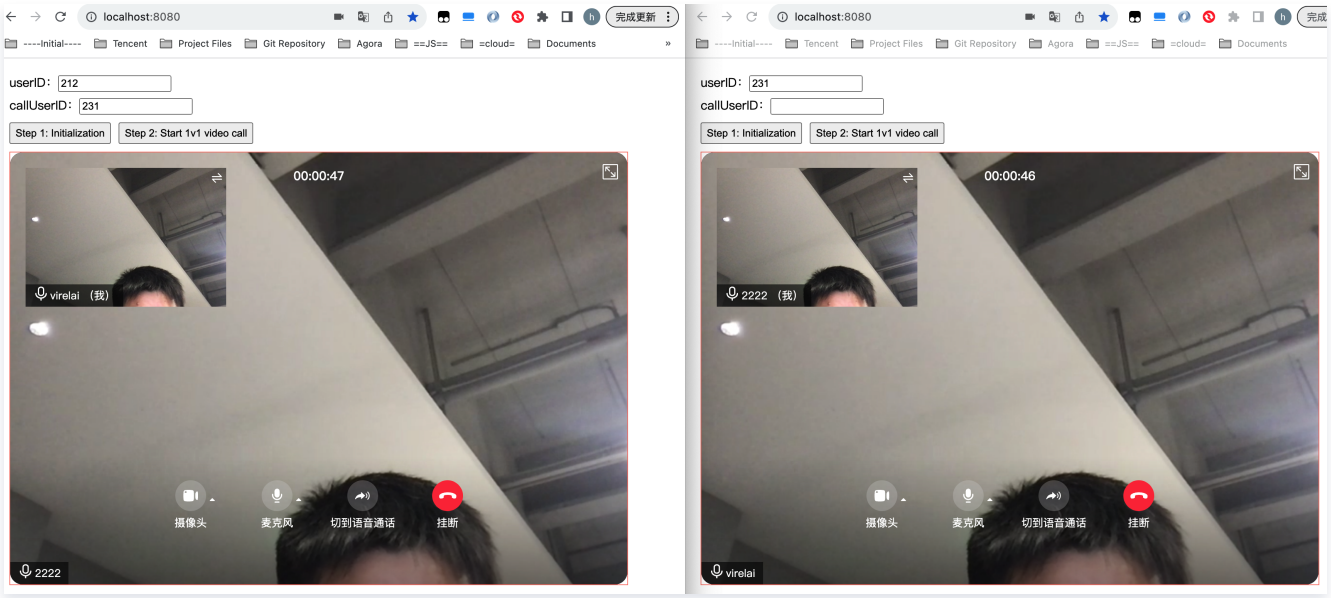
- 在终端输 `npm run serve` 运行 callkit-demo。

> **警告：本地环境请在 localhost 协议下访问，具体参考 网络访问协议说明 。**

- 点击 `Step 1: Initialization` Button，弹出 **Initialization success** 表示初始化成功。



- 点击 `Step 2: Start 1v1 video call` 发起 1v1 视频通话，具体效果如下图所示。



TUICallKit 更多特性

1、设置昵称、头像

> **说明：≥ v2.2.0 支持**

如果您需要设置当前用户的昵称或头像，可以使用如下接口进行更新。

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "jack", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] setSelfInfo failed. Reason: ${error}`);
}
```

2、自定义来电铃声

> **说明：≥ v3.0.0 支持**

- 自定义用户的来电铃音。
- 仅限传入**本地 MP3 格式**的文件地址，需要确保该文件目录是应用可以访问的。
- 铃声媒体文件必须放在 `public` 目录下引入。

```
try {
  await TUICallKitServer.setCallingBell(filePath?: string)
} catch (error: any) {
  alert(`[TUICallKit] setCallingBell failed. Reason: ${error}`);
}
```

3、群组通话

群组通话需要传入 groupID，groupID 的生成可以通过 @tencentcloud/chat 的群组接口，具体使用如下。**建议：在 IM TUIKit 的群组里发起群组通话。**

> ⚠ **注意：**
> - 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。
> - TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈： TUICallKit 产品反馈问卷 。

```
import TIM from "@tencentcloud/chat";
import { TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-vue";

const userIDList: string[] = ['user1', 'user2'];
async function groupCall() {
  const groupID: string = await createGroupID();
  try {
    await TUICallKitServer.groupCall({
      userIDList,
      groupID,
      type: TUICallType.VIDEO_CALL,
    });
  } catch (error: any) {
    alert(`[TUICallKit] groupCall failed. Reason:${error}`);
  }
}

async function createGroupID() {
  const tim = TIM.create({ SDKAppID });
  const memberList: any[] = userIDList.map(userId => ({ userID: userId }));
  const res = await tim.createGroup({
    type: TIM.TYPES.GRP_PUBLIC, // 必须是 public 群
    name: 'WebSDK',
    memberList
  });
  return res.data.group.groupID;
}
```

4、通话状态回调事件

如果您的业务需要 监听通话的状态，例如通话开始、结束，以及通话过程中的事件（详见 TUICallEvent ），具体如下：

```
import { TUICallEvent } from 'tuicall-engine-webrtc';

let handleOnCallBegin = function(event) {
    console.log(event)
};
tuiCallEngine.on(TUICallEvent.ON_CALL_BEGIN, handleOnCallBegin); // 添加监听
tuiCallEngine.off(TUICallEvent.ON_CALL_BEGIN, handleOnCallBegin); // 取消监听
```

5、悬浮窗&全屏

- 通过 `allowedMinimized` 属性可以控制悬浮窗（最小化）的开启。
- 通过 `allowedFullScreen` 属性可以控制通话窗口全屏的开启。

```
<TUICallKit
    :allowedMinimized="true"
    :allowedFullScreen="true"
```

```
/>
```

6、分辨率&填充模式

TUICallKit 组件提供了若干个功能开关，可以根据需要选择开启或关闭，具体参考 TUICallKit 属性介绍 。

- videoDisplayMode：画面显示模式。
- videoResolution：通话分辨率。

```
<TUICallKit
  :videoDisplayMode="VideoDisplayMode.CONTAIN"
  :videoResolution="VideoResolution.RESOLUTION_1080P"
/>
import { VideoDisplayMode, VideoResolution } from "@tencentcloud/call-uikit-vue";
```

7、组件回调事件

TUICallKit 组件提供了通话状态回调，可以用于业务侧实现更多交互逻辑，具体参考 TUICallKit 属性介绍 。

- beforeCalling：通话开始前会执行。
- afterCalling：通话完成后执行。
- onMinimized：组件切换最小化后会执行。
- kickedOut：该用户被踢（如多端登录下）时执行。（≥v2.3.2 支持）
- statusChanged：通话状态发生变化时执行，status 种类参考 API 文档 。（≥v2.3.2 支持）

<template>

```
<TUICallKit
  :beforeCalling="beforeCalling"
  :afterCalling="afterCalling"
  :onMinimized="onMinimized"
  :kickedOut="kickedOut"
  :statusChanged="statusChanged"
/>
```

script

```
function beforeCalling() {
  console.log("[Callkit Demo] beforeCalling");
}
function afterCalling() {
  console.log("[Callkit Demo] afterCalling");
}
function onMinimized(oldStatus: string, newStatus: string) {
  console.log("[Callkit Demo] onMinimized: " + oldStatus + " -> " + newStatus);
}
function kickedOut() {
  console.log("[Callkit Demo] kickedOut");
}
function statusChanged(args: { oldStatus: string; newStatus: string; }) {
  const { oldStatus, newStatus } = args;
  if (newStatus === STATUS.CALLING_C2C_VIDEO) {
```

```
    console.log(`[Callkit Demo] statusChanged: ${oldStatus} -> ${newStatus}`);
  }
}
```

常见问题

1. 错误提示“The package you purchased does not support this ability”？

由于您当前应用的音视频通话能力包过期或未开通，请参见 开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2. 如何购买套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请点击页面右侧，进行售前套餐咨询，或 进入 IM 社群 进行咨询和反馈。

3. 如何生成 UserSig?

UserSig 是腾讯云为其云服务设计的一种安全保护签名，是一种登录凭证，由 SDKAppID 与 SecretKey 等信息组合加密得到。

- **方式一：**控制台获取，参考 获取临时 userSig。
- **方式二：**部署临时生成脚本。

> **警告：**
> 此方式是在前端代码中配置 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功能调试**，生产环境请看方式三。

为方便初期调试，userSig 可临时使用 `GenerateTestUserSig-es.js` 中 `genTestUserSig(params)` 函数来计算，例如：

```
import { genTestUserSig } from "@tencentcloud/call-uikit-vue/debug/GenerateTestUserSig-es.js";
const { userSig } = genTestUserSig({ userID: "Alice", SDKAppID: 0, SecretKey: "YOUT_SECRETKEY" });
```

- **方式三：**正式环境使用。

  正确的 UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向项目的接口，在需要 UserSig 时由您的项目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。

4. 如何创建 userID?

- 通过 userID 与 UserSig 登录过一次，会默认创建该用户。
- 通过 即时通信 IM 控制台 进行创建和获取，单击目标应用卡片，进入应用的账号管理页面，也可创建账号并获取 userID。



5. 解决源码拷贝可能导致的报错

如果您在使 TUICallKit 组件时遇到了报错，请不要担心，大多数情况下这是由于 ESLint 和 TSConfig 配置不一致造成的。您可以查阅文档，按照要求正确配置即可。如果您需要帮助，请随时联系我们，我们将确保您能够成功地使用此组件。以下是几个常见的问题：

ESLint 报错

若 TUICallKit 与您项目的代码风格不一致导致报错，可将本组件目录屏蔽，如在项目根目录增加 .eslintignore 文件，如：

```
# .eslintignore
src/components/TUICallKit
```

TypeScript 报错

如遇 Cannot find module '../package.json' 报错，是因为 TUICallKit 内引用了 JSON 文件，可在 tsconfig.json 中添加相关配置。
其他 TSConfig 问题请参见 TSConfig Reference。

```
{
  "compilerOptions": {
    "resolveJsonModule": true
  }
}
```

6. 如何源码集成 TUICallKit?

步骤一：下载 TUICallKit 组件源码及相关插件

Vue3

```
npm install @tencentcloud/call-uikit-vue
```

Vue2.7

```
npm install @tencentcloud/call-uikit-vue2
```

Vue2.6

```
npm install @tencentcloud/call-uikit-vue2.6 @vue/composition-api
```

源码集成需要安装 unplugin-vue2-script-setup 插件。

```
npm i -D unplugin-vue2-script-setup
```

修改构建工具配置文件。

Vue Cli

```
// vue.config.js
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default

module.exports = {
  parallel: false, // disable thread-loader, which is not compactible with this plugin
  configureWebpack: {
    plugins: [
      ScriptSetup({ /* options */ }),
    ],
  },
}
```

vite

```
// vite.config.ts
import { defineConfig } from 'vite'
import { createVuePlugin as Vue2 } from 'vite-plugin-vue2'
import ScriptSetup from 'unplugin-vue2-script-setup/vite'

export default defineConfig({
  plugins: [
    Vue2(),
    ScriptSetup({ /* options */ }),
  ],
})
```

Webpack

```
// webpack.config.js
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default

module.exports = {
  /* ... */
  plugins: [
    ScriptSetup({ /* options */ }),
  ]
}
```

步骤二：源码集成 TUICallKit 组件包

将源码拷贝到自己的项目中，以拷贝到 `src/components/` 目录为例：

macOS + Vue3

```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uikit-vue/*
./src/components/TUICallKit
```

macOS + Vue2

```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uikit-vue2/*
./src/components/TUICallKit
```

Windows + Vue3

```
xcopy .\node_modules\@tencentcloud\call-uikit-vue  .\src\components\TUICallKit /i /e
```

Windows + Vue2

```
xcopy .\node_modules\@tencentcloud\call-uikit-vue2  .\src\components\TUICallKit /i /e
```

步骤三：修改 TUICallKit 引入路径

修改 TUICallKit 引入路径为本地文件引入。

> ⚠ **注意：**
> 此方法可能与您的 ESLint、TypeScript 配置冲突，若出现报错，可参考 解决源码拷贝导致的报错 。

```
import { TUICallKit, TUICallKitServer } from "./components/TUICallKit/src/index";
```

接下来请参考上面的步骤及说明，使用 TUICallKit 组件提供的功能。

TUICallKit API 文档

- TUICallKit API 文档
- TUICallEngine API 文档
- TUICallKit Vue3 Demo 快速跑通 、Vue2 Demo 快速跑通
- TUICallKit 界面定制指引
- TUICallKit (Web) 常见问题

技术咨询

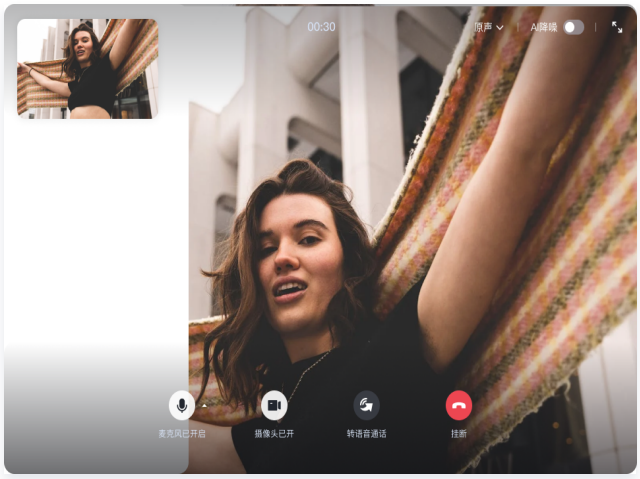
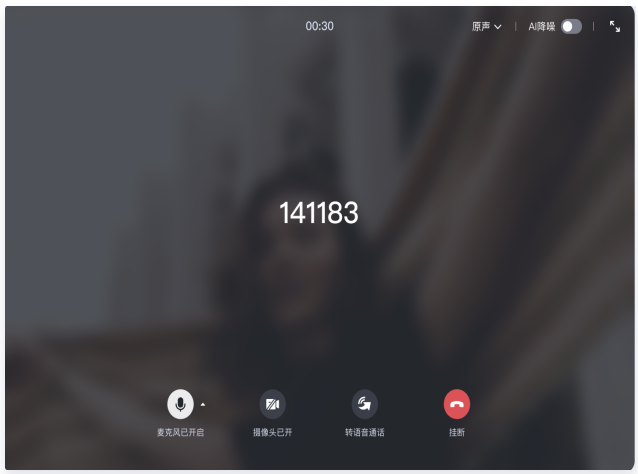
了解更多详情，请到 腾讯云通信官方社群 进行咨询和反馈。

Web（React）

最近更新时间：2024-01-03 17:30:41

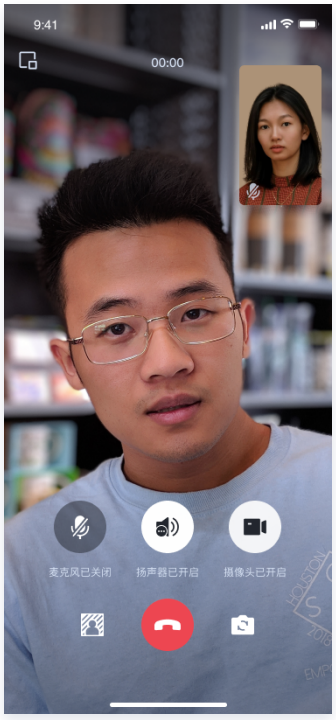
本文将介绍如何快速完成 TUICallKit 组件的接入，您将在10分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

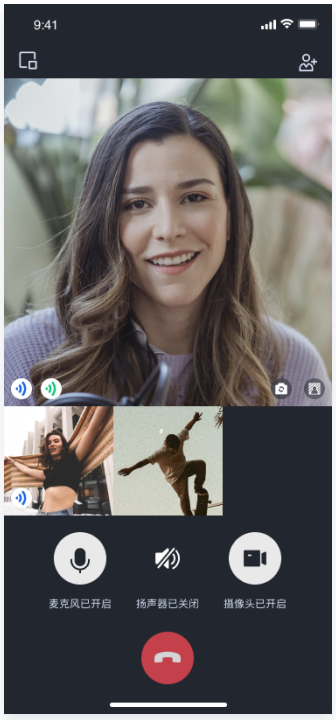
桌面端

视频通话	语音通话
	

移动端

单人通话	群组通话





TUICallKit Demo 体验

- 如果您想要直接体验音视频通话，进入 Demo 体验页面。
- 如果您想要直接跑通一个新工程，请直接阅读 Web Demo 快速跑通。

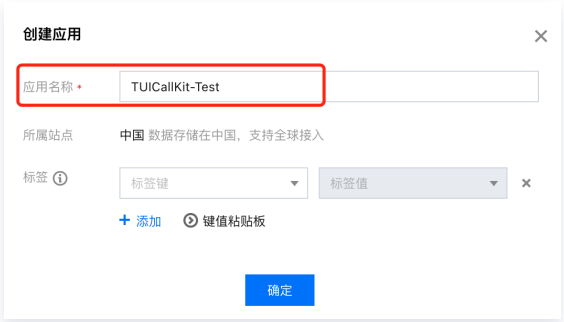
开发环境要求

- React ⩾ v18.0。
- node（12.13.0 ⩽ node 版本, 推荐使用 Node.js 官方 LTS 版本 16.17.0）。
- npm（版本请与 node 版本匹配）。

步骤一：开通 TRTC 服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务。

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建的应用，进入**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**标题，单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。



> **注意：**
> - 如果需要**正式应用上线**，可以单击 **购买正式版** 即可进入购买页面。
> - IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 **IM 购买页** 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和**密钥**（**SecretKey**）并记录下来，它们会在后续的步骤中被用到。



步骤二：创建项目

1. 创建一个新的 React 项目，您可自行选择是否需要使用 ts 模板。

```
npx create-react-app callkit-demo --template typescript
```

2. 创建项目完成后，切换到项目所在目录。

```
cd callkit-demo
```

步骤三：下载 TUICallKit 组件

1. 通过 npm 方式下载 call-uikit-react 并在项目中使用。

```
npm install @tencentcloud/call-uikit-react
```

2. 将 debug 目录复制到您的项目。

MacOS 端

```
cp -r node_modules/@tencentcloud/call-uikit-react/debug ./src
```

Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-react\debug .\src /i /e
```

步骤四：引入 TUICallKit 组件

替换 App.tsx 中的内容，或者您可以新建一个组件引入。

> **注意**
> 以下代码中未填入 **SDKAppID** 和 **SecretKey**，需在 步骤一 中获取相关信息后进行替换。

```
import React, { useState, useMemo } from "react";
// @ts-ignore
import { TUICallKit, TUICallKitServer, TUIGlobal } from "@tencentcloud/call-uikit-react";
import * as GenerateTestUserSig from "./debug/GenerateTestUserSig-es";

function App() {
  const [userData, setUserData] = useState({
    userId: "",
    callUserId: "",
    SDKAppID: 0,     // Replace with your SDKAppID
    SecretKey: "",   // Replace with your SecretKey
    isLogin: false,
  });
  const init = async () => {
    const { SDKAppID, SecretKey, userId } = userData;
    if (SDKAppID && SecretKey && userId) {
      try {
        await login(SDKAppID, SecretKey, userId);
        setUserData((prevState) => ({
          ...prevState,
          isLogin: true as boolean,
        }));
        alert("[TUICallKit] login success.");
      } catch (error) {
        console.error(error);
        alert("[TUICallKit] login failed.");
      }
    } else {
      alert("[TUICallKit] Please fill in the SDKAppID, userId, callUserId, and SecretKey.");
    }
  };
  // 完成 TUICallKit 组件的登录。这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能
  const login = async (SDKAppID: any, SecretKey: any, userId: any) => {
    const { userSig } = GenerateTestUserSig.genTestUserSig({
```

```
    userID: userId,
    SDKAppID: Number(SDKAppID),
    SecretKey: SecretKey,
  });
  await TUICallKitServer.init({
    userID: userId,
    userSig,
    SDKAppID: Number(SDKAppID),
  });
};
// 通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。
const call = async () => {
  await TUICallKitServer.call({ userID: userData.callUserId, type: 2 });
};
const callkitStyle = useMemo(() => {
  if (TUIGlobal.isPC) {
    return { width: '960px', height: '630px', margin: '0 auto' };
  }
  return { width: '100%', height: window.innerHeight };
}, [TUIGlobal.isPC]);

return (
  <>
    <TUICallKit style={callkitStyle}></TUICallKit>
    <div style={{ width: 350, margin: '0 auto' }}>
      <h4>{userData.isLogin ? "call Panel" : "Debug Panel"} </h4>
      <input
        value={userData.isLogin ? userData.callUserId : userData.userId}
        onChange={(value) => {
          const key = userData.isLogin ? "callUserId" : "userId";
          setUserData((prevState) => ({
            ...prevState,
            [key]: value.target.value,
          }));
        }}
        placeholder={
          userData.isLogin ? "Please enter userId of the call" : "Please enter the login user userId"
        }
      />
      <button onClick={userData.isLogin ? call : init}>{userData.isLogin ? "call" : "login"}</button>
    </div>
  </>
);
}

export default App;
```

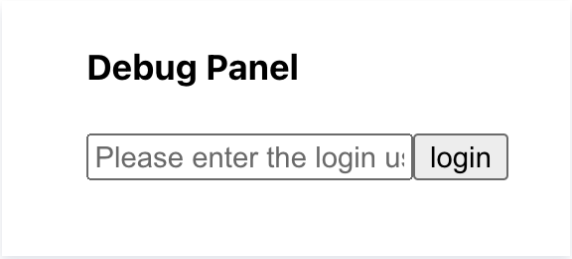
步骤五：拨打您的第一通电话

1. 在终端输 `npm run start` 运行 callkit-demo。

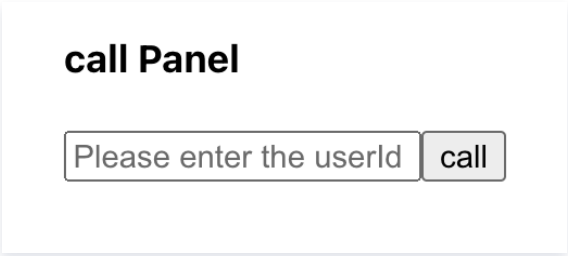
> **警告：本地环境请在 localhost 协议下访问，具体参考 网络访问协议说明 。**

```
npm run start
```

2. 输入登录的 userId，点击登录按钮，弹出 login success 表示初始化成功。



3. 输入呼叫者的 userId，点击呼叫 发起 1v1 视频通话。



TUICallKit 更多特性

1. 设置昵称、头像

如果您需要设置当前用户的昵称或头像，可以使用如下接口进行更新：

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "jack", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`);
}
```

2. 自定义来电铃声

- 自定义用户的来电铃音。
- 仅限传入**本地 MP3 格式**的文件地址，需要确保该文件目录是应用可以访问的。
- 铃声媒体文件必须放在 `public` 目录下引入。

```
try {
  await TUICallKitServer.setCallingBell(filePath?: string)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setCallingBell API. Reason: ${error}`);
}
```

3. 群组通话

群组通话需要传入 groupID，groupID 的生成可以通过 @tencentcloud/chat 的群组接口，具体使用如下。**建议：在 IM TUIKit 的群组里发起群组通话。**

> ⚠ **注意：**
> - 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

- TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈：TUIKit 产品反馈问卷 。

```
import TIM from "@tencentcloud/chat";
import { TUICallKitServer, TUICallType } from "@tencentcloud/call-uikit-react";

const userIDList: string[] = ['user1', 'user2'];
async function groupCall() {
  const groupID: string = await createGroupID();
  try {
    await TUICallKitServer.groupCall({
      userIDList,
      groupID,
      type: TUICallType.VIDEO_CALL,
    });
  } catch (error: any) {
    alert(`[TUICallKit] groupCall failed. Reason:${error}`);
  }
}

async function createGroupID() {
  const tim = TIM.create({ SDKAppID });
  const memberList: any[] = userIDList.map(userId => ({ userID: userId }));
  const res = await tim.createGroup({
    type: TIM.TYPES.GRP_PUBLIC, // 必须是 public 群
    name: 'WebSDK',
    memberList
  });
  return res.data.group.groupID;
}
```

4. 通话状态回调事件

如果您的业务需要**监听通话的状态**，例如通话开始、结束，以及通话过程中的事件（详见 TUICallEvent ），可以监听以下事件：

```
import { TUICallEvent } from 'tuicall-engine-webrtc';
TUICallKitServer.getTUICallEngineInstance().on(TUICallEvent.ON_CALL_BEGIN, handleOnCallBegin);
let handleOnCallBegin = function(event) {
    console.log(event)
};
```

5. 悬浮窗&全屏

- 通过 `allowedMinimized` 属性可以控制悬浮窗（最小化）的开启。
- 通过 `allowedFullScreen` 属性可以控制通话窗口全屏的开启。

```
<TUICallKit
    allowedMinimized={true}
    allowedFullScree={true}
/>
```

5. 分辨率&填充模式

TUICallKit 组件提供了若干个功能开关，可以根据需要选择开启或关闭，具体参考 TUICallKit 属性介绍 。

- `videoDisplayMode`：画面显示模式。
- `videoResolution`：通话分辨率。

```
import { VideoDisplayMode, VideoResolution } from "@tencentcloud/call-uikit-react";

<TUICallKit
  videoDisplayMode={VideoDisplayMode.CONTAIN}
  videoResolution={VideoResolution.RESOLUTION_1080P}
/>
```

6. 组件回调事件

TUICallKit 组件提供了组件状态回调，可以用于业务侧实现更多交互逻辑，具体参考 TUICallKit 属性介绍 。

- `beforeCalling`：通话开始前会执行。
- `afterCalling`：通话完成后执行。

```
<template>

<TUICallKit
  beforeCalling={handleBeforeCalling}
  afterCalling={handleAfterCalling}
/>
```

TUICallKit API 文档

- TUICallKit API 文档
- TUICallEngine API 文档
- TUICallKit (Web) 常见问题

常见问题

1. 错误提示“The package you purchased does not support this ability”？

由于您当前应用的音视频通话能力包过期或未开通，请参见 开通服务 ，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2. 如何购买套餐?

请参考购买链接 音视频通话 SDK 价格总览 ，如有其他问题，请点击页面右侧，进行售前套餐咨询，或 进入 IM 社群 进行咨询和反馈。

3. 如何创建 userID?

- 通过 userID 与 userSig 登录过一次，会默认创建该用户。
- 通过 即时通信 IM 控制台 进行创建和获取，单击目标应用卡片，进入应用的账号管理页面，也可创建账号并获取 userID。



技术咨询

了解更多详情您可 **腾讯云通信官方社群** 进行咨询和反馈。

uni-app（客户端）

最近更新时间：2023-12-29 17:31:22

本文将介绍如何快速完成 TUICallKit 组件的接入，您将在10分钟内完成以下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。



TUICallKit Demo 体验

- TUICallKit 插件地址：TUICallKit 插件链接。
- 如果您想要直接跑通一个新工程，请直接阅读 uni-app Demo 快速跑通。

开发环境要求

- 开发工具说明：建议使用最新的 HBuilderX 编辑器。
- 插件调试说明：原生插件暂不支持模拟器调试。
- iOS 设备要求：iOS 系统 ≥ 9.0，支持音视频通话的真机设备。
- Android 设备要求：Android 系统 ≥ 5.0（SDK API Level 21），支持音视频的真机设备，允许 USB 调试。

步骤一：开通 TRTC 服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务。

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建的应用，进入**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**标题，单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。

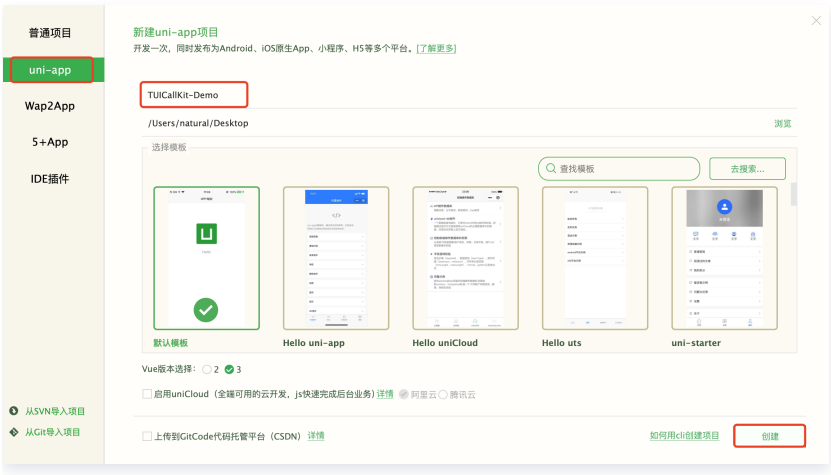


3. 在同一页面找到 SDKAppID 和密钥(SecretKey)并记录下来，会在登录 TUICallKit 组件 时用到。



步骤二：创建 uni-app 项目

打开 Hbuilderx 开发工具，点击新建 uni-app 项目：项目名称 (TUICallKit-Demo)。



步骤三：下载并导入 TUICallKit 插件

1. 访问 TencentCloud-TUICallKit 插件，在插件详情页中购买插件，购买插件时选择对应的 AppID，绑定正确的包名。



2. 在 TUICallKit-Demo 项目 中导入插件。



步骤四：使用 TUICallKit 插件

1. 在 TUICallKit-Demo/pages/index/index.vue 中引入下面代码，该段代码功能：登录 TUICallKit 组件 、拨打 1v1 视频通话 。

```
<template>
  <view class="container">
    <input type="text" v-model="inputID" :placeholder=" isLogin ? '请输入呼叫者 userID' : '请输入登录
userID' " />
    <text v-show="isLogin"> 您的 userID: {{ userID }} </text>
    <button v-show="!isLogin" @click="handleLogin"> 登录 </button>
    <button v-show="isLogin" @click="handleCall"> 拨打通话 </button>
  </view>
</template>
<script>
  const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit'); //【1】引入 TUICallKit 插件
  uni.$TUICallKit = TUICallKit;
  import { genTestUserSig } from '../../debug/GenerateTestUserSig.js'
  export default {
    data() {
      return {
        inputID: '',
        isLogin: false,
        userID: '',
      }
    },
    methods: {
```

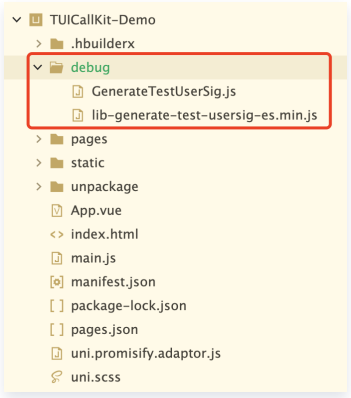
```
        handleLogin() {
            this.userID = this.inputID;
            const { userSig, sdkAppID: SDKAppID } = genTestUserSig(this.userID); // 控制台生成 userSig
            const loginParams = { SDKAppID, userID: this.userID, userSig };   // 使用 SDKAppID、userSig
            // 【2】登录 TUICallKit 组件
            uni.$TUICallKit.login( loginParams, res => {
                if (res.code === 0) {
                    this.isLogin = true;
                    this.inputID = '';
                    console.log('[TUICallKit] login success.');
                } else {
                    console.error('[TUICallKit] login failed, failed message = ', res.msg, params);
                }
            }
            );
        },
        handleCall() {
            try {
                const callParams = {
                    userID: this.inputID,
                    callMediaType: 2,   // 1 -- 语音通话，2 -- 视频通话
                    callParams: { roomID: 234, strRoomID: '2323423', timeout:30 },
                };
                // 【3】拨打 1v1 视频通话通话
                uni.$TUICallKit.call( callParams, res => {
                    console.log('[TUICallKit] call params: ', JSON.stringify(res));
                }
                );
                this.inputID = '';
            } catch (error) {
                console.log('[TUICallKit] call error: ', error);
            }
        }
    }
}
</script>
<style>
.container {
    margin: 30px;
}
.container input {
    height: 50px;
    border: 1px solid;
}
.container button {
    margin-top: 30px;
}
</style>
```

2. 填写 SDKAppID、SecretKey、userSig 参数。

客户端生成 userSig

由于 UserSig 有时效性，**如果您需要长期使用 TUICallKit，推荐您采用该方法。**

1. 单击下载 debug 文件夹，将 debug 目录复制到您的项目，如下图所示：



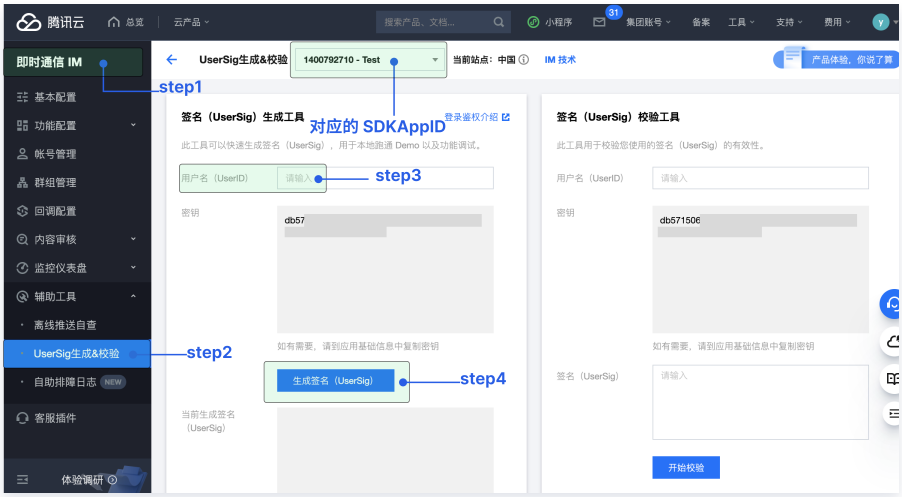
2. 填写 TUICallKit-Demo/debug/GenerateTestUserSig.js 文件的 SDKAppID 、 SecretKey （参考 步骤一：开通 TRTC 服务 ）

GenerateTestUserSig.js

```
import LibGenerateTestUserSig from './lib-generate-test-usersig-es.min.js';
/** ... */

const SDKAPPID = 0;
/** ... */

const EXPIRETIME = 604800;
/** ... */

const SECRETKEY = '';
/* ... */

export function genTestUserSig(userID) {
  const generator = new LibGenerateTestUserSig(SDKAPPID, SECRETKEY, EXPIRETIME);
  const userSig = generator.genTestUserSig(userID);

  return {
    sdkAppID: SDKAPPID,
    userSig,
  };
}
```

控制台生成 userSig

如果您想要快速体验 TUICallKit，您可以通过控制台中的 辅助工具 生成一个临时可用的 UserSig。



如果您采用的是**控制台生成**，那么需要在 TUICallKit-Demo/pages/index/index.vue 文件中，赋值这里的 SDKAppID、userSig 。



步骤五：拨打您的第一通电话

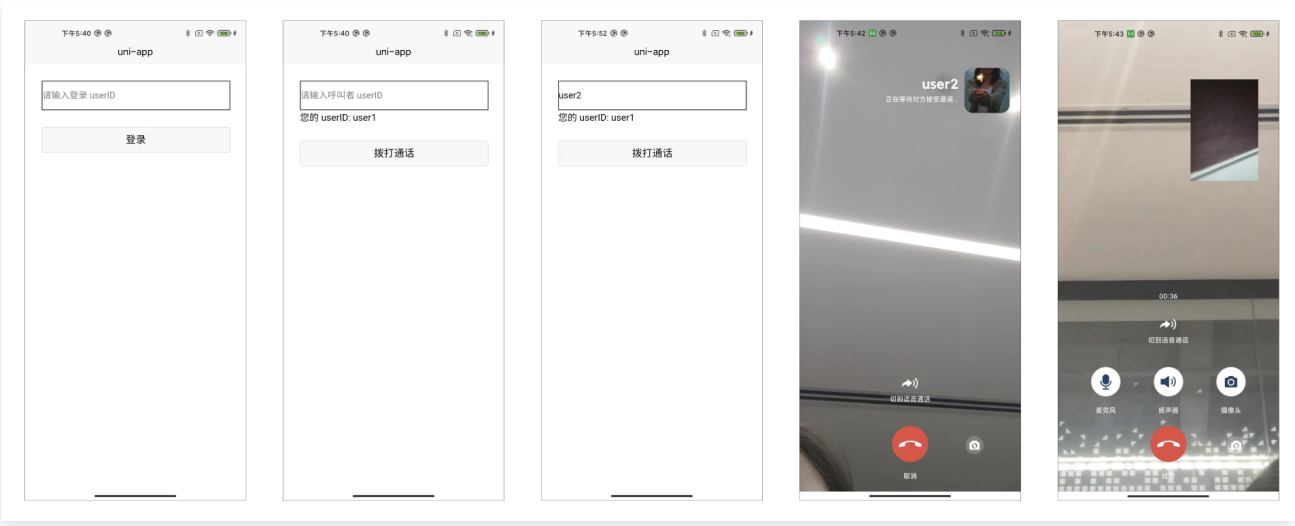
1. 制作自定义调试基座，请选择**传统打包**方式进行打包。



2. 自定义调试基座成功后，**使用自定义基座运行**项目。



3. 拨打 1v1 视频通话具体效果如图所示。



更多特性

1. 群组视频通话

通过调用 TUICallKit 的 groupCall API 并指定通话类型和被叫方的 userID，就可以发起群内的视频或语音通话。

```
const options = {
    groupID: 'xxx',   // 建议在群组里发起群通话，此时 groupID 就是群组 ID
    userIDList: ['mike', 'tom'],
    callMediaType: 2, // 语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
};
TUICallKit.groupCall(options, (res) => {
    if (res.code === 0) {
        console.log('groupCall success');
    } else {
        console.log(`groupCall failed, error message = ${res.msg}`);
    }
});
```

2. 设置昵称&头像

如果您需要自定义昵称或头像，可以使用 setSelfInfo API 进行更新。

```
const options = {
    nickName: 'jack',
    avatar: 'https:/****/user_avatar.png'
}
TUICallKit.setSelfInfo(options, (res) => {
    if (res.code === 0) {
        console.log('setSelfInfo success');
    } else {
        console.log(`setSelfInfo failed, error message = ${res.msg}`);
    }
});
```

⚠ 注意：

因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

3. 悬浮窗功能

如果您的业务需要开启悬浮窗功能，您可以在 TUICallKit 组件初始化时调用 enableFloatWindow API 开启该功能。

```
TUICallKit.enableFloatWindow(true);
```

4. 通话状态监听

如果您的业务需要**监听通话的状态**，可以通过监听 globalEvent 的事件实现。

```
const TUICallKitEvent = uni.requireNativePlugin('globalEvent');

function handleError(res) {
    console.log('onError', JSON.stringify(res);
}
TUICallKitEvent.addEventListener('onError', handleError);
TUICallKitEvent.removeEventListener('onError', handleError);
```

5. 自定义铃音

如果您需要自定义来电铃音，可以通过 setCallingBell API 进行设置。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');

// 【1】通过 uni.saveFile 保存音频文件到本地，具体参考 saveFile 接口:
https://zh.uniapp.dcloud.io/api/file/file.html#savefile
const tempFilePath = './static/rain.mp3'; // 本地存放的音频文件
let musicFilePath = '';
uni.saveFile({
  tempFilePath: tempFilePath,
  success: (res) => {
    console.warn('保存文件成功 = ', JSON.stringify(res)); // 获取的是相对路径
    musicFilePath =  res.savedFilePath;

    // 【2】相对路径转绝对路径，否则访问不到
    musicFilePath = plus.io.convertLocalFileSystemURL(musicFilePath); // 转绝对路径

    // 【3】设置铃声
    TUICallKit.setCallingBell(musicFilePath, (res) => {
      if (res.code === 0) {
        console.log('setCallingBell success');
      } else {
        console.log(`setCallingBell failed, error message = ${res.msg}`);
      }
    });
  },
  fail: (err) => {
    console.error('保存文件失败');
  },
});
```

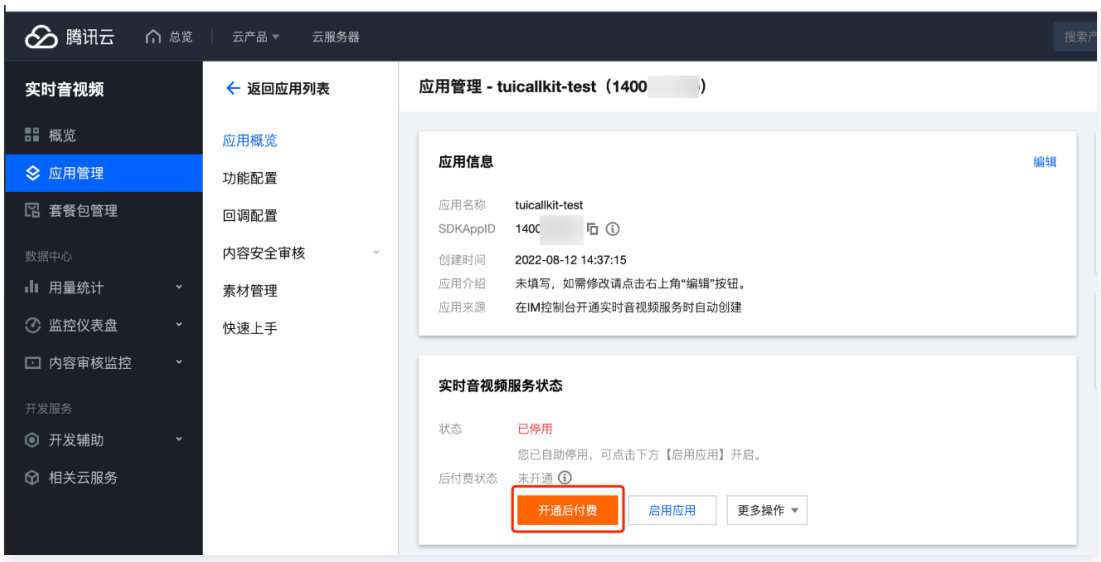
常见问题

提示“[-100013]:TRTC service is suspended...”

单击**免费体验**以后，部分之前使用过 实时音视频 TRTC 服务的用户会提示：

```
[-100013]:TRTC service is suspended. Please check if the package balance is 0 or the Tencent Cloud accountis in arrears
```

因为新的 IM 音视频通话能力是整合了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础的 PaaS 服务，所以当 实时音视频 TRTC 的免费额度（10000分钟）已经过期或者耗尽，就会导致开通此项服务失败，这里您可以单击 TRTC 控制台，找到对应 SDKAppID 的应用管理页，示例如图，开通付费功能后，再次**启用应用**即可正常体验音视频通话能力。



SDKAppID、sercretKey 参数说明

- **SDKAppID**：在 步骤一：开通 TRTC 服务 中的最后一步中您已经获取到，这里不再赘述。
- **userID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **userSig**：使用 步骤一：开通 TRTC 服务 中获取的 SecretKey 对 SDKAppID、userID 等信息进行加密，就可以得到 userSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 userSig。
- 更多信息请参见 如何计算及使用 userSig 。

> ⚠ **注意**
>
> - **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
> - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
> - userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 SDKAppID、userID 以及过期时间等信息加密得来的，而不是直接把 Secretkey 配置成 userSig。
> - userID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userID。
> - Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 userSig。

自定义调试基座注意事项

- 什么是自定义基座：参考 uni-app 官网自定义基座说明 。
- 自定义基座不是正式版，真正发布时，需要再打正式包。使用自定义基座是无法正常升级替换 APK 的。
- 请尽量不要使用本地插件，插件包超过自定义基座的限制，可能导致调试收费。
- 使用自定义基座打包 uni 原生插件（**请使用真机运行自定义基座**）
- 使用 uni 原生插件必须先提交云端打包才能生效，购买插件后在应用的 `manifest.json` 页面的 **App原生插件配置** 项下单击**选择云端插件**，选择**腾讯云原生音视频插件**。
- 直接云端打包后无法打 log，无法排查问题，需要自定义基座调试原生插件。

如何创建群组通话的 groupID

群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

相关案例-在线客服场景

我们提供了**在线客服场景**的相关源码，建议您 下载在线客服场景 并集成体验。该场景提供了示例客服群 + 示例好友的基础模板，实现功能包括：

- 支持发送文本消息、图片消息、语音消息、视频消息等常见消息。
- 支持双人语音、视频通话功能。
- 支持创建群聊会话、群成员管理等。



TUICallKit API 文档

- 快速跑通 Demo (uni-app)
- 快速集成 TUIKit 到项目
- TUICallKit API 文档
- TUICallEngine API 文档

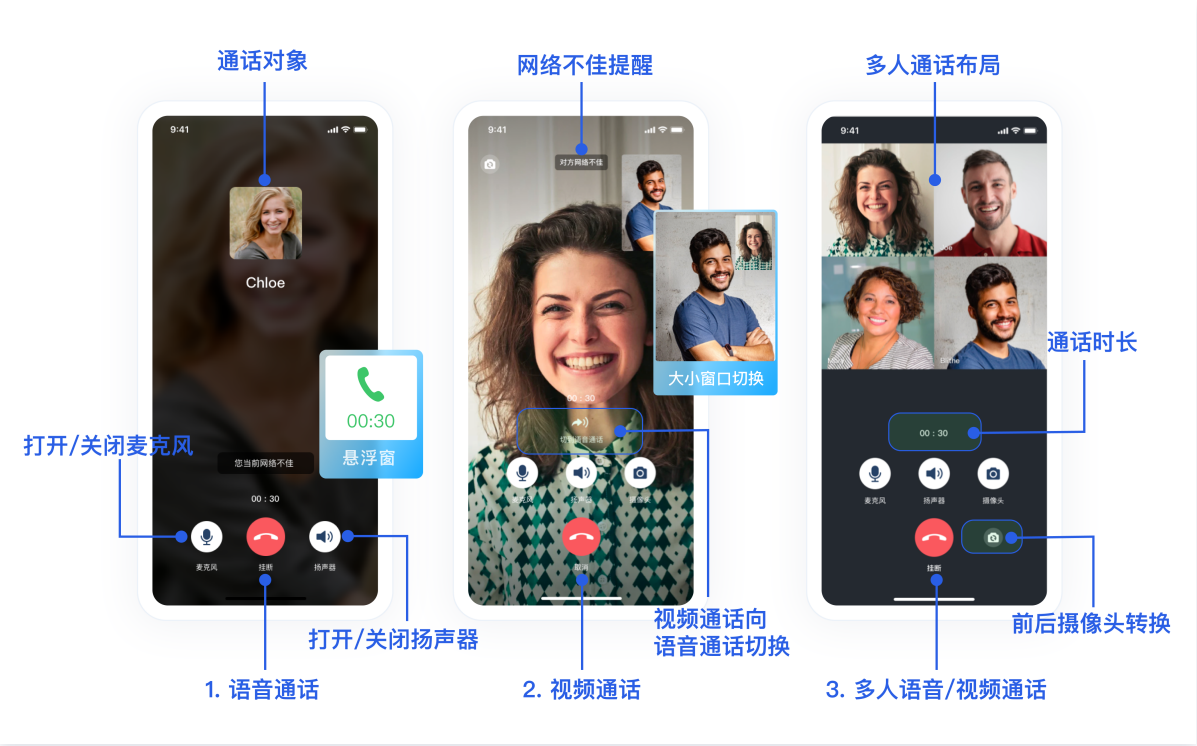
技术咨询

了解更多详情您可加入 腾讯云通信官方社群 进行咨询和反馈。

uni-app（小程序）

最近更新时间：2024-01-24 10:23:11

本文将介绍如何快速完成 TUICallKit 组件的接入，跟随本文档，您将在半小时内得到一个包含完备 UI 界面的视频通话小程序。基本功能如下图所示：



注意：

1. @tencentcloud/call-uikit-wechat ≥ **V3.0.4** 采用新架构进行重构，不再使用微信原生进行开发，而是采用 Vue（**支持 Vue2.6、Vue3**）开发的组件。
2. 如果想使用 wxcomponents 的方式集成组件，需要低于 V3.0.4。低版本的组件会针对发现的 bug 继续升级更新维护。如果您想继续使用小于 **v3.0.4 版本**来集成 TUICallKit，请移步 旧版文档 。

小程序 Demo 体验

- 如果您想要直接体验音视频通话小程序，单击 Demo 体验，扫描小程序二维码。
- 如果您想要直接跑通一个新工程，请直接阅读 uni-app demo 快速跑通 。
- 如果您想要**亲自集成** TUICallKit 组件，搭建一个音视频通话小程序，请跟随本文档。

开发环境要求

- 微信 App iOS 最低版本要求：8.0.40。
- 微信 App Android 最低版本要求：8.0.40。
- 小程序基础库最低版本要求：2.10.0。

警告：

- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，请使用**企业小程序账号**申请相关权限进行开发。
- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在**真机**上进行运行体验。

版权所有：腾讯云计算（北京）有限责任公司

小程序开发准备

步骤一：开通企业类小程序

小程序推拉流标签不支持个人小程序，只支持企业类小程序。需要在 注册 时填写**主体类型为企业**，如下图所示：



步骤二：在小程序控制台开启实时音视频接口

- 小程序推拉流标签使用权限暂时只开放给有限类目，具体支持类目参考该地址 。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限。



TUICallKit 源码集成

步骤一：开通 TRTC 服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务。

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建的应用，进入**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**标题，单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。



> **注意：**
> - 如果需要**正式应用上线**，可以单击 购买正式版 即可进入购买页面。
> - IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和**密钥**（**SecretKey**）并记录下来，它们会在后续的步骤中被用到。



步骤二：创建 uni-app 小程序项目

1. 在 HBuilder 中创建小程序项目。



2. 在终端输入 `npm init -y` ，创建 `package.json` 文件。

```
npm init -y
```

步骤三：下载并导入 TUICallKit 组件

MacOS 端

```
npm i @tencentcloud/call-uikit-wechat
```

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wechat/ ./TUICallKit
```

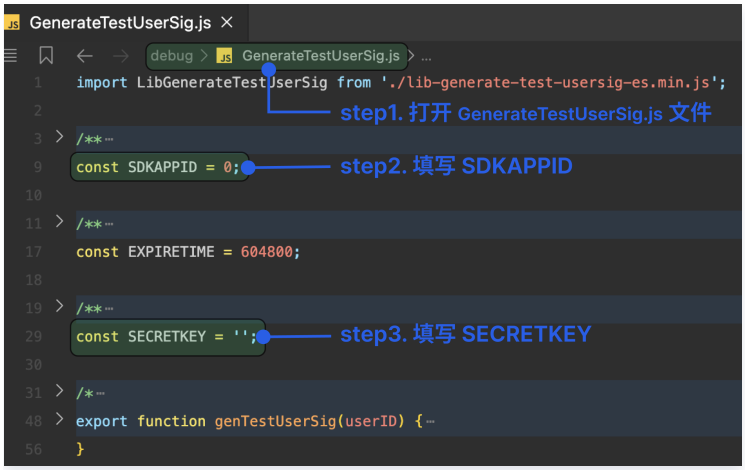
Windows 端

```
npm i @tencentcloud/call-uikit-wechat
```

```
xcopy node_modules\@tencentcloud\call-uikit-wechat\ .\TUICallKit /i /e
```

步骤四：填写 SDKAPPID 以及 SECRETKEY

修改 TUICallKit/debug/ `GenerateTestUserSig-es.js` 文件 的 SDKAPPID 以及 SECRETKEY。



步骤五：调用 TUICallKit 组件

说明：

下面会重点介绍各种接入方式包体积占比，请针对自身项目，选择合适的接入方式。

接入方式	主包接入	分包接入
能否使用全局监听	是	否
是否占用主包体积	是	否
Vue2 项目体积占比	约占用 1.3MB 主包体积，其中 TUICallKit/debug 目录用于生成 userSig，生产环境推荐在服务端生成，可以节省约 150kb 主包体积。	不会占用任何主包体积。
Vue3 项目体积占比	约占用 1.3MB 主包体积，其中 TUICallKit/debug 目录用于生成 userSig，生产环境推荐在服务端生成，可以节省约 150kb 主包体积。	参见 体积优化文档，不会占用任何主包体积。

全局监听：使小程序具有所有页面都能唤起通话页面的能力。

主包接入

注意：

主包接入的方式默认集成全局监听

在 pages.json 文件注册全局监听页面。

```
{
  "path": "TUICallKit/src/Components/TUICallKit",
  "style": {
    "navigationBarTitleText": "uni-app"
  }
}
```

Vue3

1. 修改 **pages/index/index.vue** 文件。

```
<template>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
    <view class="login">
      <button
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
</template>
<script setup>
import { ref } from "vue";
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-es.js";
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/src/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/src/TUICallService/serve/callManager";
uni.CallManager = new CallManager();
let userID = ref("");
let isLogin = ref(false);
const loginHandler = async () => {
  if (!userID.value) return;
  const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
    userID: userID.value,
  });
  await uni.CallManager.init({
    sdkAppID: SDKAppID,
    userID: userID.value,
    userSig: userSig,
    globalCallPagePath: "TUICallKit/src/Components/TUICallKit",
  });
  isLogin.value = true;
  userID.value = "";
};

const callHandler = async () => {
  await TUICallKitServer.call({
    userID: userID.value,
    type: 2,
  });
};
</script>
```

```
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input {
  display: flex;
  font-size: 20px;
}

.login {
  width: 100vw;
  bottom: 5vh;
  margin: 70rpx;
}

.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

Vue2

1. 下载依赖。

```
npm i unplugin-vue2-script-setup
```

2. 修改 `vue.config.js` 文件。

```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
    config.plugins.delete('fork-ts-checker');
  },
};
```

3. 修改 main.js 文件。

```
import vueComposition from "@vue/composition-api"
Vue.use(vueComposition)
```

4. 修改 **pages/index/index.vue** 文件。

```
<template>
<view>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
    <view class="login">
      <button
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
</view>
</template>
<script>
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-es.js";
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/src/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/src/TUICallService/serve/callManager";
uni.CallManager = new CallManager();
export default {
  data() {
    return {
      isLogin: false,
      userID: "",
    };
  },
  methods: {
    async loginHandler() {
      if (!this.userID) return;
      const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
        userID: this.userID,
      });
      await uni.CallManager.init({
        sdkAppID: SDKAppID,
        userID: this.userID,
        userSig: userSig,
        globalCallPagePath: "TUICallKit/src/Components/TUICallKit",
      });
      this.isLogin = true;
```

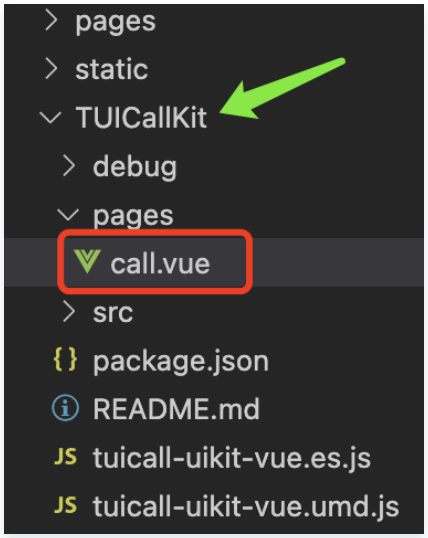
```
      this.userID = "";
    },

    async callHandler() {
      await TUICallKitServer.call({
        userID: this.userID,
        type: 2,
      });
    },
  },
};
</script>
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input {
  display: flex;
  font-size: 20px;
}

.login {
  width: 100vw;
  bottom: 5vh;
  margin: 70rpx;
}

.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

分包接入

1. 修改 manifest.json 小程序相关配置。

```
"mp-weixin": {
    "usingComponents": true,
    "optimization": {
        "subPackages": true
    }
},
```

2. 新建 /TUICallkit/pages/call.vue 目录。



3. 修改 pages.json，添加分包页面。

```
"subPackages": [
    {
     "root": "TUICallKit",
     "pages": [
      {
       "path": "pages/call",
       "style": {
        "navigationBarTitleText": "uni-app"
       }
      }
     ]
    }
],
```

4. 集成 TUICallKit。

Vue3 环境

修改 **TUICallKit/pages/call.vue** 文件。

```
<template>
  <view>
    <TUICallKit></TUICallKit>
    <view class="loginBox">
      <input
        class="input-box"
        v-model="userID"
        :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
        placeholder-style="color:#BBBBBB;"
      />
      <view class="login">
        <button
```

```
        class="loginBtn"
        @click="!isLogin ? loginHandler() : callHandler()"
      >
        {{ !isLogin ? "登录" : "呼叫" }}
      </button>
    </view>
  </view>
 </view>
</template>
<script setup>
import { ref } from "vue";
import * as GenerateTestUserSig from "../debug/GenerateTestUserSig-es.js";
import { TUICallKitServer } from "../src/index";
import TUICallKit from "../src/Components/TUICallKit";
let userID = ref("");
let isLogin = ref(false);
const loginHandler = async () => {
 if (!userID.value) return;
 const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
  userID: userID.value,
 });
 await TUICallKitServer.init({
  sdkAppID: SDKAppID,
  userID: userID.value,
  userSig: userSig,
 });
 isLogin.value = true;
 userID.value = "";
};

const callHandler = async () => {
 await TUICallKitServer.call({
  userID: userID.value,
  type: 2,
 });
};
</script>
<style>
.loginBox {
 margin-top: 200px;
 display: flex;
 flex-direction: column;
 align-items: center;
 justify-content: center;
}

input {
 display: flex;
 font-size: 20px;
}

.login {
 width: 100vw;
 bottom: 5vh;
 margin: 70rpx;
}
```

```
.login button {
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
</style>
```

Vue2 环境

1. 下载依赖。

```
npm i unplugin-vue2-script-setup
```

2. 修改 `vue.config.js` 文件。

```
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
  parallel: false,
  configureWebpack: {
    plugins: [
      ScriptSetup({
        /* options */
      }),
    ],
  },
  chainWebpack(config) {
    config.plugins.delete('fork-ts-checker');
  },
};
```

3. 修改 `main.js` 文件。

```
import vueComposition from "@vue/composition-api"
Vue.use(vueComposition)
```

4. 修改 **TUICallKit/pages/call.vue** 文件。

```
<template>
<view>
  <TUICallKit></TUICallKit>
  <view class="loginBox">
    <input
      class="input-box"
      v-model="userID"
      :placeholder="!isLogin ? '请输入用户ID' : '搜索用户ID'"
      placeholder-style="color:#BBBBBB;"
    />
```

```
      <view class="login">
        <button
          class="loginBtn"
          @click="!isLogin ? loginHandler() : callHandler()"
        >
          {{ !isLogin ? "登录" : "呼叫" }}
        </button>
      </view>
    </view>
  </view>
</template>
<script>
import * as GenerateTestUserSig from "../debug/GenerateTestUserSig-es.js";
import { TUICallKitServer } from "../src/index";
import TUICallKit from "../src/Components/TUICallKit.vue";
export default {
  components: {
    TUICallKit,
  },
  data() {
    return {
      isLogin: false,
      userID: "",
    };
  },
  methods: {
    async loginHandler() {
      if (!this.userID) return;
      const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
        userID: this.userID,
      });
      await TUICallKitServer.init({
        sdkAppID: SDKAppID,
        userID: this.userID,
        userSig: userSig,
      });
      this.isLogin = true;
      this.userID = "";
    },

    async callHandler() {
      await TUICallKitServer.call({
        userID: this.userID,
        type: 2,
      });
    },
  },
};
</script>
<style>
.loginBox {
  margin-top: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
```

```
  }

  input {
    display: flex;
    font-size: 20px;
  }

  .login {
    width: 100vw;
    bottom: 5vh;
    margin: 70rpx;
  }

  .login button {
    width: 80%;
    background-color: #006eff;
    border-radius: 50px;
    color: white;
  }
  </style>
```

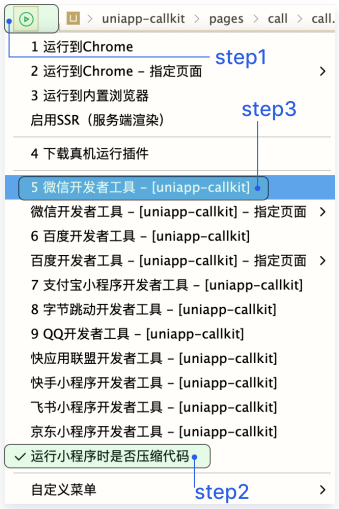
5. 修改 pages/index/index.vue 文件。

> **说明：**
> 此处的代码的作用是，主包路由跳转到 TUICallKt 分包页面，注意需要根据您的业务进行调整，此处以空项目举例。

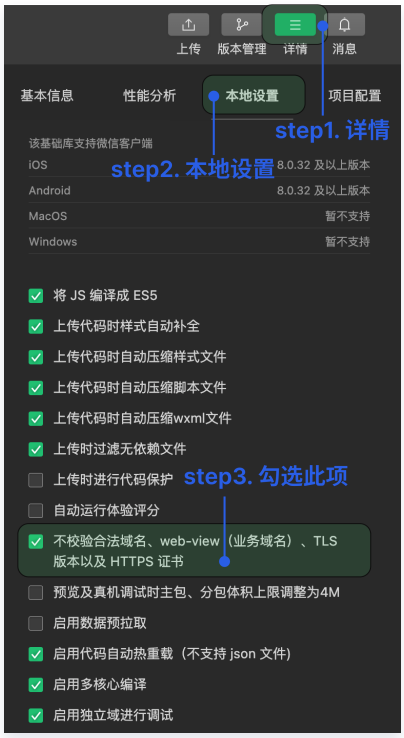
```
<template>
  <view>
    <button @click="jumpUrl">跳转分包</button>
  </view>
</template>

<script>
export default {
  methods: {
    jumpUrl() {
      uni.navigateTo({ url: "/TUICallKit/pages/call" });
    },
  },
};
</script>
```

步骤六：运行到微信开发者工具

1. 运行到微信开发者工具。



2. 请在本地设置里面勾选上“不校验合法域名、web-view (业务域名)、TLS 版本以及 HTTPS 证书”。

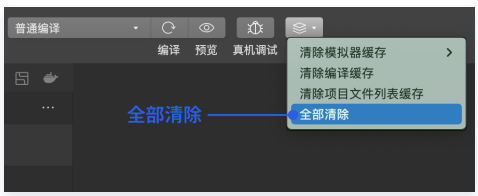


警告：

如果不勾选该条目，则会在控制台出现如下错误。



3. 单击**清缓存 > 全部清除**，避免开发者工具的缓存造成渲染异常。



4. 编译小程序。



5. 该项目快速集成后的预期效果图。



步骤七：拨打您的第一通电话

1. 请单击预览，扫描二维码，在真机环境使用小程序。



2. 登录后，请输入呼叫用户 ID，拨打您的第一通电话。具体效果如下图所示：



> **注意：**
> 第一次使用小程序通话，需要获取摄像头和麦克风权限。

更多特性

1. 设置昵称和头像

如果您需要自定义昵称和头像，可以使用如下接口进行更新：

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "jack", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`);
}
```

2. 自定义铃声

如果您需要自定义来电铃音，可以通过如下接口进行设置：

- 传入本地铃声文件应为绝对路径。
- 如需恢复默认铃声，`filePath` 传空即可。
- 如果打包 Vue2 项目没有铃声，具体参见 Vue2 uni-app 打包微信小程序项目中，铃声没有声音?

```
try {
  await TUICallKitServer.setCallingBell(filePath?: string)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setCallingBell API. Reason: ${error}`);
}
```

3. 群组通话

如果您需要实现群组（即多人）音视频通话，可以通过如下接口进行设置：

```
TUICallKitServer.groupCall({
```

```
    userIDList: ["jane", "mike", "tommy"],
    type: 1,   // 1-语音通话 2-视频通话
    groupID: "12345678"
});
```

4. 悬浮窗

默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 `true` 后显示。

```
TUICallKitServer.enableFloatWindow(enable?: Boolean)
```

API 文档

小程序 TUICallKIt API 文档请参见： TUICallKit API 。

常见问题

1. 包体积过大?

> ⚠ **注意：**
> **TUICallKit 版本须大于或等于3.1.2。**

- 体积优化文档 。

2. 什么是 SDKAppID、SecretKey?

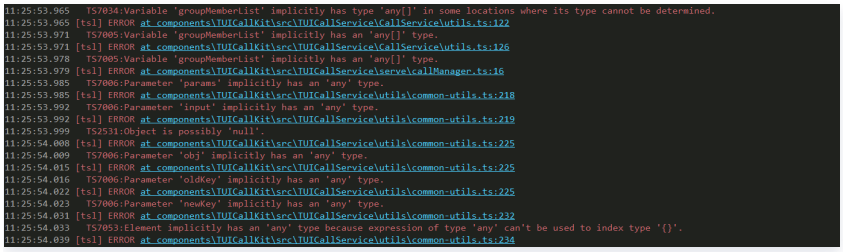
- **SDKAppID**：IM 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话彼此不能互通。
- **Secretkey**：IM 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 IM 服务的鉴权用票据 UserSig。

3. 什么是 UserSig，如何生成 UserSig?

- UserSig 是用户登录即时通信 IM 的密码，其本质是对 UserID 等信息加密后得到的密文。
- UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向项目的接口，在需要 UserSig 时由您的项目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 。

4. 处理 TUICallKit 的 ts 语法校验错误

如果出现如下所示 error, 请在项目中添加如下配置。



.eslintignore

```
TUICallKit
```

tsconfig.json

```
{
  "compilerOptions": {
     "noImplicitAny": true
  }
}
```

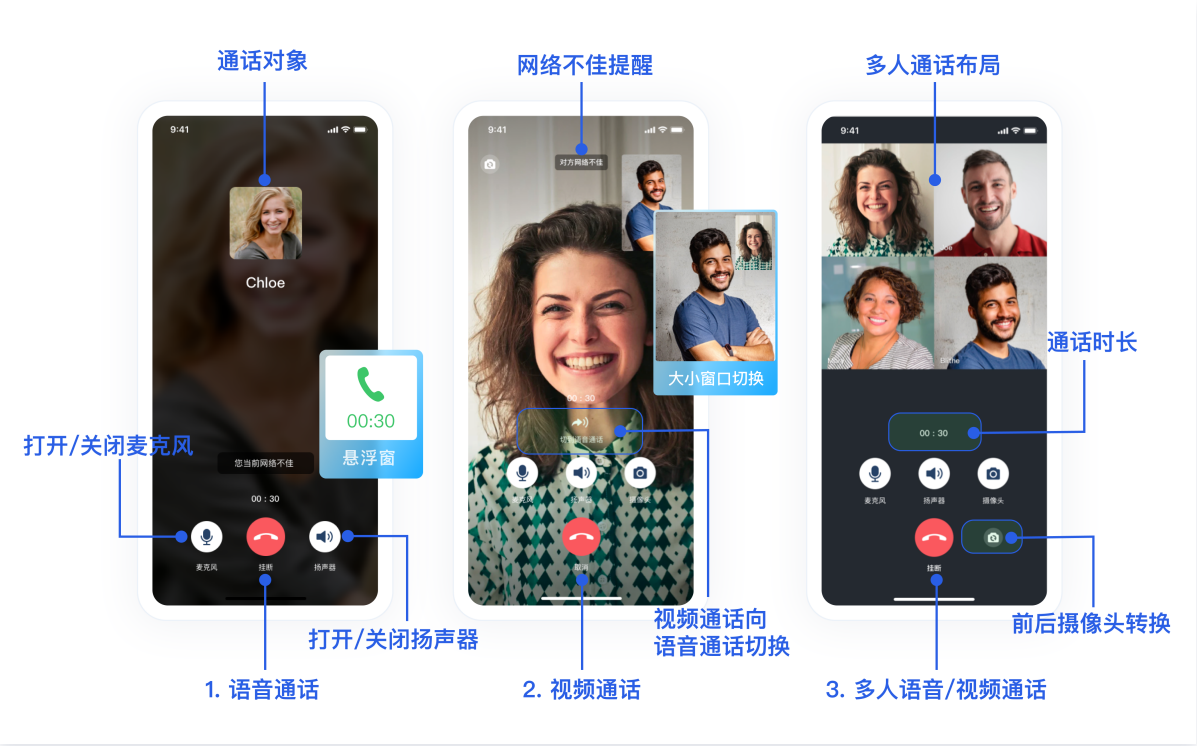
技术咨询

了解更多详情您可 腾讯云通信官方社群 进行咨询和反馈。

微信小程序

最近更新时间：2024-01-03 17:30:41

本文将介绍如何快速完成 TUICallKit 组件的接入，跟随本文档，您将在半小时内得到一个包含完备 UI 界面的视频通话小程序。基本功能如下图所示：



小程序 Demo 体验

- 如果您想要直接体验音视频通话小程序，单击 Demo 体验，扫描小程序二维码。
- 如果您想要直接跑通一个新工程，请直接阅读 微信小程序 Demo 快速跑通 。

开发环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.10.0。

警告：

- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，请使用**企业小程序账号**申请相关权限进行开发。
- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在**真机**上运行体验。

小程序开发准备

步骤一：开通企业类小程序

小程序推拉流标签不支持个人小程序，只支持企业类小程序。需要在 注册 时填写**主体类型为企业**，如下图所示：



步骤二：在小程序控制台开启实时音视频接口

- 小程序推拉流标签使用权限暂时只开放给有限类目，具体支持类目参考该地址 。
- 符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限。



步骤三：在小程序控制台配置域名

在 微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名中设置 request 合法域名 和 socket 合法域名。

- request 合法域名:

```
https://official.opensso.tencent-cloud.com
https://yun.tim.qq.com
https://cloud.tencent.com
https://webim.tim.qq.com
https://query.tencent-cloud.com
https://web.sdk.qcloud.com
https://ap-nanjing.cls.tencentcs.com
```

- **socket 合法域名：**

```
wss://wss.im.qcloud.com
wss://wss.tim.qq.com
```

- **如下图所示，服务器域名配置：**



TUICallKit 源码集成

步骤一：开通 TRTC 服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务。

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建的应用，进入**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**标题，单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。



> **注意：**
> - 如果需要**正式应用上线**，可以单击 购买正式版 即可进入购买页面。
> - IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和**密钥（SecretKey）**并记录下来，它们会在后续的步骤中被用到。

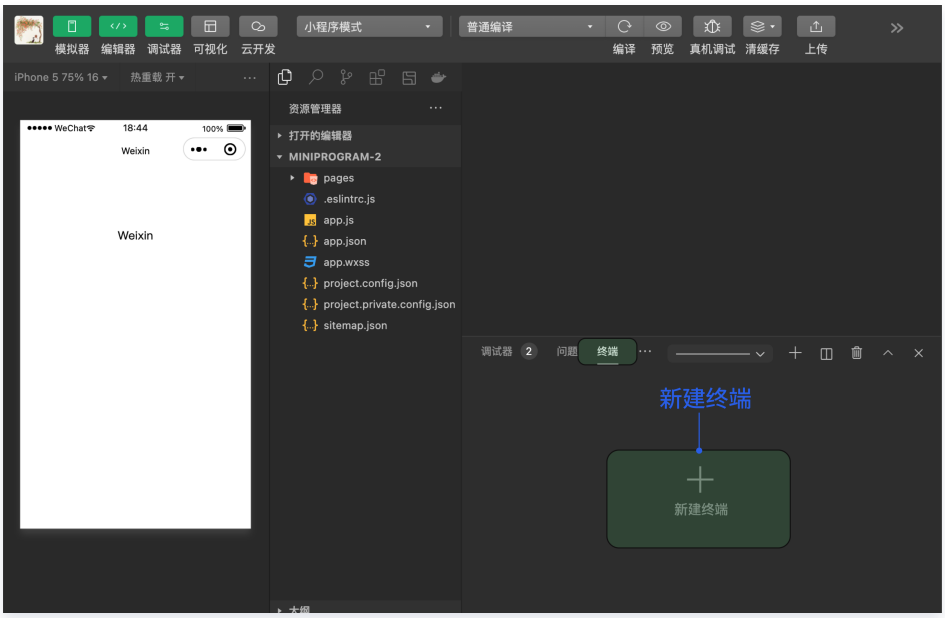


步骤二：创建小程序项目

1. 在微信开发者工具上创建一个小程序项目，选择**不使用模板**。

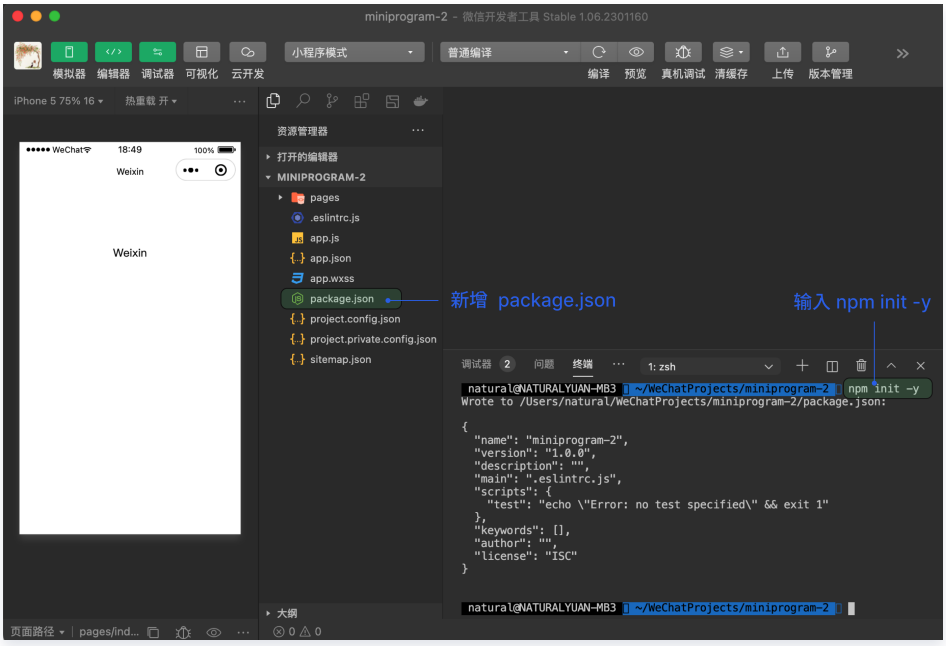


2. 新建终端。



3. 执行 `npm init -y` 命令生成 `package.json` 文件。

```
npm init -y
```



步骤三：下载并导入 TUICallKit 组件

1. 下载 TUICallKit 组件。

```
npm i @tencentcloud/call-uikit-wechat@2.x
```

MacOS 端

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wechat/ ./TUICallKit
```

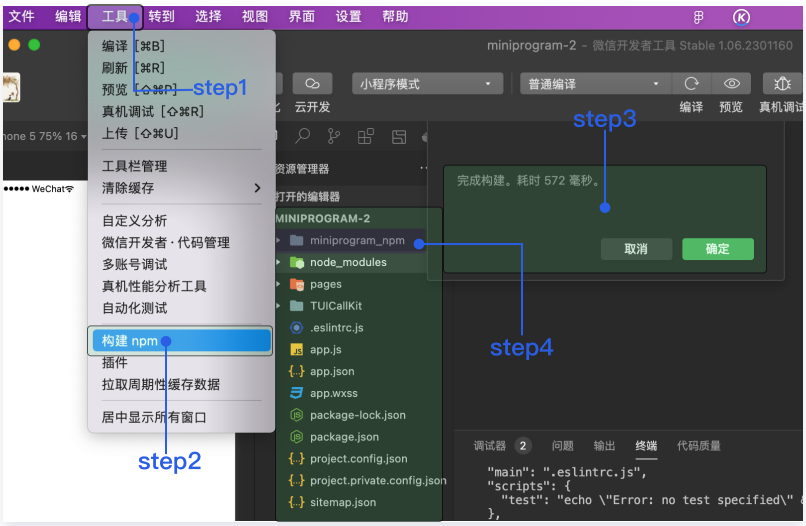
Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-wechat .\TUICallKit /i /e
```

2. 执行完以上命令后，您的目录下生成 TUICallKit 文件夹，其中包含有 TUICallKit 组件。目录如下：

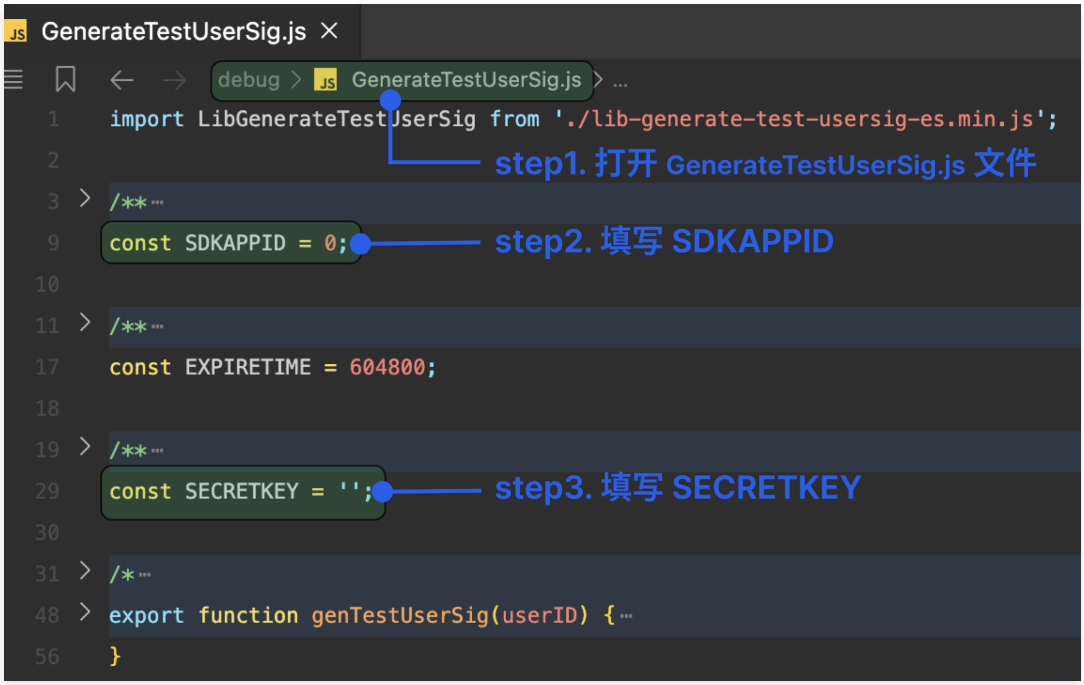


3. 构建 npm，打开微信开发者工具单击工具 > 构建 npm，新增 miniprogram_npm 目录。目录如下：



步骤四：填写 SDKAPPID 以及 SECRETKEY

修改 TUICallKit/debug/ GenerateTestUserSig-es.js 文件 的 SDKAPPID 以及 SECRETKEY。



步骤五：调用 TUICallKit 组件

1. 修改 app.json 文件，添加如下代码，新增全局监听页面。

```
{
 "pages": [
  "pages/index/index",
  "TUICallKit/pages/globalCall/globalCall"
 ],
 "window": {
  "navigationBarTextStyle": "black",
  "navigationStyle": "custom"
 },
 "style": "v2"
```

```
  }
```

2. 修改 index 文件夹下的文件。

index.wxml

```
<view class="container">
  <view class="box">
    <view class="input-box">
      <input type="text" maxlength="20" placeholder="{{isLogin?'请输入呼叫者userID':'请输入登录者userID'
}}" value="{{userID}}" bindinput='bindInputUserID' placeholder-style="color:#BBBBBB;" />
    </view>
    <view class='login'>
      <button class='loginBtn' bindtap="{{isLogin?'call':'login'}}">{{isLogin?'呼叫':'登录'}}</button>
    </view>
  </view>
</view>
```

index.js

```
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/TUICallService/index";
// 导入 CallManager 模块，使您的应用具有全局监听来电的能力
import { CallManager } from "../../TUICallKit/TUICallService/serve/callManager";
import * as GenerateTestUserSig from "../../TUICallKit/debug/GenerateTestUserSig-es.js";
wx.CallManager = new CallManager();
Page({
  data: {
    userID: "",
    isLogin: false,
  },

  bindInputUserID(e) {
    this.setData({
      userID: e.detail.value,
    });
  },

  async login() {
    const userID = this.data.userID;
    if (!userID) return;
    const { userSig, SDKAppID } = GenerateTestUserSig.genTestUserSig({
      userID: userID,
    });
    await wx.CallManager.init({
      sdkAppID: SDKAppID,
      userID: userID,
      userSig: userSig,
      globalCallPagePath: "TUICallKit/pages/globalCall/globalCall",
    });
    wx.showToast({
      title: "登录成功",
      icon: "error",
```

```
    });
    this.setData({
      isLogin: true,
      userID: "",
    });
  },

  async call() {
    await TUICallKitServer.call({
      userID: this.data.userID,
      type: 2,
    });
  },
});
```

index.wxss

```
.container {
  width: 100vw;
  height: 100vh;
}

.box{
  flex: 1;
  width: 100vw;
  margin-top: -40px;
  background: #ffffff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
}

input{
  display: flex;
  font-size: 20px;
  width: 60vw;
}

.login {
  display: flex;
  width: 100vw;
  text-align: center;
  bottom: 5vh;
  margin: 70rpx;
}
.login button{
  width: 80%;
  background-color: #006eff;
  border-radius: 50px;
  color: white;
}
```

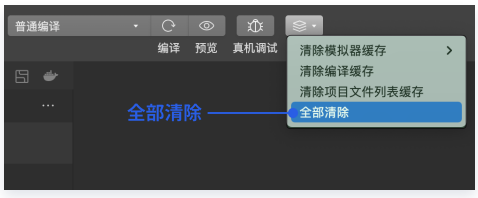
步骤六：编译运行

1. 请在本地设置里面勾选上“不校验合法域名、web-view (业务域名)、 TLS 版本以及 HTTPS 证书”。

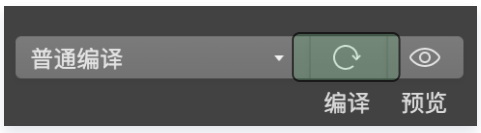


> **警告：**
> **如果不勾选该条目，则会在控制台出现如下错误。**
>
> 
>

2. 单击**清缓存 > 全部清除**，避免开发者工具的缓存造成渲染异常。



3. 编译小程序。



4. 该项目快速集成后的预期效果图。



步骤七：拨打您的第一通电话

- 请单击预览，扫描二维码，在真机环境使用小程序。



- 登录后，请输入呼叫用户 ID，拨打您的第一通电话。具体效果如下图所示：



> **注意：**
> 第一次使用小程序通话，需要获取摄像头和麦克风权限。

TUICallKit 更多特性

1. 设置昵称、头像

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "jack", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`);
}
```

2. 自定义铃声

如果您需要自定义来电铃音，可以通过如下接口进行设置：

- 传入本地铃声文件应为绝对路径。
- 如需恢复默认铃声，`filePath` 传空即可。

```
try {
  await TUICallKitServer.setCallingBell(filePath?: string)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setCallingBell API. Reason: ${error}`);
}
```

3. 群组通话

如果您需要实现群组（即多人）音视频通话，可以通过如下接口进行设置：

```
TUICallKitServer.groupCall({
    userIDList: ["jane", "mike", "tommy"],
```

```
    type: 1,   // 1-语音通话 2-视频通话
    groupID: "12345678"
});
```

API 文档

微信小程序 TUICallKIt API 文档请参考： TUICallKit API 。

常见问题

出现推流黑屏，样式错乱怎么回事?

警告：

微信小程序 Skyline 渲染引擎与 live-pusher 存在兼容性问题，如果您使用微信开发者工具 1.06.2307260+ 版本以上的新创建的项目，请在 app.json 中删除 Skyline 的相关配置。

```
"renderer": "skyline",
"rendererOptions": {
  "skyline": {
    "defaultDisplayBlock": true,
    "disableABTest": true,
    "sdkVersionBegin": "3.0.0",
    "sdkVersionEnd": "15.255.255"
  }
},
"componentFramework": "glass-easel",
"sitemapLocation": "sitemap.json",
"lazyCodeLoading": "requiredComponents"
```

什么是 SDKAppID、SecretKey?

- **SDKAppID**：IM 的应用 ID，用于业务隔离，即不同的 SDKAppID 的通话彼此不能互通。
- **Secretkey**：IM 的应用密钥，需要和 SDKAppID 配对使用，用于签出合法使用 IM 服务的鉴权用票据 UserSig。

什么是 UserSig，如何生成 UserSig?

- UserSig 是用户登录即时通信 IM 的密码，其本质是对 UserID 等信息加密后得到的密文。
- UserSig 签发方式是将 UserSig 的计算代码集成到您的服务端，并提供面向项目的接口，在需要 UserSig 时由您的项目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 。

错误提示“The package you purchased does not support this ability”，如何处理?

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。具体可参考步骤三 。

注意：

单击**免费体验**以后，部分之前使用过 实时音视频 TRTC 服务的用户会提示：

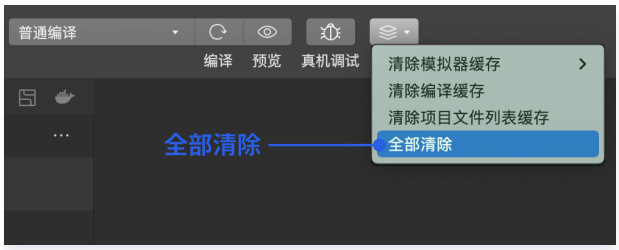
```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or the Tencent Cloud accountis in arrears
```

当实时音视频 TRTC 的免费额度（10000分钟）已经过期或者耗尽，就会导致开通此项服务失败，这里您可以单击 TRTC 控制台，找到对应 SDKAppID 的应用管理页，开通后付费功能后，再次**启用应用**即可正常体验音视频通话能力。具体参考下图：



微信开发者工具有路径报错，如何处理?

如果微信开发者工具有路径保存，请清除缓存，避免开发者工具的缓存造成渲染异常。具体参考下图：



技术咨询

了解更多详情您可 腾讯云通信官方社群 进行咨询和反馈。

Flutter

最近更新时间：2023-12-26 17:09:43

本文将介绍如何用最短的时间完成 TUICallKit 组件的接入，跟随本文档，您将在一个小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的视频通话功能。

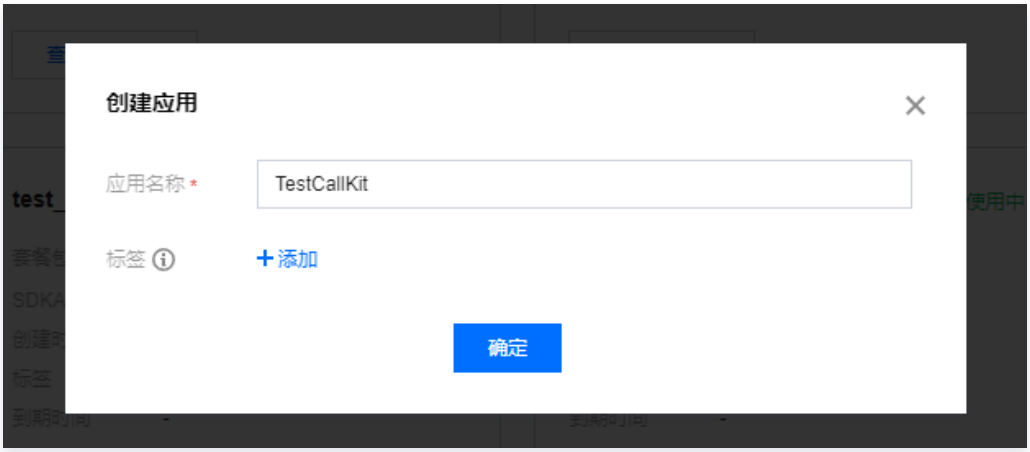
环境准备

Flutter 3.0及更高版本。

步骤一：开通服务

TUICallKit 是基于腾讯云 即时通信 IM 和 实时音视频 TRTC 两项付费 PaaS 服务构建出的音视频通信组件。您可以按照如下步骤开通相关的服务并体验 7 天的免费试用服务：

1. 登录到 即时通信 IM 控制台，单击**创建新应用**，在弹出的对话框中输入您的应用名称，并单击**确定**。



2. 单击刚刚创建出的应用，进入基本配置页面，并在页面的右下角找到含 UI 低代码场景方案功能区，单击免费体验即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



> **注意**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

3. 在同一页面找到 **SDKAppID** 和 **密钥(SecretKey)** 并记录下来，它们会在后续的 步骤四：登录 TUI 组件 中被用到。



友情提示：

单击免费体验以后，部分之前使用过 实时音视频 TRTC 服务的用户会提示：

```
[-100013]:TRTC service is  suspended. Please check if the package balance is 0 or the Tencent Cloud accountis in arrears
```

因为新的 IM 音视频通话能力是整合了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础的 PaaS 服务，所以当 实时音视频 TRTC 的免费额度（10000分钟）已经过期或者耗尽，就会导致开通此项服务失败，这里您可以单击 TRTC 控制台，找到对应 SDKAppID 的应用管理页，示例如图，开通后付费功能后，再次启用应用即可正常体验音视频通话能力。



步骤二：导入 TUICallKit 组件

使用 Flutter 导入组件，具体步骤如下：

1. 在您的 pubspec.yaml 文件中添加 tencent_calls_uikit 插件依赖，点击插件超链接切换到 Versions 目录下查询最新版本。

```
dependencies:
  tencent_calls_uikit: 最新版本
```

2. 执行以下命令安装组件：

```
flutter pub get
```

步骤三：完成工程配置

1. 将 TUICallKit的navigatorObserver 添加到 App组件中，以MateriaApp为例，代码如下：

```
import 'package:tencent_calls_uikit/tuicall_kit.dart';

MaterialApp (
    navigatorObservers：[TUICallKit.navigatorObserver]，
    ...
)
```

2. 如果您需要编译运行在 Android 平台，由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 proguard-rules.pro 文件中添加如下代码：

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

3. 如果您的工程需要在iOS 模拟器上调试，您需要在工程的 /ios/Podfile 文件中添加以下代码：

```
post_install do |installer|
  installer.pods_project.targets.each do |target|
    flutter_additional_ios_build_settings(target)
    target.build_configurations.each do |config|
      config.build_settings['VALID_ARCHS'] = 'arm64 arm64e x86_64'
      config.build_settings['VALID_ARCHS[sdk=iphonesimulator*]'] = 'x86_64'
    end
  end
end
```

4. 如果您需要使用 iOS 的音视频功能，您需要授权麦克风和摄像头的使用权限。

   授权操作方法：在您的 iOS 工程的 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。

```
<key>NSCameraUsageDescription</key>
<string>CallingApp需要访问您的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>CallingApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```

步骤四：登录 TUICallKit 组件

在您的项目中添加如下代码，这个步骤异常关键，因为只有在登录成功后才能正常使用 TUICallKit 的各项功能，故请您耐心检查相关参数是否配置正确：

```
TUIResult result = TUICallKit.instance.login(SDKAppID,      // 请替换为第一步得到的SDKAppID
                                             'userId',      // 请替换为您的User ID
                                             'userSig');    // 您可以在控制台计算一个UserSig并填到该位置
```

在调用完成后，您可以通过 TUIResult 类型的返回值查看是否登录成功。

参数说明：

- **sdkAppId**：在步骤一中的最后一步中您已经获取到，这里不再赘述。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **userSig**：使用步骤三中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig，也可以参照我们的 Github 中的示例代码 进行调试，更多信息请参见 如何计算及使用 UserSig 。

> ⚠ **注意：**
>
> 尊敬的开发者，您好，当前步骤是目前开发者反馈问题最多的步骤，常见问题如下：
>
> - sdkAppId 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
> - userSig 被错配成了加密密钥（Secretkey），userSig 是用 SecretKey 把 sdkAppId、userId 以及过期时间等信息加密得来的，而不是直接把 Secretkey 配置成 userSig。
> - userId 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 userId 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 userId。

> **警告：**
> Github 中的示例代码使用了 genTestUserSig 函数在本地计算 userSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 userSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUICallKit 组件时向您的服务器请求实时计算出的 userSig。

步骤五：拨打通话

1对1视频通话

通过调用 TUICallKit 的 call 函数并指定通话类型和被叫方的 userId，就可以发起语音或者视频通话。

```
// 假设给mike拨打视频电话
TUICallKit.instance.call('mike', TUICallMediaType.video);
```

参数	类型	含义
userId	String	目标用户的 UserID：`"mike"`
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

群内视频通话

通过调用 TUICallKit 的 groupCall 函数并指定通话类型和被叫方的 UserID 列表，就可以发起群内的语音或者视频通话。

```
// 假设在groupId:0001的群组内，向denny, mike 和 tommy 发起视频通话
TUICallKit.instance.groupCall('0001', ['denny', 'mike', 'tommy'], TUICallMediaType.video);
```

参数	类型	含义
groupId	String	群组 ID
userIdList	Array	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

> **说明**
> - 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。
> - TUICallKit 目前还不支持发起非群组的多人视频通话，如果您有此类需求，欢迎反馈： TUIKit 需求收集表 。

步骤六：接听通话

在 步骤四 完成后，收到来电请求后，TUICallKit 组件会自动启动相应的接听界面。

步骤七：更多特性

一、设置昵称&头像

如果您需要自定义昵称或头像，可以使用如下接口进行更新：

```
// 函数定义为: Future<TUIResult> setSelfInfo(String nickname, String avatar)
TUIResult result = TUICallKit.instance.setSelfInfo('userName', 'url:********');
```

> **注意**
> 因为用户隐私限制，非好友之间的通话，被叫的昵称和头像更新可能会有延迟，一次通话成功后就会顺利更新。

二、离线唤醒

完成以上步骤，就可以实现音视频通话的拨打和接通，但如果您的业务场景需要在 `应用的进程被杀死后` 或者 `应用退到后台后` ，还可以正常接收到音视频通话请求，就需要增加离线唤醒功能，详情见 离线唤醒（Flutter）。

三、悬浮窗功能

如果您的业务需要开启悬浮窗功能，您可以在 TUICallKit 组件初始化时调用以下接口开启该功能：

```
TUICallKit.instance.enableFloatWindow(true);
```

四、通话状态回调

如果您的业务需要观察**通话状态**，例如通话开始、结束，以及通话过程中的网络质量等等，可以添加观察者，观察如下事件：

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError: (int code, String message) {
      //您的回调处理逻辑
    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole) {
      //您的回调处理逻辑
    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole, double totalTime)
{
      //您的回调处理逻辑
    },, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {
      //您的回调处理逻辑
    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId, TUICallMediaType
callMediaType) {
      //您的回调处理逻辑
    }
));
```

五、自定义铃音

如果您需要自定义来电铃音，可以通过如下接口进行设置：

```
TUICallKit.instance.setCallingBell('flie path');
```

常见问题

1、错误提示“The package you purchased does not support this ability”？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

2、如何购买套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请单击页面右侧，进行售前套餐咨询，或者加入 QQ 群：605115878，进行咨询和反馈。

3、Android 编译报错：uses-sdk:minSdkVersion 16 cannot be smaller than version 19.

详细的报错信息如下：

```
uses-sdk:minSdkVersion 16 cannot be smaller than version 19 declared in library [:tencent_calls_uikit]
/Users/xxx/xxxx/xxxx/xxxx/demo/flutter/callkit/callkit/build/tencent_calls_uikit/intermediates/merged_manifest/debug/AndroidManifest.xml as the library might be using APIs not available in 16
Suggestion: use a compatible library with a minSdk of at most 16,
        or increase this project's minSdk version to at least 19,
        or use tools:overrideLibrary="com.tencent.cloud.tuikit.flutter.tuicallkit" to force usage (may lead to runtime failures)
```

这是因为 `tencent_call_uikit` Plugin Android 端支持的 minSdkVersion 是 19，可以将您工程中的 `android/app/build.gradle` 文件中的 `minSdkVersion` 修改为 19 及其以上即可。

交流与反馈

- 如果您在使用过程中，有什么建议或者意见，可以在这里反馈：TUICallKit 产品反馈问卷，感谢您的反馈。
- 如果您是开发者，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群：605115878，进行技术交流和产品沟通。

界面定制（TUICallKit）
Android

最近更新时间：2023-03-24 17:50:25

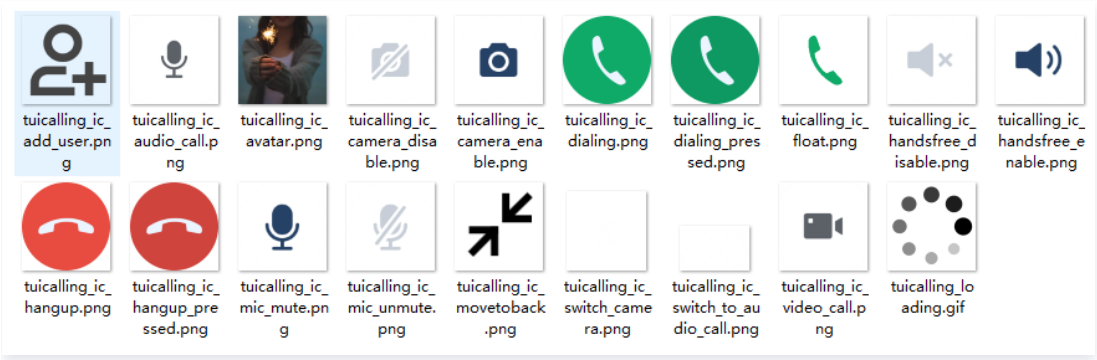
本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUICallKit 的用户界面进行调整，TUICallKit 的界面源代码位于 Github 中的 `Android/tuicallkit` 文件夹下面：

替换图标

您可以直接替换 `res\drawable-xxhdpi` 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 `res\raw` 文件夹下的三个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_hangup.mp3	被挂断的声音
phone_ringing.mp3	接到呼叫时的声音

替换文案

您可以通过修改 values-zh 和 values-en 中的 `strings.xml` 文件来修改视频通话界面中的字符串内容。

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI 组件实现的，您可以删掉 tuicallkit 文件夹，完全基于 TUICallEngine 实现一套自己的 UI 界面。

TUICallEngine

TUICallEngine 是整个通话组件的底层接口，主要提供了1对1音视频通话和群组内通话的发起、接听、拒绝、结束以及设备操作等关键接口。

API	描述

createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调

onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调

关键类型定义

API	描述
TUICallDefine.MediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallDefine.Role	通话的角色，枚举类型：主叫、被叫
TUICallDefine.Status	通话的状态，枚举类型：空闲、待接听、接听中
TUICommonDefine.RoomId	音视频房间 ID，支持数字、字符串两种类型
TUICommonDefine.Camera	摄像头 ID 参数，枚举类型：前摄、后摄
TUICommonDefine.AudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUICommonDefine.NetworkQualityInfo	当前的网络质量信息

iOS

最近更新时间：2023-04-26 16:50:14

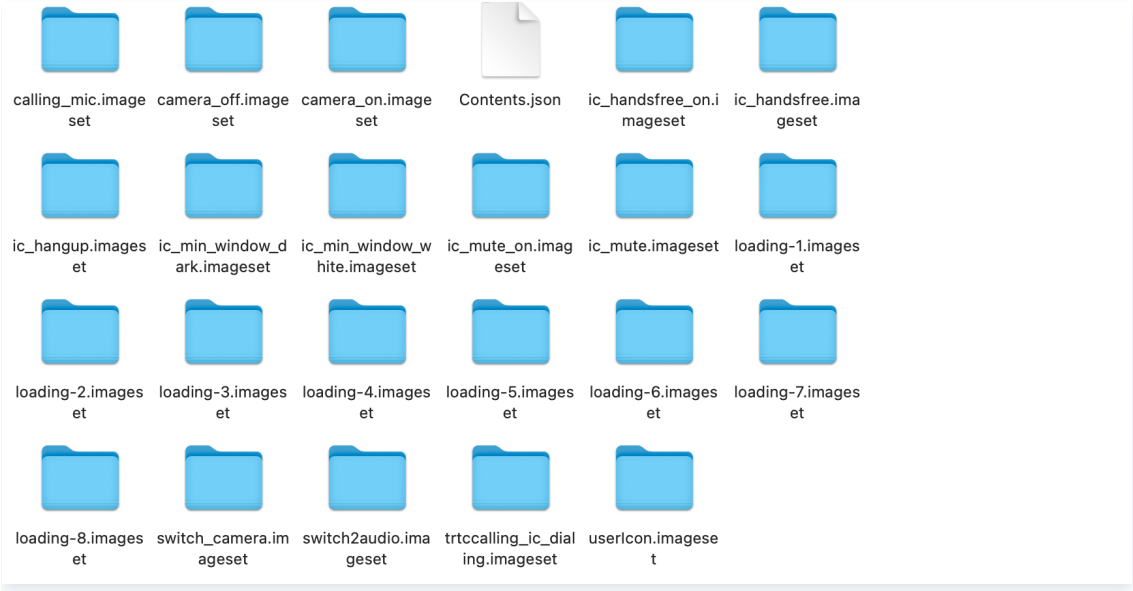
本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUICallKit 的用户界面进行调整，TUICallKit 的界面源代码位于 Github 中的 `iOS/TUICallKit` 文件夹下面：

替换图标

您可以直接替换 `Resources\Calling.xcassets` 文件夹下的图标，以确保整个 app 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 `Resources\AudioFile` 文件夹下的三个音频文件来达到替换铃声的目的。

文件名	用途
phone_dialing.m4a	发起呼叫时的声音
phone_hangup.mp3	被挂断的声音
phone_ringing.flac	接到呼叫时的声音

替换文案

您可以通过修改 zh-Hans.lproj 和 en.lproj 中的 `CallingLocalized.strings` 文件来修改视频通话界面中的字符串内容。

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI 组件实现的，您可以删掉 tuicallkit 文件夹，完全基于 TUICallEngine 实现一套自己的 UI 界面。

TUICallEngine

TUICallEngine 是整个通话组件的底层接口，主要提供了1对1音视频通话和群组内通话的发起、接听、拒绝、结束以及设备操作等关键接口。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
setRenderView	设置显示视频画面的 View 对象
startRemoteView	设置显示视频画面的 View 对象
stopRemoteView	设置显示视频画面的 View 对象
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调

onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调

关键类型定义

API	描述
TUICallMediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallRole	通话的角色，枚举类型：主叫、被叫
TUICallStatus	通话的状态，枚举类型：空闲、待接听、接听中
TUIRoomId	音视频房间 ID，支持数字、字符串两种类型
TUICamera	摄像头 ID 参数，枚举类型：前摄、后摄
TUIAudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUINetworkQuality	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

Web

最近更新时间：2023-08-03 10:13:52

本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：界面微调方案和自实现 UI 方案。

- 界面微调方案：将会在已有的开源 TUICallKit 做代码修改，适用于 `Vue2.7 + Typescript` 或者 `Vue3 + Typescript` 项目，若您采用其他语言或者技术栈，请使用自实现 UI 方案。
- 自实现 UI 方案：TUICallKit 是 UI 组件，其底层使用的是腾讯云 TUICallEngine SDK。SDK 封装了音视频通话常见场景的操作，适用于任意前端框架，您可以使用这套 JavaScript 接口制作交互更灵活的业务页面。

方案一：界面微调方案

TUICallKit 是含 UI 音视频通话组件，请点击 发布日志 查看最近更新记录。为了方便的进行源码调整，推荐直接将组件拷贝到您的项目中（源码为 TypeScript 版本）

1. 下载源码

Vue3

```
npm install @tencentcloud/call-uikit-vue
```

Vue2

```
npm install @tencentcloud/call-uikit-vue2
```

2. 将源码拷贝到自己的项目中，以拷贝到 `src/components/` 目录为例：

macOS + Vue3

```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uikit-vue/* ./src/components/TUICallKit
```

macOS + Vue2

```
mkdir -p ./src/components/TUICallKit && cp -r ./node_modules/@tencentcloud/call-uikit-vue2/* ./src/components/TUICallKit
```

Windows + Vue3

```
xcopy .\node_modules\@tencentcloud\call-uikit-vue  .\src\components\TUICallKit /i /e
```

Windows + Vue2

```
xcopy .\node_modules\@tencentcloud\call-uikit-vue2  .\src\components\TUICallKit /i /e
```

3. 修改引入路径

需要将 CallKit 改为从本地文件中引入，如下方代码。其他用法细节可参考 TUICallKit 快速接入。

```
import { TUICallKit, TUICallKitServer } from "./components/TUICallKit/src/index";
```

> **注意：**
> 此方法可能与您的 ESLint、TypeScript 配置冲突，若出现报错，可参考 步骤4 。

4. **解决源码拷贝可能导致的报错**

如果您在使 TUICallKit 组件时遇到了报错，请不要担心，大多数情况下这是由于 ESLint 和 TSConfig 配置不一致造成的。您可以查阅文档，按照要求正确配置即可。如果您需要帮助，请随时联系我们，我们将确保您能够成功地使用此组件。以下是几个常见的问题：

ESLint 报错

若 TUICallKit 与您项目的代码风格不一致导致报错，可将本组件目录屏蔽，如在项目根目录增加 `.eslintignore` 文件，如：

```
# .eslintignore
src/components/TUICallKit
```

TypeScript 报错

如遇 `Cannot find module '../package.json'` 报错，是因为 TUICallKit 内引用了 JSON 文件，可在 `tsconfig.json` 中添加相关配置，示例：

```
{
  "compilerOptions": {
    "resolveJsonModule": true
  }
}
```

其他 TSConfig 问题请参见 TSConfig Reference。

调整 UI 布局

版本 ⩾ v2.3.2

您可以通过修改 `/src/Components/TUICallKit-vue3/components /` 文件下的不同页面来修改音视频通话界面，每个单页面的功能如下：

```
- components/
    - common                公共组件
    - SingleCall            单人通话相关组件
    - GroupCall             群组通话相关组件
```

版本 < v2.3.2

您可以通过修改 src/components/ 文件下的不同页面来修改音视频通话界面，每个单页面的功能如下：

```
- components/
    - Calling-C2CVideo.vue        1v1 视频通话
    - Calling-Group.Vue           多人音频、视频通话
    - ControlPanel.vue            控制面板
    - ControlPanelItem.vue        控制面板子项
    - Dialing.vue                 拨打电话页面、来电页面、1v1 音频通话
    - MicrophoneIcon.vue          可显示音量变化的麦克风 Icon
    - TUICallKit.vue              TUICallKit 总组件
```

样式文件在 src/style.css ，根据不同的 UI 样式，可以进行自行调整。

调整业务逻辑

版本 ⩾ v2.3.2

主要业务逻辑代码在 CallService/index.ts 中，可以在 server.ts 中找到 handleKickedOut() 函数中添加处理逻辑，例如：

```
private _handleKickedOut(event: any): void {
  console.warn(`${NAME.PREFIX}kickOut event data: ${JSON.stringify(event)}.`);
  this.kickedOut && this.kickedOut(event);
  TUIStore.update(StoreName.CALL, NAME.CALL_TIPS, this._generateText(CallTips.KICK_OUT));
  this._resetCallStore();
}
```

版本 < v2.3.2

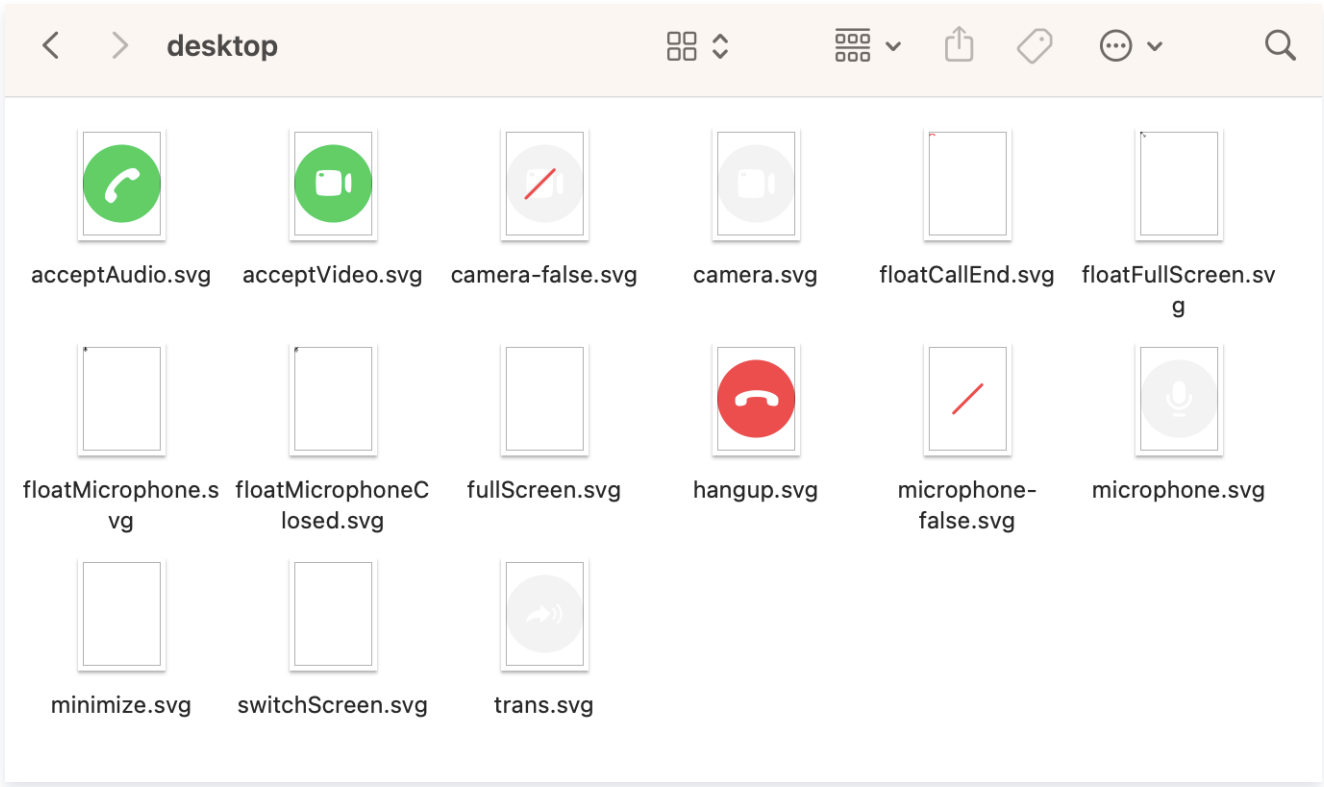
主要业务逻辑代码在 server.ts 中，配合 store/index.ts 使用。如想要针对性的对“被踢出”状态做处理，可以在 server.ts 中找到 handleKickedOut() 函数中添加处理逻辑，例如：

```
private handleKickedOut(event: any) {
    console.error("TUICallKit Kicked Out", JSON.stringify(event));
    // do some thing...
    alert("该用户被踢出，请重新登录！");
}
```

替换图标

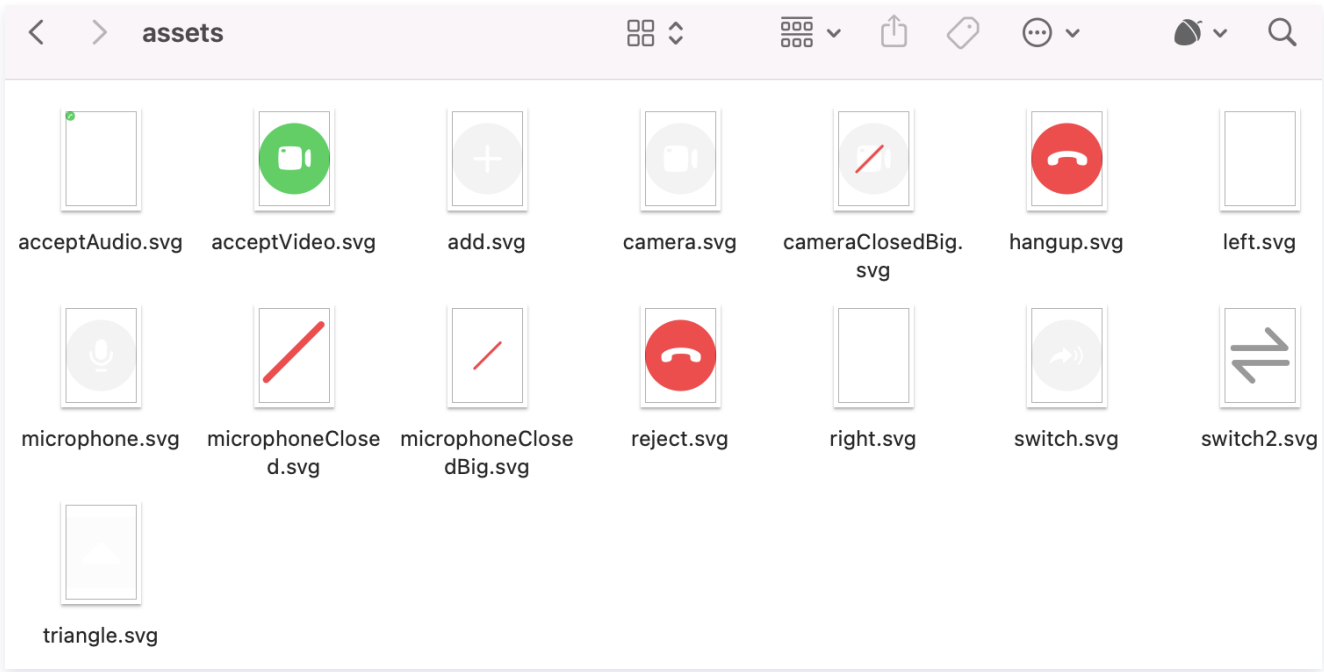
版本 ⩾ v2.3.2

您可以直接修改 /src/Components/TUICallKit-vue3/static / 文件夹下的图标组件，以确保整个应用中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



版本 < v2.3.2

您可以直接修改 `src/icons` 文件夹下的图标组件，以确保整个应用中的图标色调风格保持一致，请在替换时保持图标文件的名字不变，图标预览可参考 `src/assets` 。



替换铃声

您可以替换 `TUICallKit\static` 文件夹下的两个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_ringing.mp3	接到呼叫时的声音

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI SDK实现的，您可以完全基于 TUICallEngine 实现一套自己的 UI 界面。详情可见：

- TUICallEngine 接入指引
- TUICallEngine API 接口地址

微信小程序

最近更新时间：2023-03-27 10:22:04

本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

TUICallKit 是含 UI 音视频通话组件，请点击发布日志查看最近更新记录。为了方便的进行源码调整，推荐直接将组件拷贝到您的项目中：

1. 下载 TUICallKit 源码

TUICallKit 版本 ⩾ v1.3.0

```
npm i @tencentcloud/call-uikit-wechat
```

TUICallKit 版本 ⩽ v1.2.7

```
npm i @tencentcloud/call-uikit-wechat@1.2.7
```

2. 拷贝 TUICallKit 源码至项目中

macOS 端

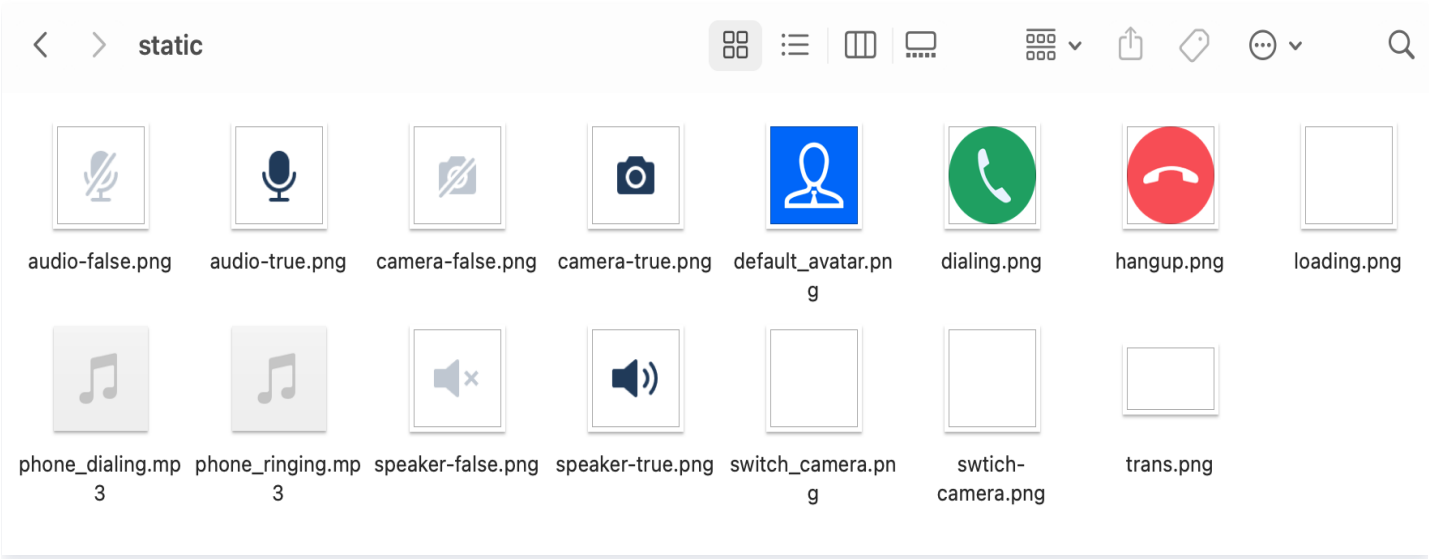
```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wechat/ ./TUICallKit
```

Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-wechat .\TUICallKit /i /e
```

替换图标

您可以直接替换 `TUICallKit\static` 文件夹下的图标，以确保整个小程序中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 `TUICallKit\static` 文件夹下的两个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_ringing.mp3	接到呼叫时的声音

版权所有：腾讯云计算（北京）有限责任公司

Flutter

最近更新时间：2023-08-18 15:51:52

本文将介绍如何定制 TUICallKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。
注意：页面定制方案需要使用 tencent_calls_uikit 插件 1.8.0 以上的版本。

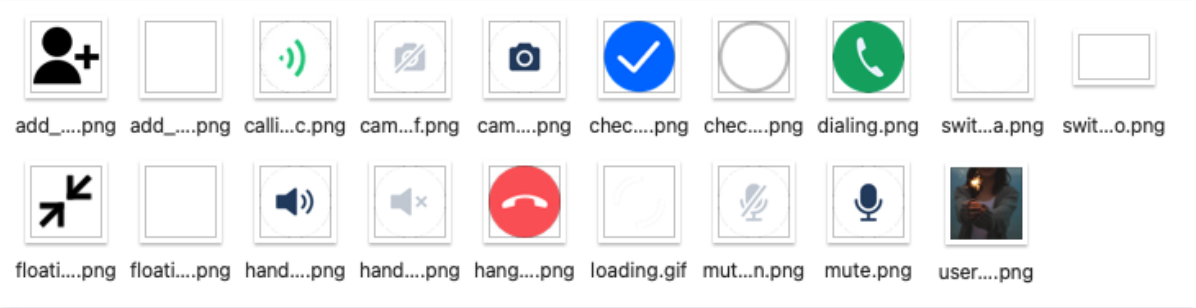
方案一：界面微调方案

您可以下载 tencent_calls_uikit 插件的最新版本到本地，之后在您的工程中使用本地依赖的方式接入插件，本地依赖方式如下：
在工程 **pubspec.yaml** 文件中的 **dependencies** 节点下，添加 **tencent_calls_uikit** 插件依赖，示例如下：

```
dependencies:
  tencent_calls_uikit:
    path: 插件目录
```

替换图标

您可以直接替换 **assets\images** 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换铃声

您可以替换 **assets\audios** 文件夹下的三个音频文件来达到替换铃声的目的：

文件名	用途
phone_dialing.mp3	发起呼叫时的声音
phone_hangup.mp3	被挂断的声音
phone_ringing.mp3	接到呼叫时的声音

替换文案

您可以通过修改 **lib\src\i18n** 目录下 **strings.g.dart** 文件中的的字符串来修改视频通话界面中的字符串内容。

方案二：自实现 UI 方案

TUICallKit 的整个通话功能是基于 TUICallEngine 这个无 UI 组件实现的，您可以不依赖 **tencent_calls_uikit** 插件，而直接依赖 tencent_calls_engine 插件，完全基于 TUICallEngine 实现一套自己的 UI 界面。

TUICallEngine

TUICallEngine 是整个通话组件的底层接口，主要提供了1对1音视频通话和群组内通话的发起、接听、拒绝、结束以及设备操作等关键接口。

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，接收自己感兴趣的回调事件。

离线唤醒（TUICallKit）
Flutter

最近更新时间：2023-12-22 17:52:01

离线唤醒功能，能够让您的 App 在后台运行或者离线状态下依然能够收到音视频通话的响铃呼叫，TUICallKit Flutter 需要依赖厂商：Apple、Google、Vivo、Xiaomi 提供的推送能力来进行消息通知。

配置离线推送

1. iOS

iOS 平台的离线唤醒支持 APNs 和 VoIP 具体配置流程请参考：

- APNs 离线唤醒配置流程
- VoIP 离线唤醒配置流程

2. Android

Android 平台具体配置流程，请参考：离线推送（Flutter）。

自定义离线消息内容

在基础的离线推送配置成功后，您就可以正常收到对来电推送，如果您需要自定义离线消息的内容，可通过 call 和 groupcall 接口中的参数 params 中的 TUIOfflinePushInfo 实现自定义通知内容，可以自定义的内容包括**标题、内容、离线消息铃声**等内容。具体使用如下：

```
Future<void> call(String userId, TUICallMediaType callMediaType, [TUICallParams? params])

class TUICallParams {
  TUIOfflinePushInfo offlinePushInfo;  // 离线消息信息
}

class TUIOfflinePushInfo {
  String? title;                    //离线消息title
  String? desc;                     //离线消息内容
  bool? ignoreIOSBadge;
  String? iOSSound;                 //iOS离线消息铃声
  String? androidSound;             //Android离线消息铃声
  String? androidOPPOChannelID;         //OPPO离线消息参数
  int? androidVIVOClassification;       //VIVO离线消息参数
  String? androidXiaoMiChannelID;       //小米离线消息参数
  String? androidFCMChannelID;          //FCM参数
  String? androidHuaWeiCategory;        //华为离线消息参数
  bool? isDisablePush;              //是否开启离线推送
  TUICallIOSOfflinePushType? iOSPushType;  //离线消息类型
}}

enum TUICallIOSOfflinePushType {
  APNs,
  VoIP,
}
```

自定义离线消息铃声

1. iOS

修改 call 和 groupcall 接口中的参数 params 中的 TUIOfflinePushInfo.iOSSound 设置 iOS 平台上的离线消息铃声。

2. Android

FCM

在完成第一步：配置离线推送之后，即可在离线的情况下收到消息推送，但是是默认的铃声，如果您需要支持自定义铃声，还需要进行如下操作：

1、在IM 控制台-基础配置，选择 Android 原生离线推送配置 -> 选择 Google 的证书进行编辑，添加 ChannelID，如下图所示



2、在工程的 android/app/src/main/res/raw/ 目录下，放入自定义的铃声文件，示例："incoming_call.mp3'

3、在登录成功之后，通过 `tim_ui_kit_push_plugin` 插件 向 Android 系统注册播放自定义铃音的 NotificationChannel，示例代码如下

```
void createNotificationChannel() {
  AndroidNotificationChannelGroup group = new AndroidNotificationChannelGroup("GroupID",
"GroupName");
  AndroidFlutterLocalNotificationsPlugin plugin = AndroidFlutterLocalNotificationsPlugin();
  plugin.createNotificationChannelGroup(group);

  AndroidNotificationChannel channel = AndroidNotificationChannel(
      "fcm_push_channel", "ChannelName",
      groupId: "GroupID",
      importance: Importance.high,
      enableLights: true,
      enableVibration: true,
      sound: RawResourceAndroidNotificationSound("incoming_call")); // Note: Media files do not need a suffix
  plugin.createNotificationChannel(channel);
}
```

联系我们

如遇到该问题，需要做兼容处理，您可以加入我们的 QQ 群（605115878）进行咨询与反馈。

Android

最近更新时间：2023-10-17 16:48:51

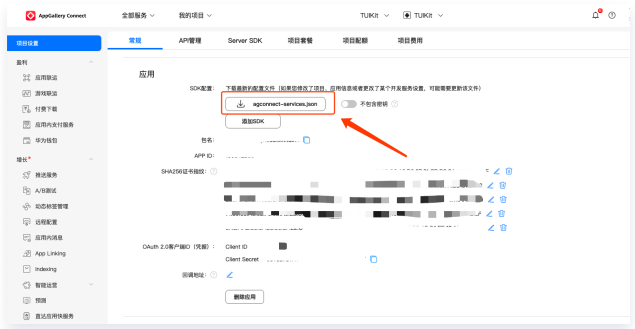
离线唤醒功能，能够让您的应用在 后台运行 或者 离线状态 下依然能够收到音视频通话的响铃呼叫，TUICallKit 集成了 TUIOfflinePush 组件实现离线唤醒功能，本文将介绍如何在音视频通话项目中集成 TUIOfflinePush 组件。

前期准备

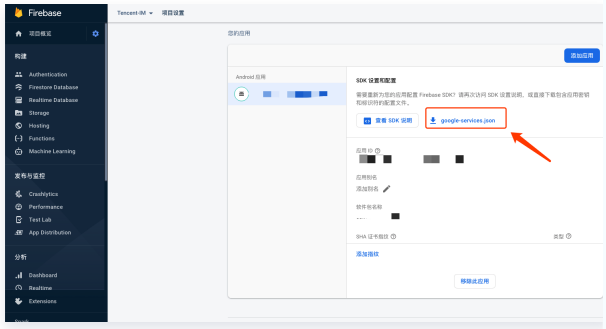
1. **注册应用到厂商推送平台**
   离线推送功能依赖厂商原始通道，您需要将自己的应用注册到各个厂商的推送平台，得到 APPID 和 APPKEY 等参数，具体请参见 离线推送(Android) 中的步骤一。

   下述两个文件在后续步骤会用到：
   - 注册华为平台的时候，下载 agconnect-services.json 文件，并保存。
   - 注册 Google 平台的时候，下载 google-services.json 文件，并保存。

   华为

   Google FCM

2. **在 IM 控制台进行配置**
   注册厂商通道需要传入自己的包名，各厂商填入的包名需保持一致，用于消息互通。具体请参见 离线推送(Android) 中的步骤二，记录生成的 ID，APPID 以及 APPKEY，他们会在后续步骤中用到。
   推送证书 ID 如下：



步骤一：下载并导入组件

1. 在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuiofflinepush 子目录到您当前工程中的 app 同级目录中，如下图：



2. 在工程根目录下找到 settings.gradle 文件，并在其中增加如下代码：

```
include ':tuiofflinepush'
```

3. 在 app 目录下找到 build.gradle 文件，并在其中增加如下代码，它的作用是声明当前 app 对新加入的 tuiofflinepush 组件的依赖：

```
api project(':tuiofflinepush')
```

步骤二：完成工程配置

1. 在 app 目录下，找到 build.gradle 文件，修改应用包名为自己的包名。

```
applicationId 'com.****.trtc'
```

2. 在 app 目录下，找到 build.gradle 文件，设置 ViVo 接入参数 VIVO_APPKEY 、 VIVO_APPID 以及 HONOR_APPID ，避免编译或运行报错。

```
manifestPlaceholders = [
    "VIVO_APPKEY": "PLACEHOLDER",
    "VIVO_APPID" : "PLACEHOLDER",
    "HONOR_APPID": "PLACEHOLDER"
]
```

3. 配置 华为 和 谷歌 文件：
   在 app 目录下，替换 google-services.json 文件，该文件为您在 前期准备 注册 Google 平台的时候保存的文件。
   在 app 目录下，增加 agconnect-services.json 文件，该文件为您在 前期准备 注册华为平台的时候保存的文件。
4. 将 前期准备 记录的 ID、APPID 和 APPKEY 填入 PrivateConstants 文件中，并检查参数是否配置正确。填充的参数如下：

```
public class PrivateConstants {
    /****** 小米离线推送参数start ******/
    // 在腾讯云控制台上传第三方推送证书后分配的证书 ID
    public static final long XM_PUSH_BUZID = 您应用分配的证书 ID;
    // 小米开放平台分配的应用APPID及APPKEY
    public static final String XM_PUSH_APPID = "您应用分配的 APPID";
    public static final String XM_PUSH_APPKEY = "您应用分配的 APPKEY";
    /****** 小米离线推送参数end ******/
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

> **注意**
> 这一步非常重要，请认真检查参数是否配置正确。

完成以上步骤，您的工程就具备了 TUICallKit 的离线唤醒功能。

步骤三：自定义离线通知内容

Kotlin

TUICallKit 提供了默认的通知样式，但是如果您需要自定义通知内容，请修改 OfflinePushInfoConfig.kt 文件 。

```
object OfflinePushInfoConfig {
        pushInfo.desc = "You have receive a new call"
        // OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as
the console.
        pushInfo.androidOPPOChannelID = "tuikit"
        pushInfo.isIgnoreIOSBadge = false
        pushInfo.iosSound = "phone_ringing.mp3"
        pushInfo.androidSound = "phone_ringing" //Note：don't add suffix
        //VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
        pushInfo.androidVIVOClassification = 1
        //FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
        pushInfo.androidFCMChannelID = "fcm_push_channel"
        //Huawei message type
        pushInfo.androidHuaWeiCategory = "IM"
        //IOS push type: if you want to use VoIP, please modify type to
TUICallDefine.IOSOfflinePushType.VoIP
        pushInfo.iosPushType = TUICallDefine.IOSOfflinePushType.APNs
        return pushInfo
    }
}
```

Java

TUICallKit 提供了默认的通知样式，但是如果您需要自定义通知内容，请修改 OfflinePushInfoConfig.java 文件 。

```
public static TUICallDefine.OfflinePushInfo createOfflinePushInfo(Context context) {
    TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
    pushInfo.setTitle("mike");
    pushInfo.setDesc("You have receive a new call");
    // OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
    pushInfo.setAndroidOPPOChannelID("tuikit");
    pushInfo.setIgnoreIOSBadge(false);
    pushInfo.setIOSSound("phone_ringing.mp3");
    pushInfo.setAndroidSound("phone_ringing"); //Note：don't add suffix
    //VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
```

```
    pushInfo.setAndroidVIVOClassification(1);
    //FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
    pushInfo.setAndroidFCMChannelID("fcm_push_channel");
    //Huawei message type
    pushInfo.setAndroidHuaWeiCategory("IM");
    //IOS push type: if you want to use VoIP, please modify type to TUICallDefine.IOSOfflinePushType.VoIP
    pushInfo.setIOSPushType(TUICallDefine.IOSOfflinePushType.APNs);
    return pushInfo;
}
```

步骤四：自定义离线通知铃音

离线通知铃声仅支持以下厂商自定义：**华为、小米、FCM 和 APNs**，其他厂商 OPPO、VIVO、荣耀等暂不支持。
TUICallKit 引入了 TUIOfflinePushPush 组件，需在应用启动时，调用以下方法开启离线通知自定义铃音能力。

Kotlin

```
class DemoApplication : Application() {
    override fun onCreate() {
        TUIOfflinePushConfig.getInstance().isAndroidPrivateRing = true
    }
}
```

Java

```
public class DemoApplication extends Application {
    @Override
    public void onCreate() {
        TUIOfflinePushConfig.getInstance().setAndroidPrivateRing(true);
    }
}
```

1、华为 & APNs

调用接口 setAndroidSound() 和 setIOSSound()。
定制铃音资源文件，将铃音文件添加到本地 Android Studio 工程 res/raw 目录下。参考 [步骤三] 设置铃声。

Kotlin

```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
```

Java

```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
```

> **注意**
> - IMSDK 6.1.2155 及以上版本支持。
> - 华为推送消息的铃声为首次创建通知渠道时的铃声，之后再次设置其他铃声无效，若想更新自定义通知铃音需卸载重装，详见：华为自定义铃声 。

2、小米

（1）Android 8.0 之前，调用setAndroidSound() 和 setIOSSound()。定制铃音资源文件，将铃音文件添加到工程 res/raw 目录下（参考上述华为调用）。
（2）Android 8.0 之后，还需要登录小米厂商控制台并 创建channel并做好配置 ，其中铃音文件需要添加到本地 Android Studio 工程的 res/raw 目录下。

Kotlin

```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
pushInfo.androidXiaoMiChannelID = "厂商申请的channelID"
```

Java

```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
pushInfo.setAndroidXiaoMiChannelID("厂商申请的channelID");
```



> ⚠ 注意
> - Android 8.0之前的方式，IMSDK 6.1.2155 及以上版本支持。
> - Android 8.0之后的方式，IMSDK 7.0.3754 及以上版本支持。

3、FCM

（1）Android 8.0 之前，调用 setAndroidSound() 和 setIOSSound()。定制铃音资源文件，将铃音文件添加到工程 res/raw 目录下（参考上述华为调用）。
（2）Android 8.0之后，FCM 需要配置 channelID 以及铃音资源，TUICallKit 引入的 tuiofflinepush 组件已经处理了自定义铃声的播放。铃音文件需要添加到您本地 Android Studio 工程的 res/raw 目录下，并指定铃音名称和 channel ID 的名称，详见 PrivateConstants 。

```
public class PrivateConstants {
    // FCM 通道指定 channel ID
    public static String fcmPushChannelId = "FCM ChannelID";
    // FCM 需指定通道的推送铃音名称，与AndroidStudio 工程里 res/raw 目录中的铃音文件名一致，不需要后缀名
    public static String fcmPushChannelSoundName = "铃声名称";
}
```

Kotlin

```
val pushInfo = OfflinePushInfo()
pushInfo.iosSound = "铃声名称.mp3"
pushInfo.androidSound = "铃声名称"
pushInfo.androidFCMChannelID = "厂商申请的channelID"
```

Java

```
TUICallDefine.OfflinePushInfo pushInfo = new TUICallDefine.OfflinePushInfo();
pushInfo.setIOSSound("铃声名称.mp3");
pushInfo.setAndroidSound("铃声名称");
pushInfo.setAndroidFCMChannelID("厂商申请的channelID");
```

> ⚠ 注意
> - Android 8.0之前的方式，IMSDK 6.1.2155 及以上版本支持。
> - Android 8.0之后的方式，IMSDK 7.0.3754 及以上版本支持。
> - FCM 自定义铃声或者设置 channel id 仅支持证书模式。



集成效果

下图是 TUICallKit 含 UI 方案集成离线推送的效果（不同厂商展示效果可能会有差异）：



常见问题

收不到离线推送通知，请参见 TUIOfflinePush-常见问题 进行排查。

1、收不到通知

- 用厂商控制台进行推送测试，能成功说明厂商通道没有问题。再检查 TUIOfflinePush 控制台厂商参数配置是否正确，按要求进行填写。（经测试：vivo x9必须在控制台配置消息类别）。
- 部分手机收到通知会放到 `不重要的通知` 中，请下拉状态栏，检查是否归纳到 `不重要的通知` 中。
- 检查 TUIOfflinePush 注册是否成功，过滤以下日志：

```
TUIOfflinePush
```

2、锁屏时无法点亮屏幕

Android 手机由于厂商和平台的限制，在锁屏情况下需要不同的权限。请按以下情况进行排查。

- **确认打开厂商锁屏下通知权限**
  部分厂商统一做了约束，例如小米锁屏下离线通知到达时未亮屏：在**设置 > 锁屏**里，单击开关**锁屏来通知时亮屏**，打开开关。
- **确认打开应用锁屏通知权限**
  例如：小米需要锁屏显示权限。

> **说明**
> 如遇到该问题，需要做兼容处理，您可以加入我们的 QQ 群（**605115878**）进行咨询与反馈。

3、单击离线推送通知，拉不起通话界面

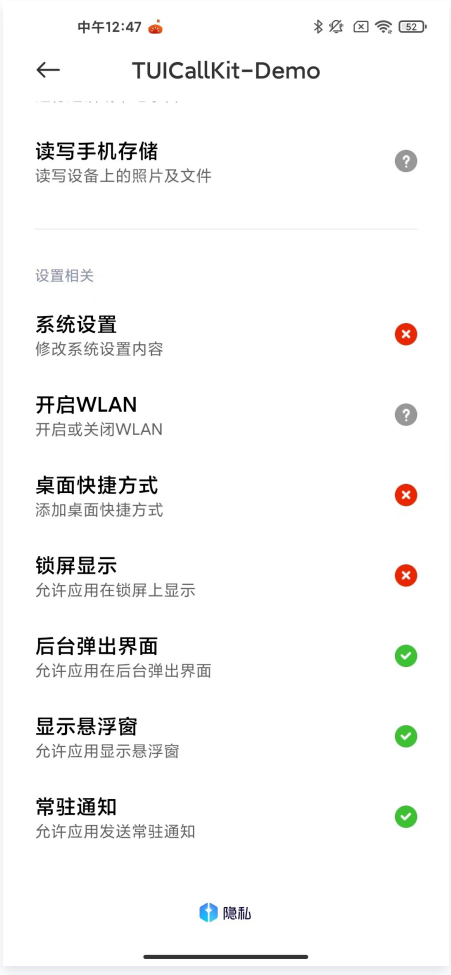
需要检查下是否收到了通话请求，可以过滤以下日志：

```
onReceiveNewInvitation
```

4、应用在后台时，不能自动将通话界面拉取到前台

将应用从后台自动拉取到前台，需要检查 App 是否开启了“后台自启动”或“悬浮窗”权限。
值得注意的是，不同厂商、甚至同一厂商不同 Android 版本，其对于应用开放的权限以及权限名称也会存在不一致。例如，小米6只需要开启**后台弹出界面**权限，而红米需要同时打开**后台弹出界面**和**显示悬浮窗**权限。

> **说明**
> - 在 TUICallKit 含 UI 方案中，开启悬浮窗的时候，会跳转手机的设置界面，引导用户开启悬浮窗和后台拉起应用权限；您可以根据自己的业务需求，提前引导用户开启该权限。
> - 如果您在测试过程中发现手动开启了所有权限，依然无法自动拉起通话界面到前台，需要做兼容处理。可以加 QQ 群：**605115878**，联系我们协助处理。

OPPO 手机的权限设置位置	小米手机的权限设置位置
	



5、角标问题

目前仅 华为 支持收到离线消息时生成角标，其他厂商暂不支持。华为设置角标方法详见其官方文档：华为桌面角标开发指导 。
在引入的 TUIOfflinePush 组件中，也调用了该方法进行设置：

Kotlin

```
fun updateBadge(context: Context, number: Int) {
    if (BrandUtil.getInstanceType() != TUIOfflinePushConfig.BRAND_HUAWEI) {
        return
    }
    try {
        val extra = Bundle()
        extra.putString("package", context.getPackageName())
        extra.putString("class", PrivateConstants.huaweiBadgeClassName) //huaweiBadgeClassName is your
app main activity
        extra.putInt("badgenumber", number)
        context.getContentResolver().call(Uri.parse("content://com.huawei.android.launcher.settings/badge/"),
"change_badge", null, extra)
    } catch (e: Exception) {
    }
}
```

Java

```
public void updateBadge(final Context context, final int number) {
    if (BrandUtil.getInstanceType() != TUIOfflinePushConfig.BRAND_HUAWEI) {
        return;
    }
    try {
        Bundle extra = new Bundle();
        extra.putString("package", context.getPackageName());
        extra.putString("class", PrivateConstants.huaweiBadgeClassName); //huaweiBadgeClassName is your
app main activity
        extra.putInt("badgenumber", number);
        context.getContentResolver().call(Uri.parse("content://com.huawei.android.launcher.settings/badge/"),
"change_badge", null, extra);
    } catch (Exception e) {
    }
}
```

如果想要清除角标，可以在适当的时机调用该方法，将角标设置为0。

技术咨询

如有其他问题，欢迎您加入我们的 TUIKit 组件交流群 QQ 群：605115878，进行技术交流和产品沟通。

iOS
APNs

最近更新时间：2023-05-09 09:36:35

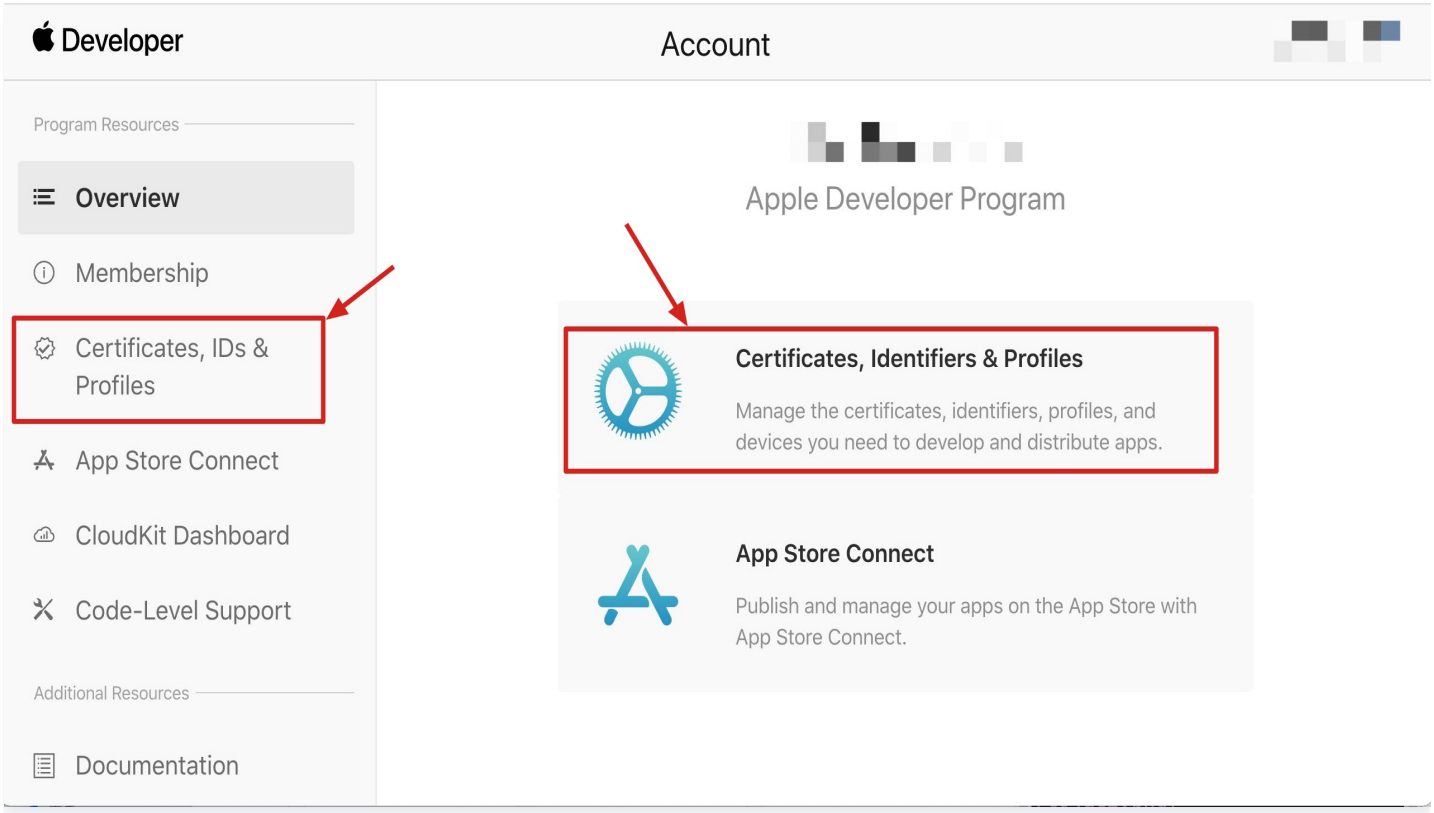
离线唤醒功能，能够让您的 App 在后台运行或者离线状态下依然能够收到音视频通话的响铃呼叫，TUICallKit 使用 Apple 提供的系统级推送通道（APNs）来进行消息通知。

配置离线推送

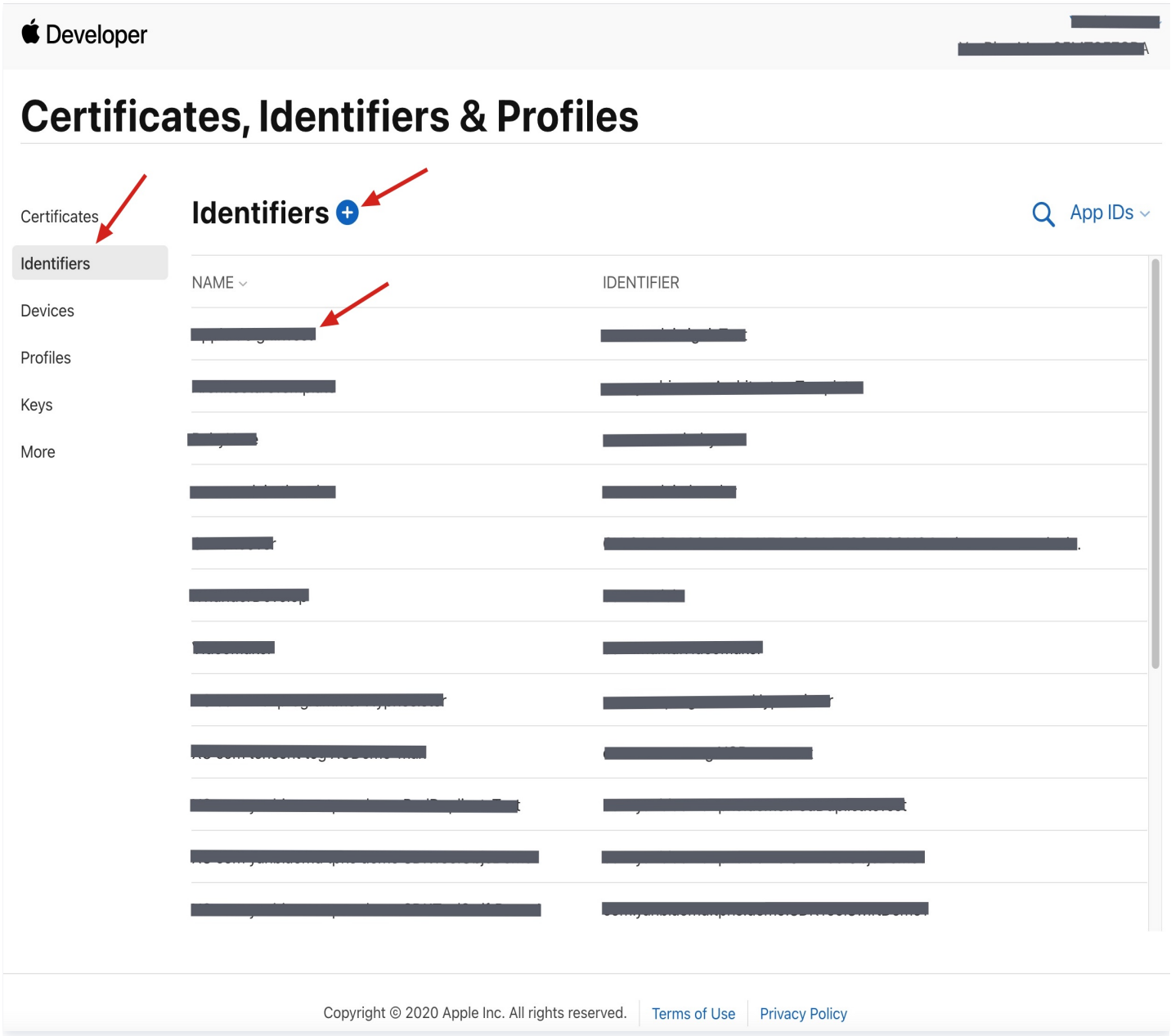
1. 开启 App 远程推送。
2. 生成证书。
3. 上传证书到 IM 控制台。
4. 完成工程配置
5. 在 App 每次登录时，向苹果获取 deviceToken。
6. 调用 setAPNS 接口将其上报到 IM 后台。

步骤1. 开启 App 远程推送

1. 登录 苹果开发者中心 网站，单击 ” Certificates,Identifiers & Profiles” 或者侧栏的 ” Certificates, IDs & Profiles” ，进入 Certificates, IDs & Profiles 页面。



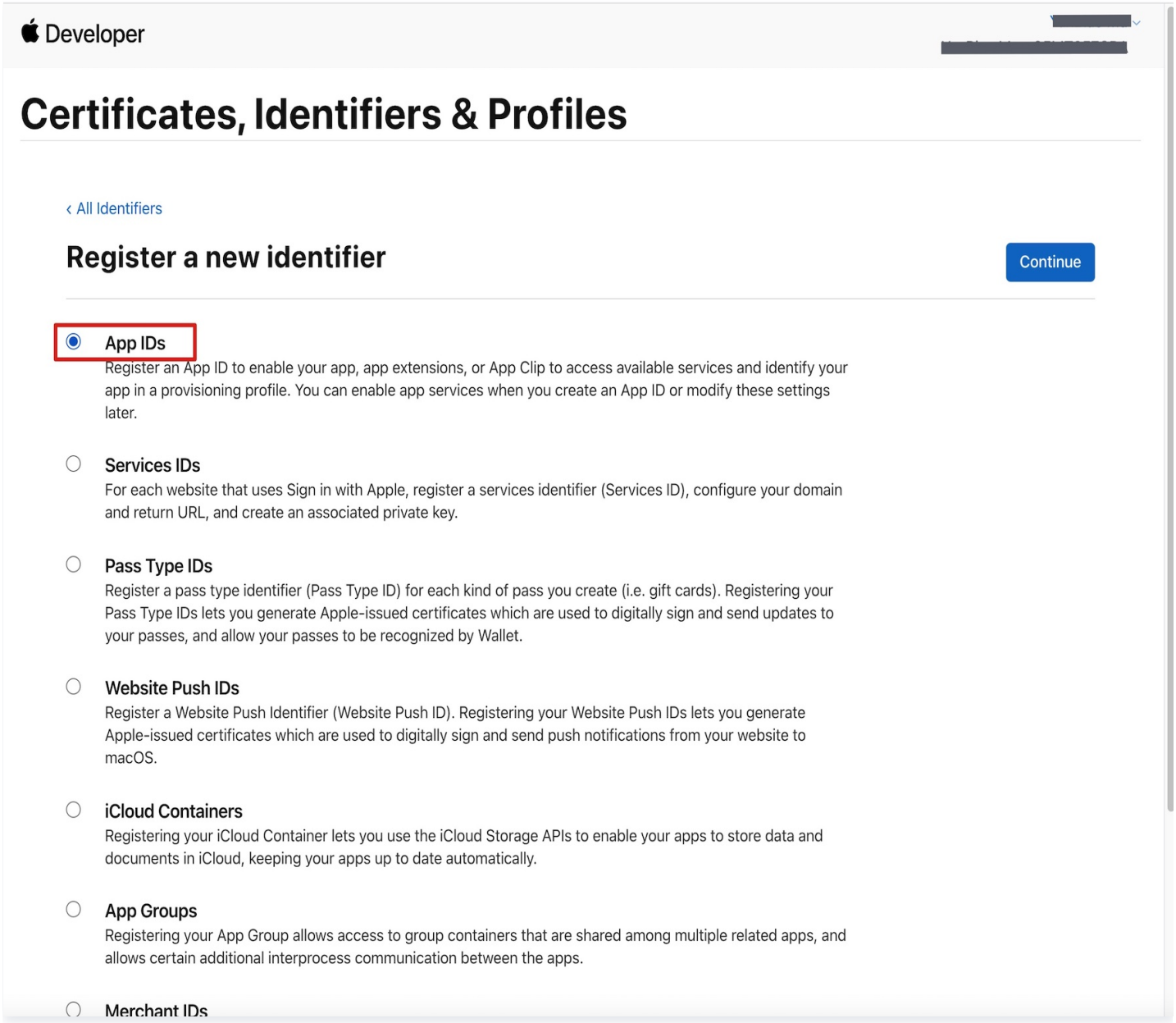
2. 单击 Identifiers 右侧的“+ “。



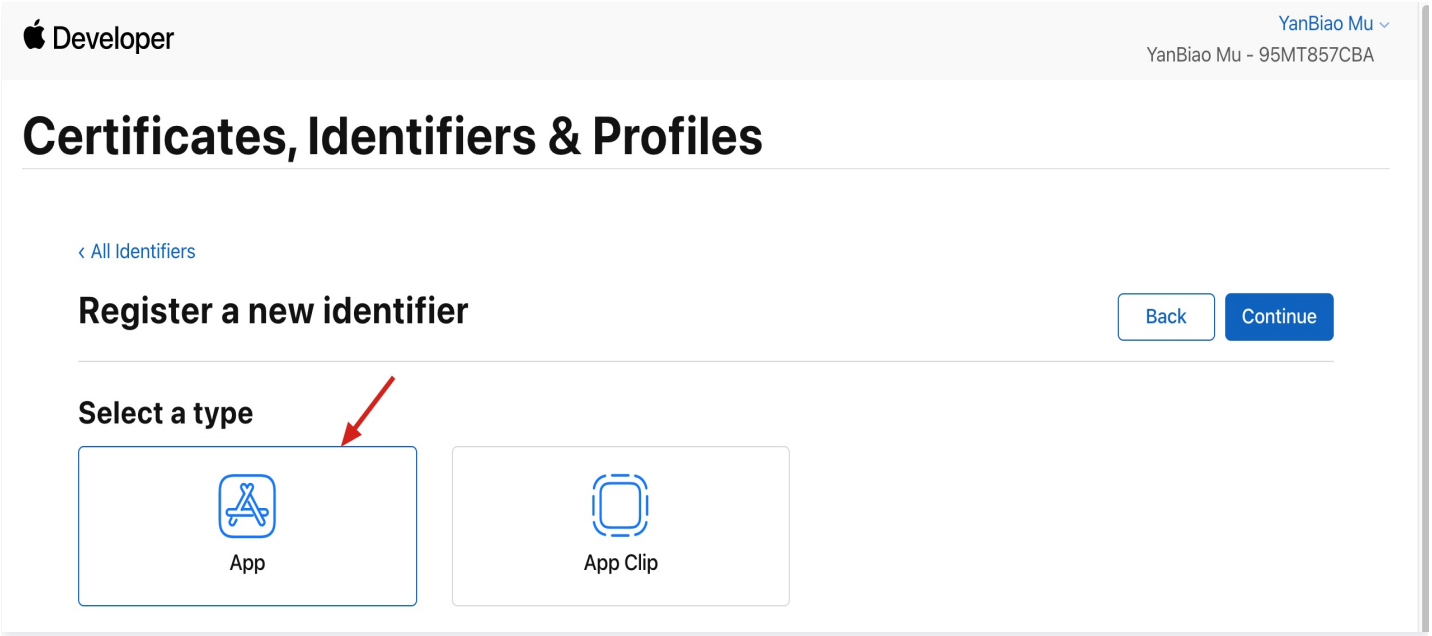
3. 您可以参考如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **说明：**
> 需要注意的是，您 App 的 `Bundle ID` 不能使用通配符 *，否则将无法使用远程推送服务。

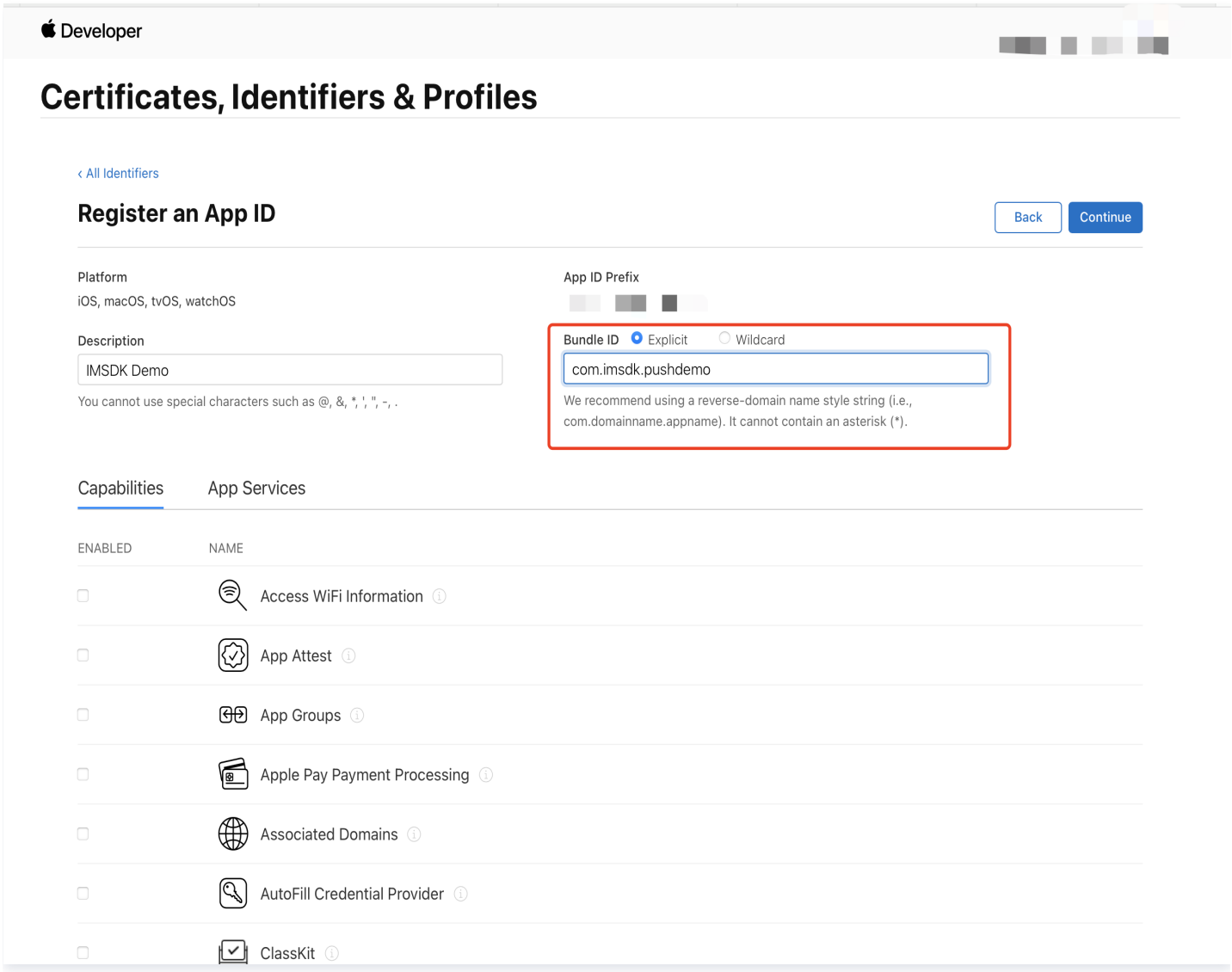
4. 勾选 “App IDs”，单击 ”Continue” 进行下一步。



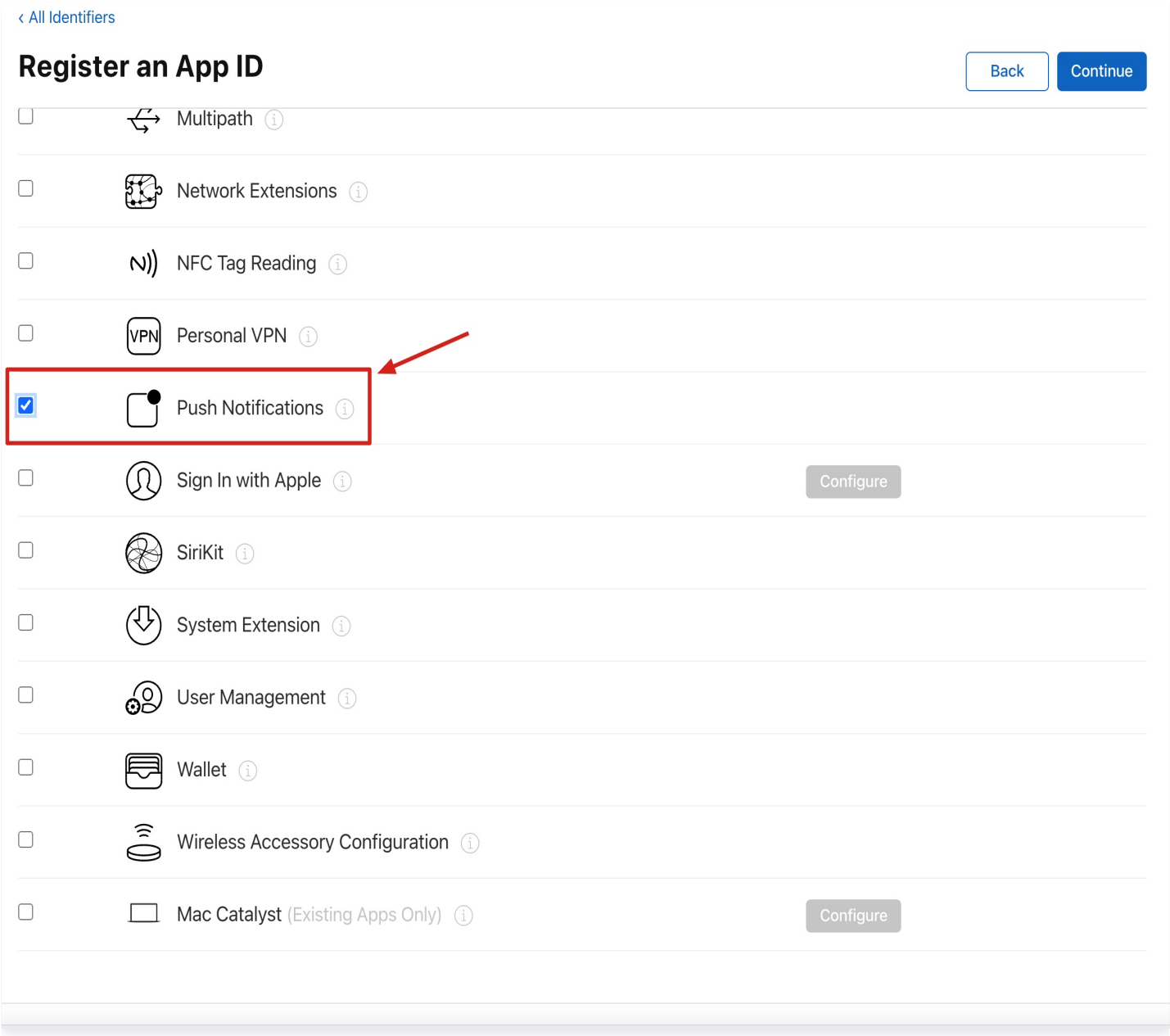
5. 选择 “App”，单击 ”Continue“ 进行下一步。



6. 配置 Bundle ID 等其他信息，单击 “Continue“ 进行下一步。

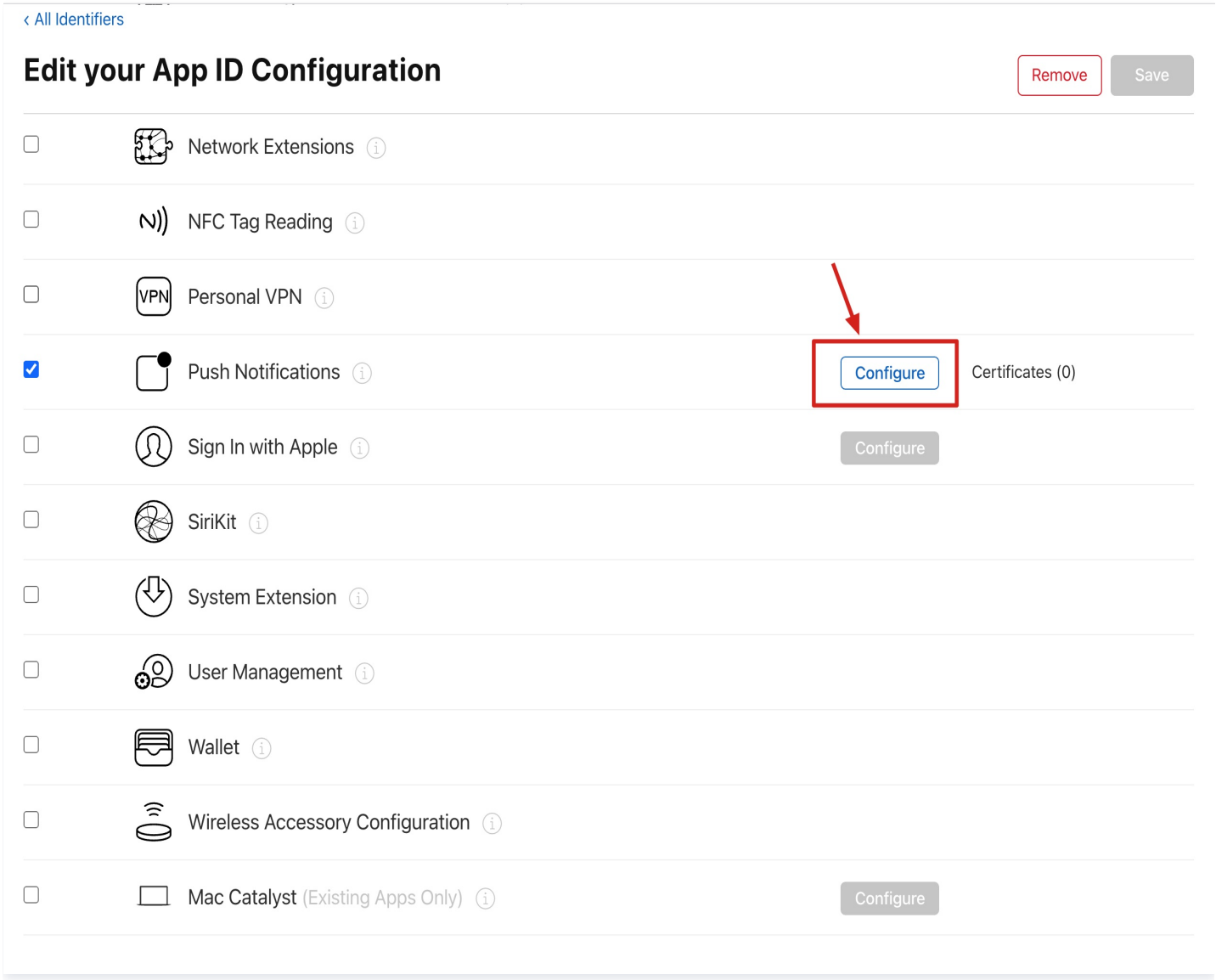


7. 勾选 “Push Notifications”，开启远程推送服务。

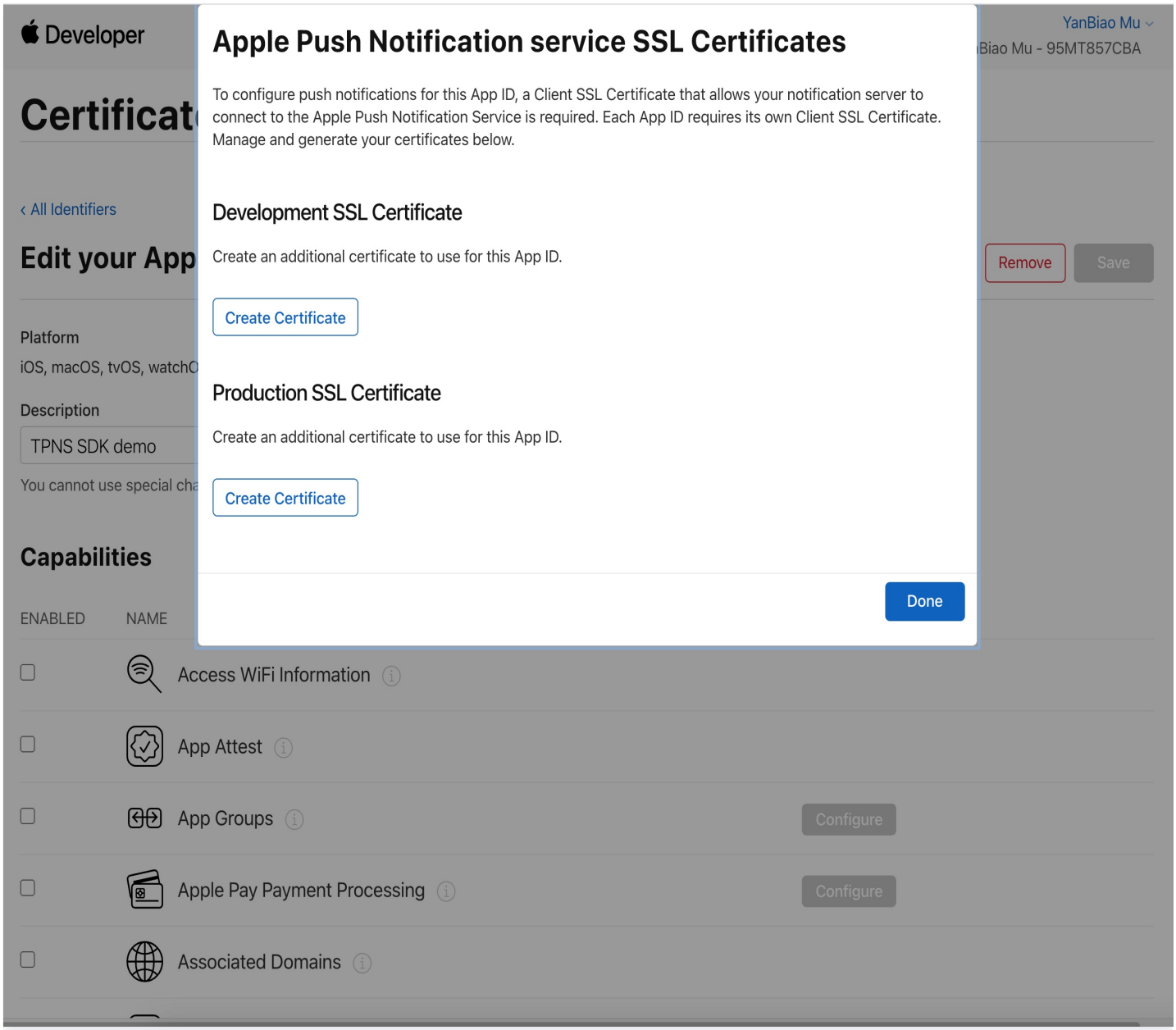


步骤2：生成证书

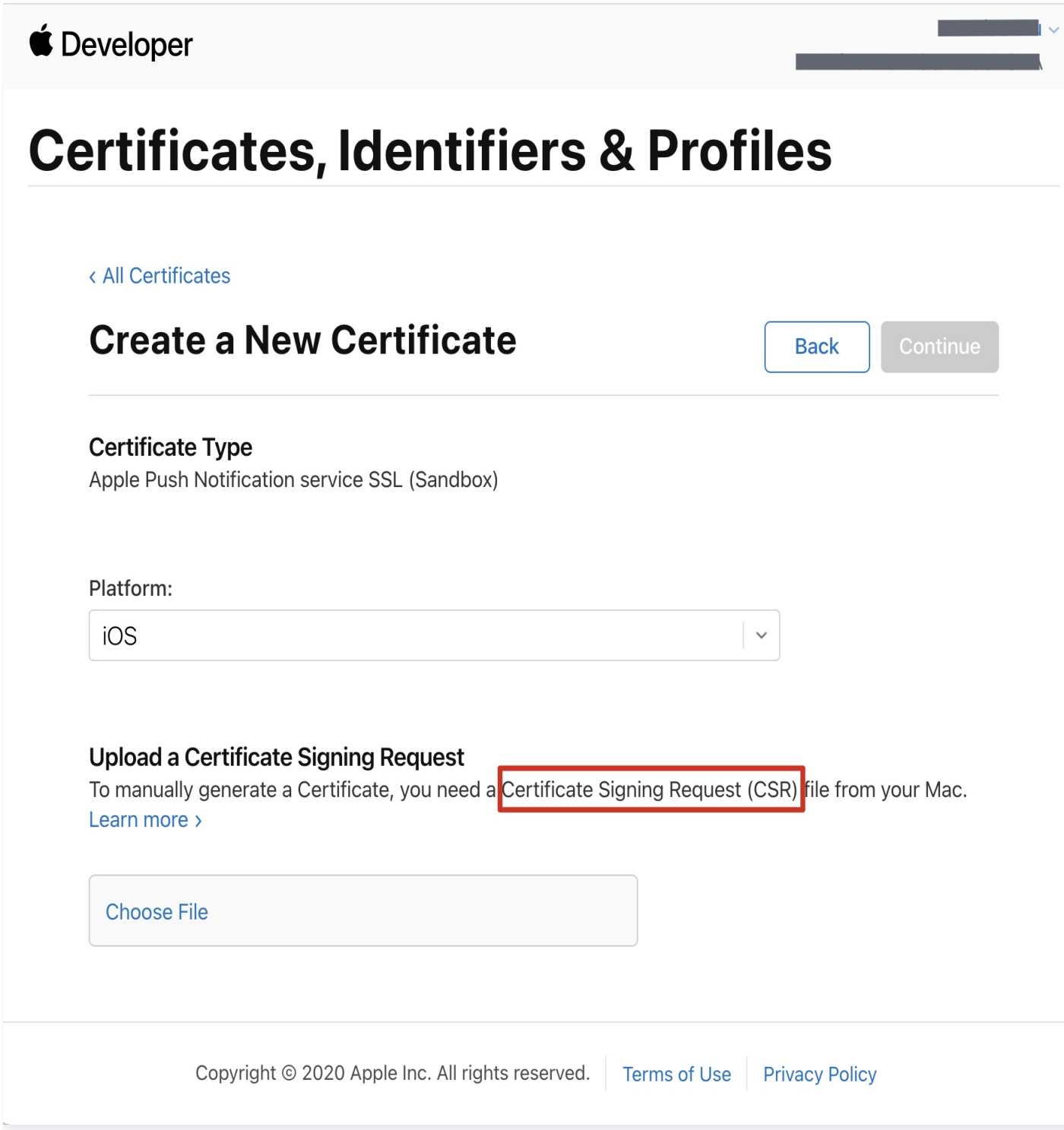
1. 选中您的 AppID，选择 “Configure “。



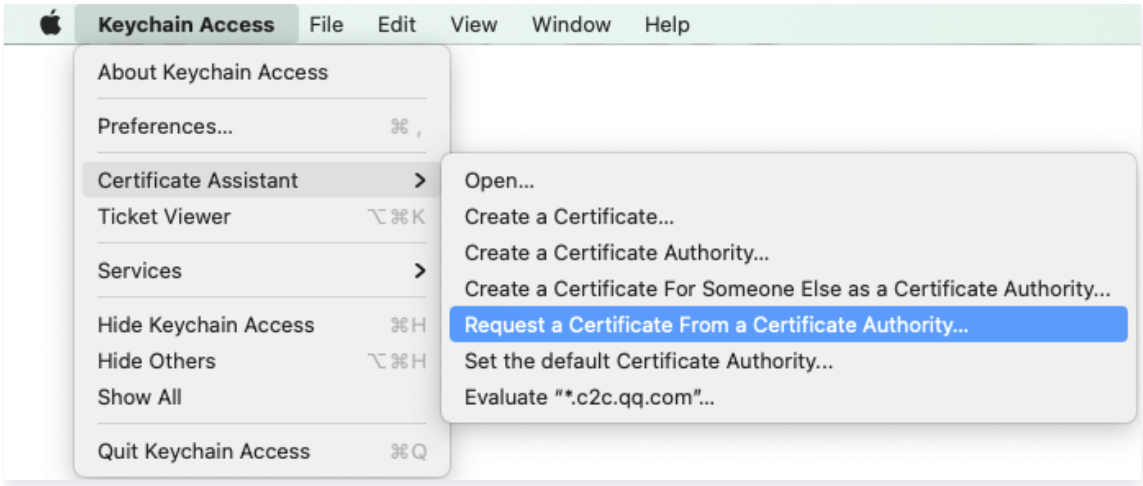
2. 可以看到在 “Apple Push Notification service SSL Certificates” 窗口中有两个 `SSL Certificate` ，分别用于开发环境（ `Development` ）和生产环境（ `Production` ）的远程推送证书，如下图所示：



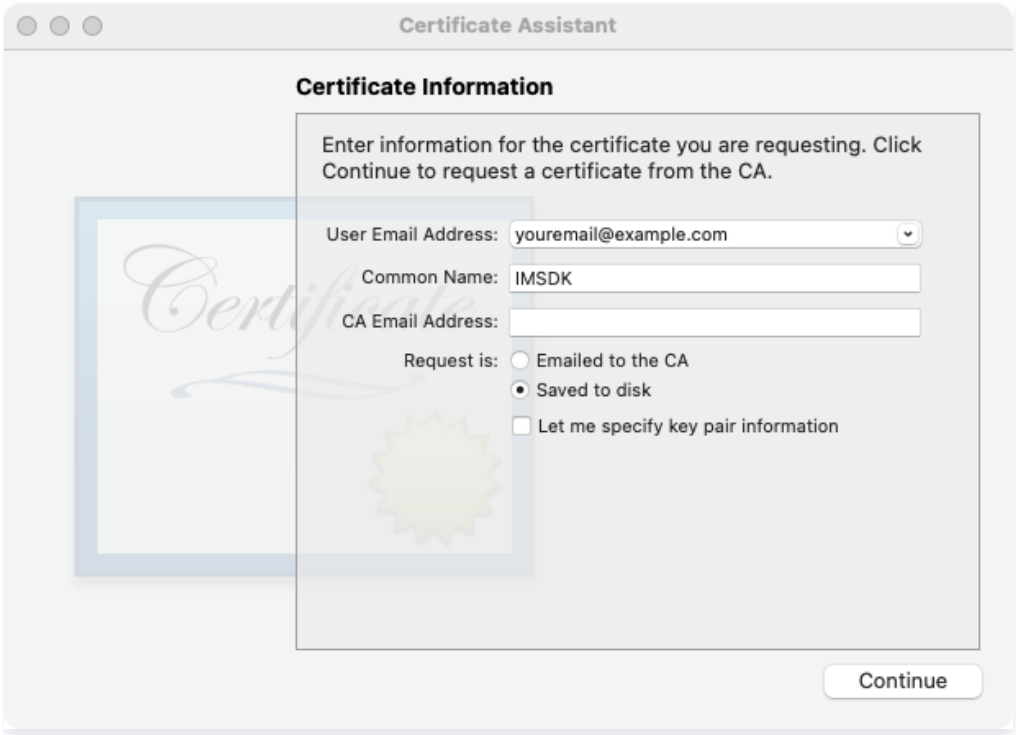
3. 我们先选择开发环境（ Development ）的 “Create Certificate”，系统将提示我们需要一个 Certificate Signing Request（CSR） 。



4. 在 Mac 上打开钥匙串访问工具（Keychain Access），在菜单中选择 ”钥匙串访问” > “证书助理 “ > “从证书颁发机构请求证书 “（ Keychain Access > Certificate Assistant > Request a Certificate From a Certificate Authority ）。



5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择 “存储到磁盘（Saved to disk）”，单击继续，系统将生成一个 *.certSigningRequest 文件。



6. 返回上述 第3步 中 Apple Developer 网站刚才的页面，单击 ” Choose File “ 上传生成的 *.certSigningRequest 文件。

Developer

Certificates, Identifiers & Profiles

‹ All Certificates

Create a New Certificate

Back Continue

Certificate Type
Apple Push Notification service SSL (Sandbox)

Platform:
iOS

Upload a Certificate Signing Request
To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac.
Learn more ›

Choose File

Copyright © 2020 Apple Inc. All rights reserved. Terms of Use Privacy Policy

7. 单击 ”Continue “，即可生成推送证书。

Developer

Certificates, Identifiers & Profiles

‹ All Certificates

Create a New Certificate

Back Continue

Certificate Type
Apple Push Notification service SSL (Sandbox)

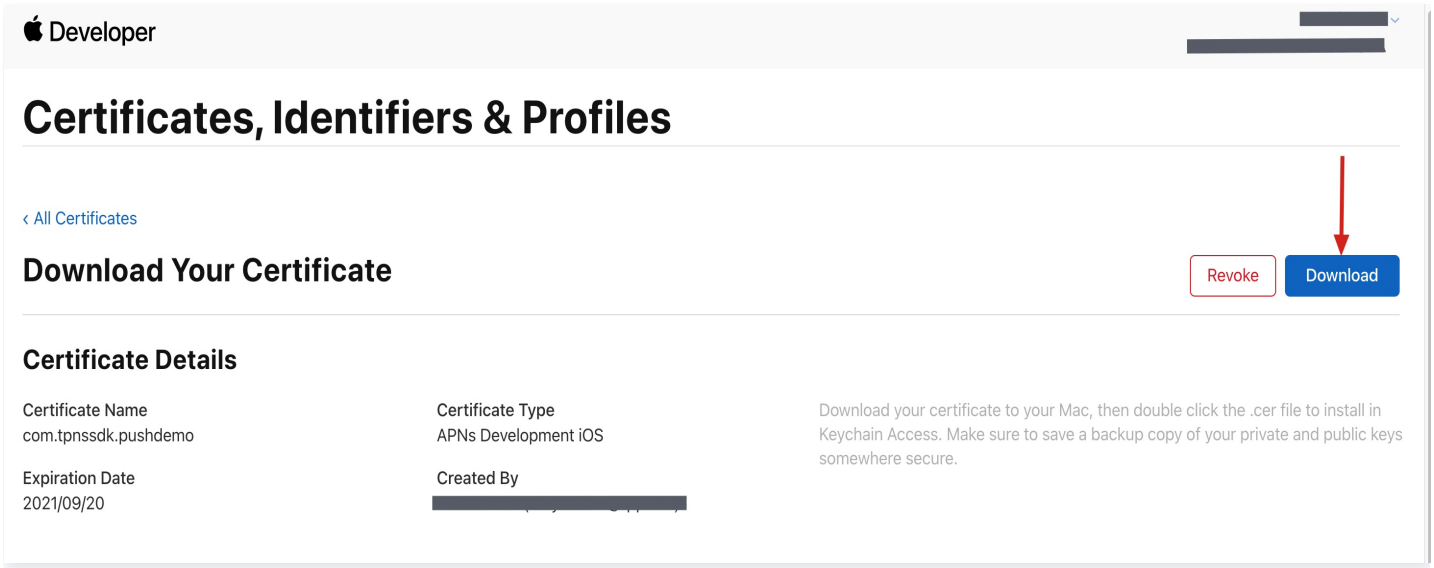
Platform:
iOS

Upload a Certificate Signing Request
To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac.
Learn more ›

Choose File CertificateSigningRequest.certSigningRequest

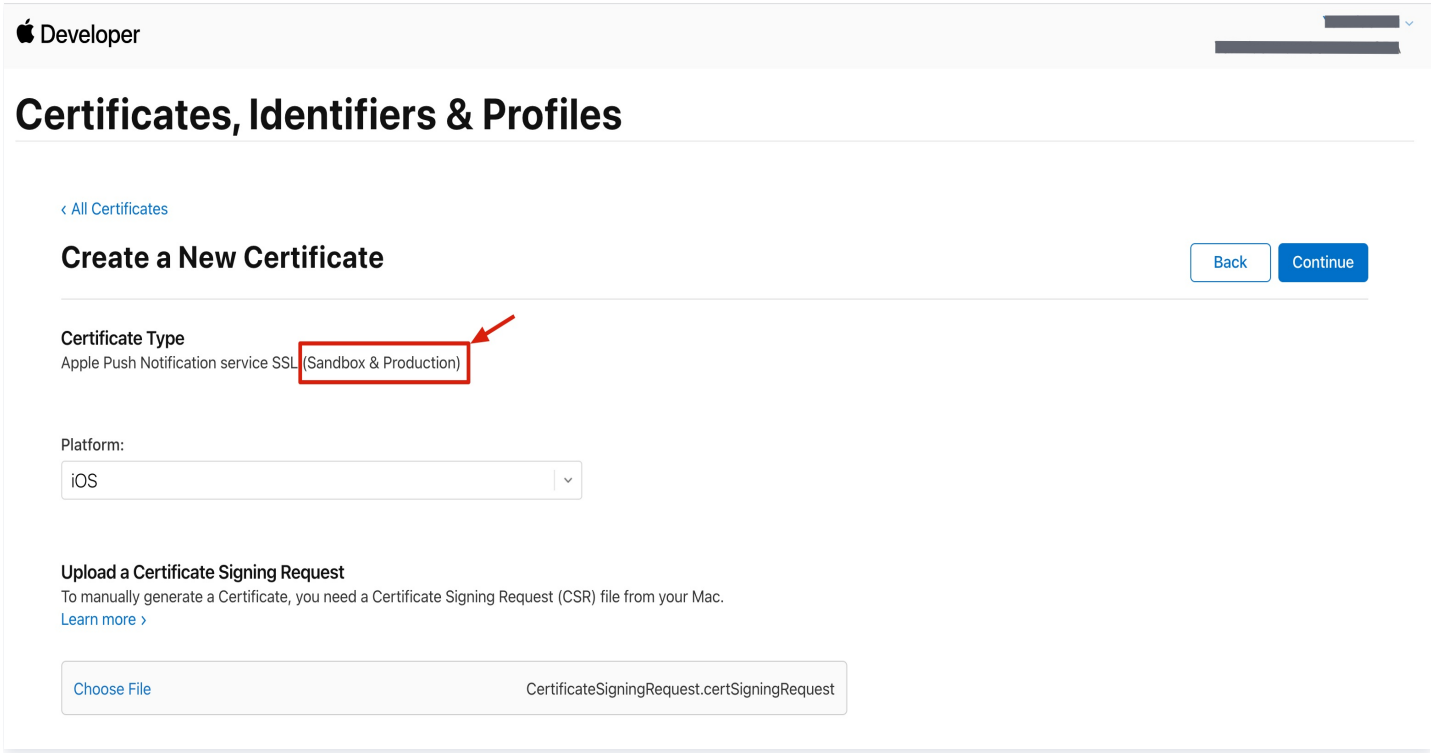
Copyright © 2020 Apple Inc. All rights reserved. Terms of Use Privacy Policy

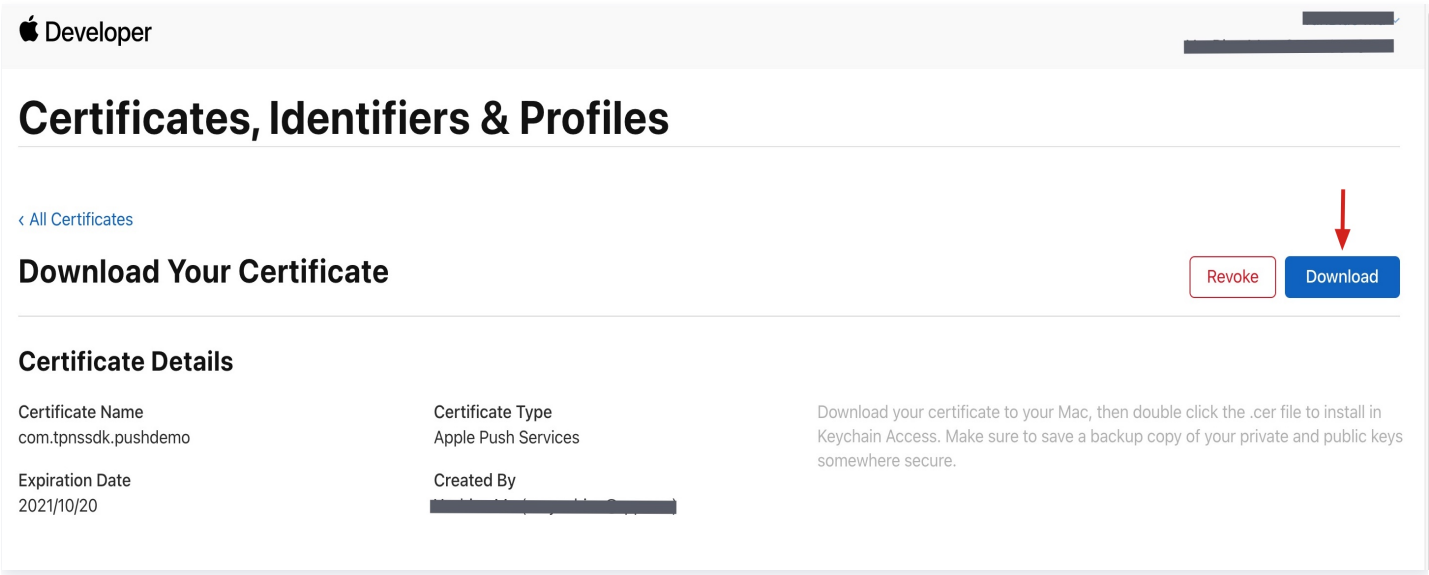
8. 单击 “Download“ 下载开发环境的 `Development SSL Certificate` 到本地。



9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **说明：**
> 生产环境的证书实际是开发(`Sandbox`)+生产(`Production`)的合并证书，可以同时作为开发环境和生产环境的证书使用。



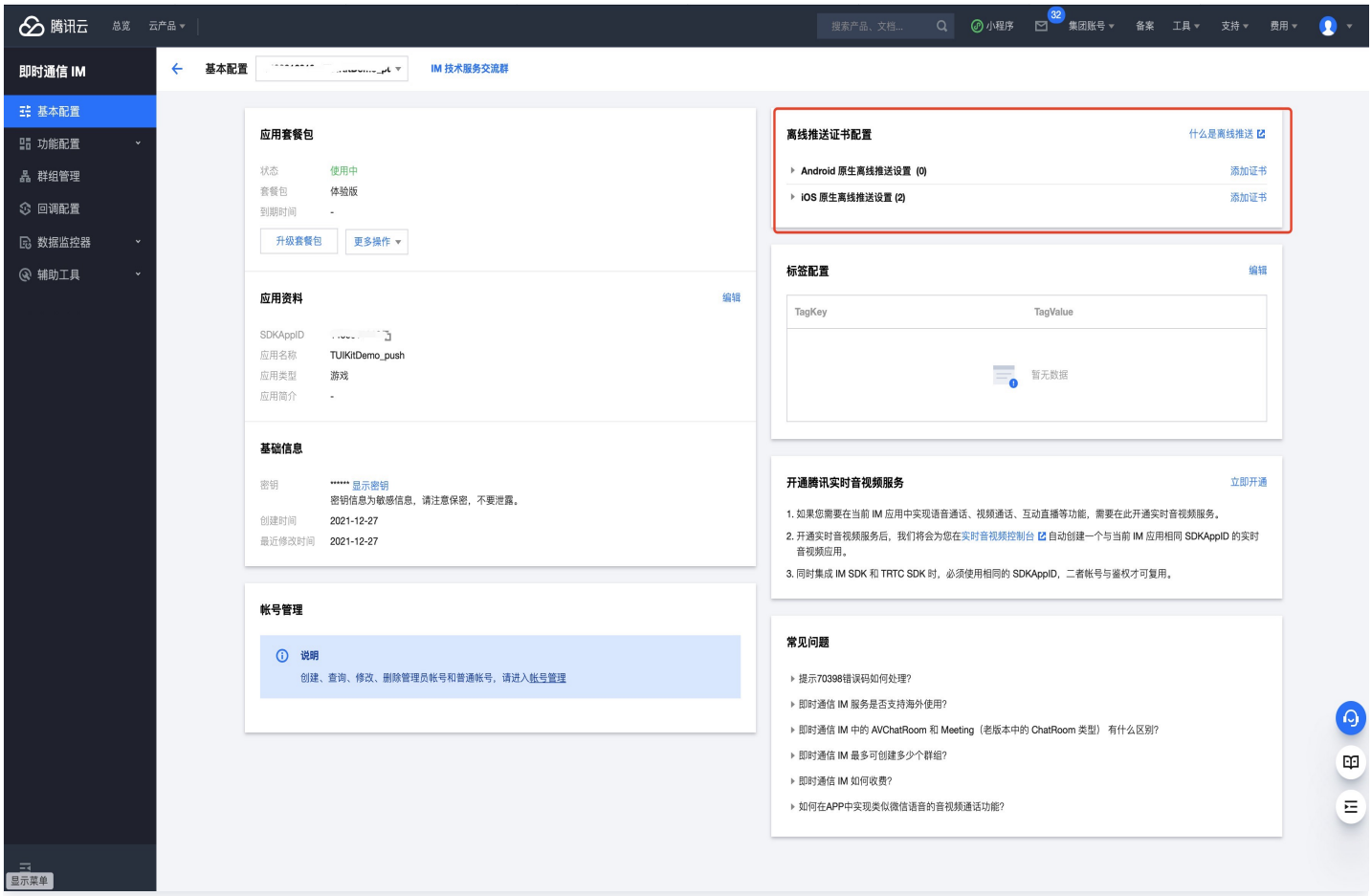


10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在“登录“ > “我的证书”，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 P12 文件。

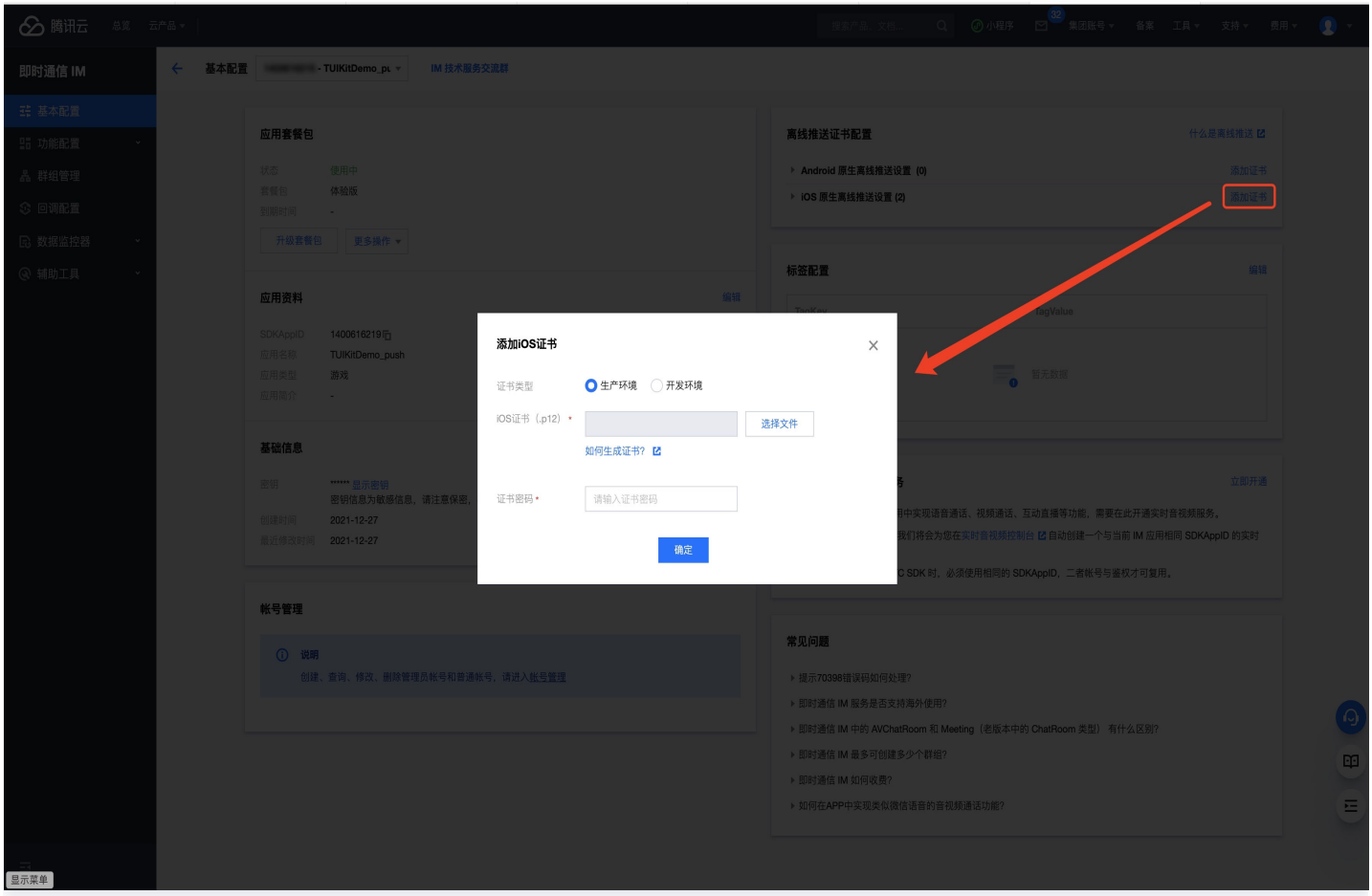
> **说明：**
> 保存 `P12` 文件时，请务必要为其设置密码。

步骤3：上传证书到 IM 控制台

1. 登录 即时通信 IM 控制台 。
2. 单击目标应用卡片，进入应用的基础配置页面。



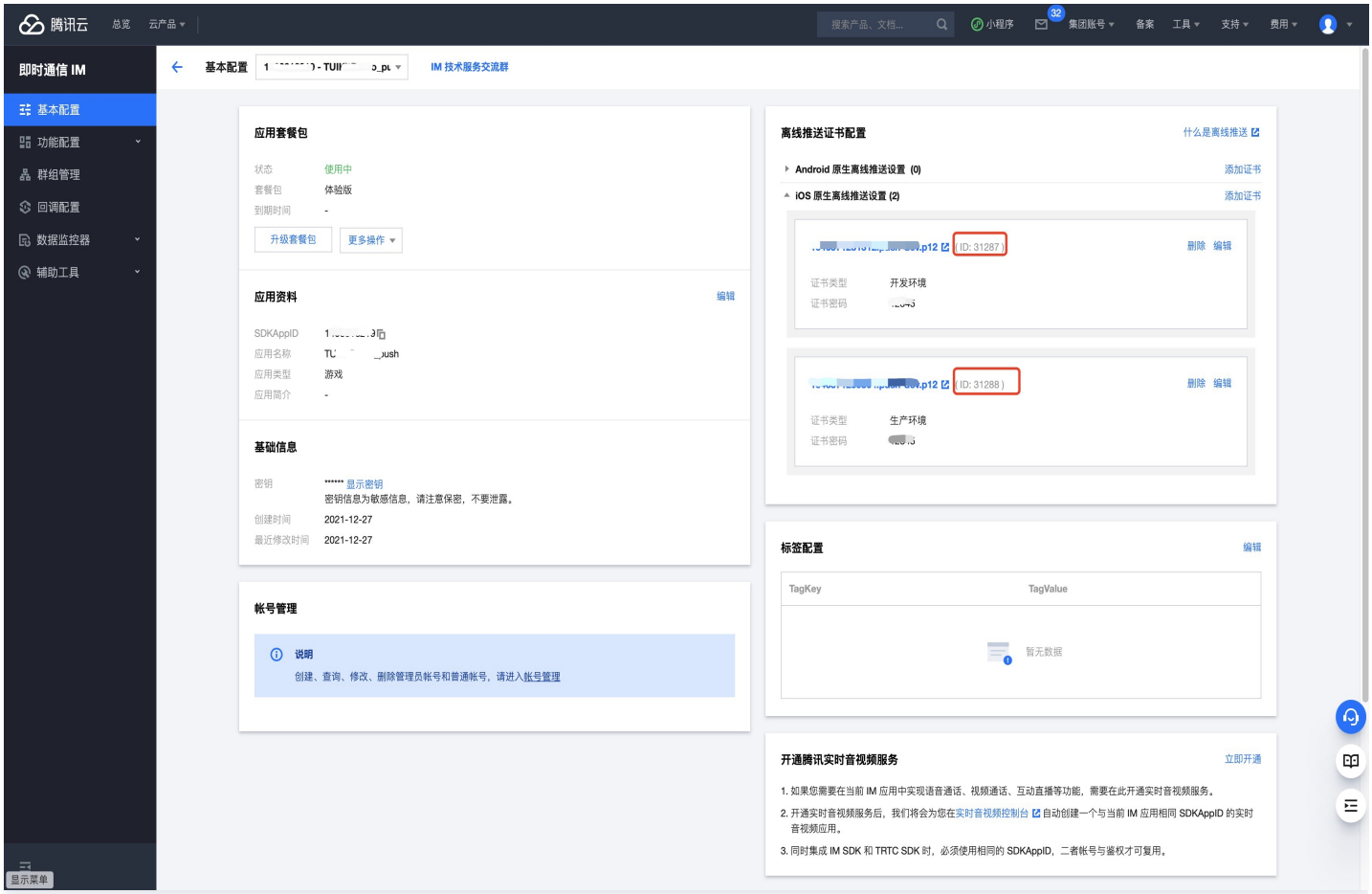
3. 单击 “iOS 原生离线推送设置“ 右侧的 “添加证书”。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击【确认】。



说明：

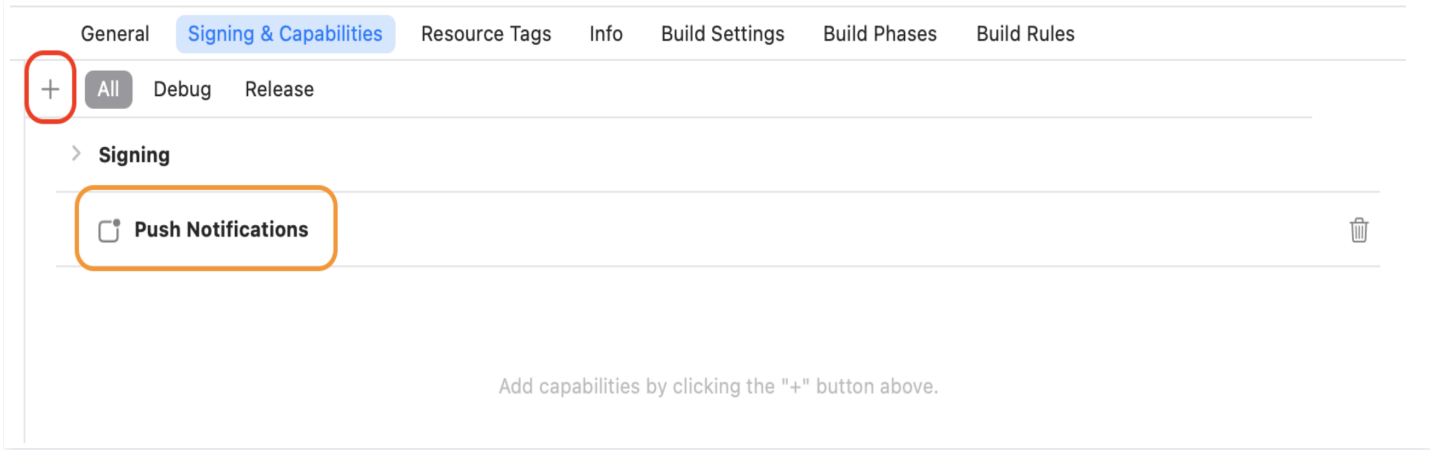
- 上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。
- 上传证书需要设置密码，无密码收不到推送。
- 发布 App Store 的证书需要设置为生产环境，否则无法收到推送。
- 上传的 p12 证书必须是自己申请的真实有效的证书。

5. 待推送证书信息生成后，记录证书的 ID。



步骤4：完成工程配置

要在应用程序中添加所需的权限，请在 Xcode 项目中启用推送通知功能。
打开 Xcode 项目，在 **Project > Target > Capabilities** 页面中点击红框中的加号按钮，然后选择并添加 **Push Notifications**，添加后的结果如图中黄框所示：



步骤5：在 App 每次登录时，向苹果获取 deviceToken

您可以在您的 App 中添加如下代码，用来向苹果的后台服务器获取 deviceToken：

> **说明：**
> 考虑到合规，建议您在用户同意隐私协议之后再向苹果请求 deviceToken。

```
// 向苹果后台请求 DeviceToken
- (void)registerNotification {
    [[UIApplication sharedApplication] registerUserNotificationSettings:
            [UIUserNotificationSettings settingsForTypes:
            (UIUserNotificationTypeSound | UIUserNotificationTypeAlert | UIUserNotificationTypeBadge)
            categories:nil]];
    [[UIApplication sharedApplication] registerForRemoteNotifications];
}

// 在 AppDelegate 的回调中会返回 deviceToken，需要在登录后上报给腾讯云后台
-(void)application:(UIApplication *)app didRegisterForRemoteNotificationsWithDeviceToken:(NSData
*)deviceToken {
    // 记录下 Apple 返回的 deviceToken
    _deviceToken = deviceToken;
}
```

步骤6：调用 接口将其上报到 IM 后台

在 IM SDK 登录成功后，就可以调用 setAPNS 接口，将 步骤5 中获取的 deviceToken 上传到腾讯云后台，示例代码如下：

```
- (void)pushRegisterIfLogined {
  if (self.deviceToken) {
      V2TIMAPNSConfig *config = [[V2TIMAPNSConfig alloc] init];
      // 用户自己到苹果注册开发者证书，在开发者帐号中下载并生成证书(p12 文件)，将生成的 p12 文件传到腾讯证书管理
控制台，
      // 控制台会自动生成一个证书 ID，将证书 ID 传入以下 businessID 参数中。
      config.businessID = sdkBusiId;
      // 苹果后台服务器获取的 deviceToken
      config.token = self.deviceToken;
      [[V2TIMManager sharedInstance] setAPNS:config succ:^{
          NSLog(@"%s, succ", __func__");
      } fail:^(int code, NSString *msg) {
          NSLog(@"%s, fail, %d, %@", __func__, code, msg);
      }];
  }
  // ...
}
```

自定义离线铃声

请在调用 call 拨打电话的时候设置 params 的 offlinePushInfo 中的 iOSSound 字段， iOSSound 传语音文件名。

> **注意：**
> - 离线推送声音设置（仅对 iOS 生效），如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀名）设置给 iOSSound。
> - 铃声时长应小于30s。

Objective-C

```
[[TUICallKit createInstance] call:@"mike 的 id" params:[self getCallParams]
callMediaType:TUICallMediaTypeVideo];

- (TUICallParams *)getCallParams {
    TUIOfflinePushInfo *offlinePushInfo = [self createOfflinePushInfo];
    TUICallParams *callParams = [TUICallParams new];
    callParams.offlinePushInfo = offlinePushInfo;
    callParams.timeout = 30;
    return callParams;
}

+ (TUIOfflinePushInfo *)createOfflinePushInfo {
    TUIOfflinePushInfo *pushInfo = [TUIOfflinePushInfo new];
    pushInfo.title = @"";
    pushInfo.desc = TUICallingLocalize(@"TUICallKit.have.new.invitation");
    pushInfo.iOSPushType = TUICallIOSOfflinePushTypeAPNs;
    pushInfo.ignoreIOSBadge = NO;
    pushInfo.iOSSound = @"phone_ringing.mp3";
    pushInfo.AndroidSound = @"phone_ringing";
    // OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
    // OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
    pushInfo.AndroidOPPOChannelID = @"tuikit";
    // FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
    pushInfo.AndroidFCMChannelID = @"fcm_push_channel";
    // VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
    pushInfo.AndroidVIVOClassification = 1;
    // HuaWei message type: https://developer.huawei.com/consumer/cn/doc/development/HMSCore-
Guides/message-classification-0000001149358835
    pushInfo.AndroidHuaWeiCategory = @"IM";
    return pushInfo;
}
```

Swift

```
let params = TUICallParams()
let pushInfo: TUIOfflinePushInfo = TUIOfflinePushInfo()
pushInfo.title = "TUICallKit"
pushInfo.desc = "TUICallKit.have.new.invitation"
pushInfo.iOSPushType = .apns
pushInfo.ignoreIOSBadge = false
pushInfo.iOSSound = "phone_ringing.mp3"
pushInfo.androidSound = "phone_ringing"
// OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
// OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
pushInfo.androidOPPOChannelID = "tuikit"
// FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
pushInfo.androidFCMChannelID = "fcm_push_channel"
// VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
```

版权所有：腾讯云计算（北京）有限责任公司

```
pushInfo.androidVIVOClassification = 1
// HuaWei message type: https://developer.huawei.com/consumer/cn/doc/development/HMSCore-Guides/message-classification-0000001149358835
pushInfo.androidHuaWeiCategory = "IM"
params.userData = "User Data"
params.timeout = 30
params.offlinePushInfo = pushInfo
TUICallKit.createInstance().call(userId: "123456", callMediaType: .audio, params: params) {

} fail: {
    code, message in
}
```

常见问题

1、收不到推送，且后台报错 bad devicetoken?

Apple 的 deviceToken 与当前编译环境有关。如果 登录 IMSDK 后上传 deviceToken 到腾讯云 所使用的证书 ID 和 token 不一致，就会报错。

- 如果使用的是 Release 环境编译，则 `- application:didRegisterForRemoteNotificationsWithDeviceToken:` 回调返回的是发布环境的 token，此时 businessID 需要设置生产环境的 [证书 ID](#businessid :缺少内容)。
- 如果使用的是 Debug 环境编译，则 `- application:didRegisterForRemoteNotificationsWithDeviceToken:` 回调返回的是开发环境的 token，此时 businessID 需要设置开发环境的证书 ID。

Objective-C

```
V2TIMAPNSConfig *config = [[V2TIMAPNSConfig alloc] init];
// 用户自己到苹果注册开发者证书，在开发者帐号中下载并生成证书(p12 文件)，将生成的 p12 文件传到腾讯证书管理控制台，
// 控制台会自动生成一个证书 ID，将证书 ID 传入以下 businessID 参数中。*/
config.businessID = sdkBusiId;
config.token = self.deviceToken;
[[V2TIMManager sharedInstance] setAPNS:config succ:^{

} fail:^(int code, NSString *msg) {

}];
```

Swift

```
let config = V2TIMAPNSConfig()
config.businessID = sdkBusiId
config.token = deviceToken
V2TIMManager.sharedInstance().setAPNS(config) {

} fail: { code, message in

}
```

2、iOS 开发环境下，注册偶现不返回 deviceToken 或提示 APNs 请求 token 失败?

此问题现象是由于 APNs 服务不稳定导致的，可尝试通过以下方式解决：

1. 给手机插入 SIM 卡后使用 4G 网络测试。
2. 卸载重装、重启 App、关机重启后测试。
3. 打生产环境的包测试。
4. 更换其它 iOS 系统的手机测试。

版权所有：腾讯云计算（北京）有限责任公司

VoIP (可选)

最近更新时间：2023-12-26 17:09:43

VoIP（Voice over IP）Push 是 Apple 提供的用于响应 VoIP calls 的通知机制。结合 Apple 提供的 PushKit.framework 和 CallKit.framework 能够实现系统级的通话效果。目前 VoIP Push 暂时只支持离线状态下推送。

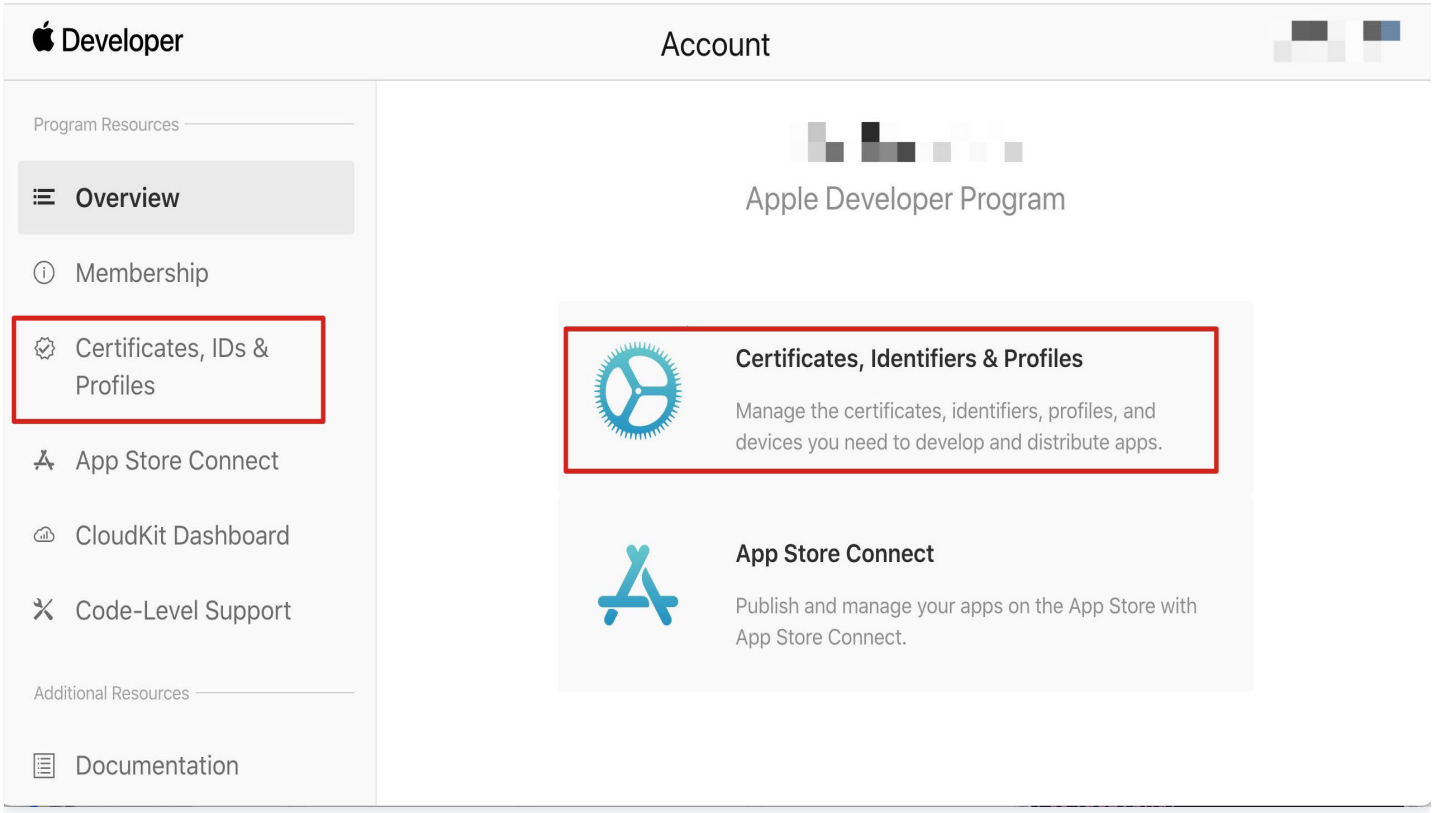
说明：
- Apple 要求，VoIP Push 在 iOS 13.0 以后需要搭配 CallKit.framework 使用，否则 App 运行后会 crash。
- VoIP Push 无法复用 APNs 普通推送证书，需要单独在苹果开发者网站上 申请 VoIP Push 证书 。

配置 VoIP Push

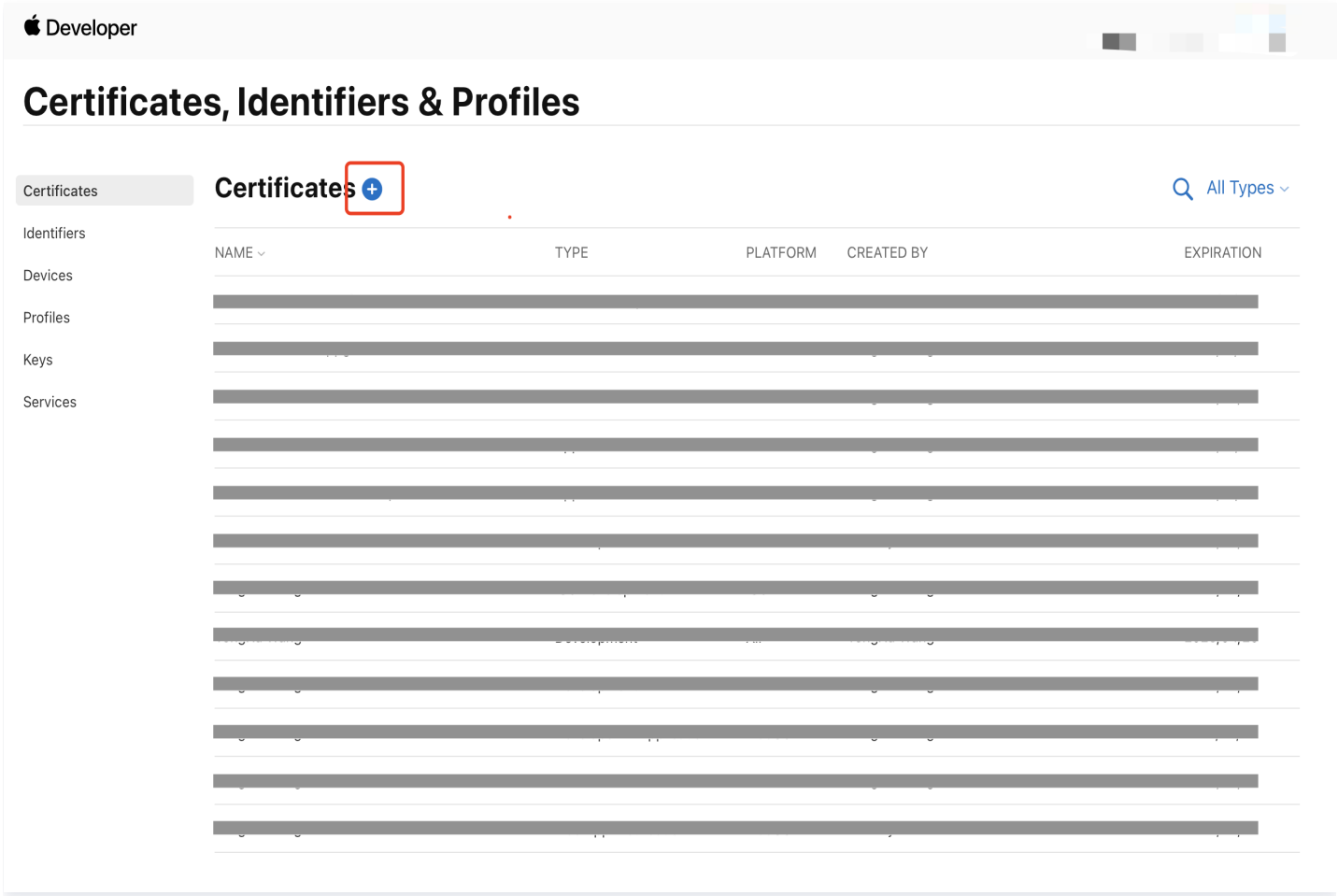
如果想要接收 VoIP Push，需要遵循如下几个步骤：
1. 申请 VoIP Push 证书。
2. 上传证书到 IM 控制台。
3. 完成工程配置。
4. 接入 TUICallKitVoIPExtension 组件。

步骤1：申请 VoIP Push 证书

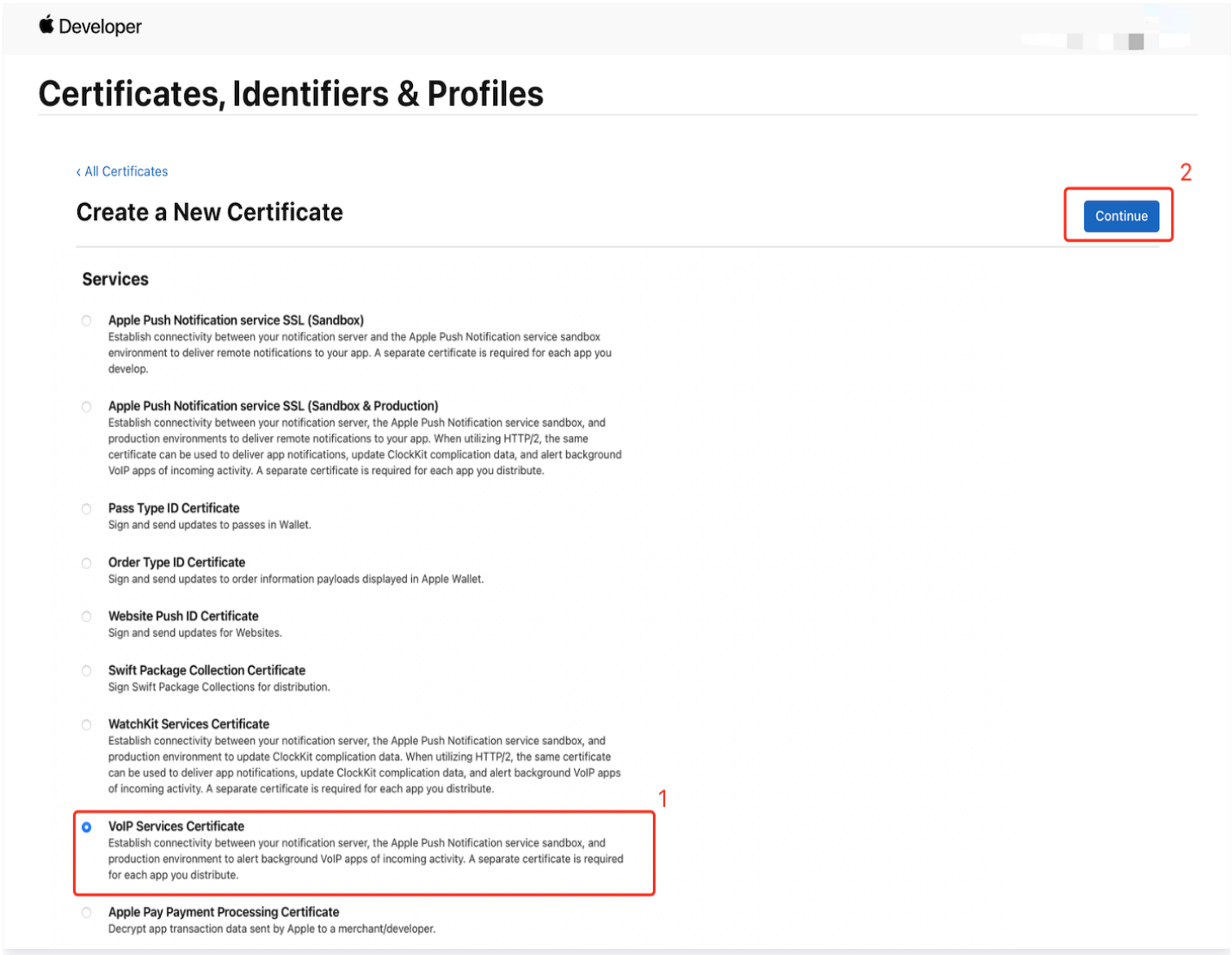
在申请 VoIP Push 证书之前，请先登录 苹果开发者中心 网站，开启 App 的远程推送功能。当您的 AppID 具备了 Push Notification 能力后，按照如下步骤申请并配置 VoIP Push 证书：
1. 登录 苹果开发者中心 网站，单击 Certificates,Identifiers & Profiles 或者侧栏的 Certificates, IDS & Profiles，进入 Certificates, IDS & Profiles 页面。



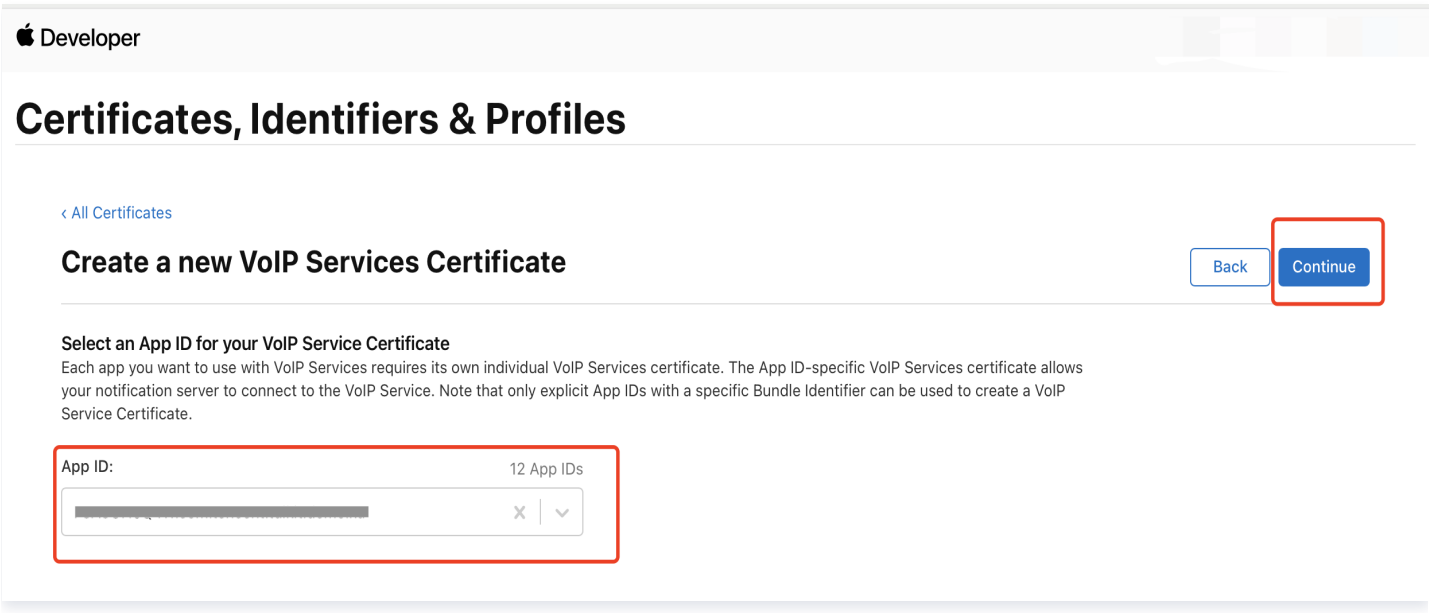
2. 单击 Certificates 右侧的 +。



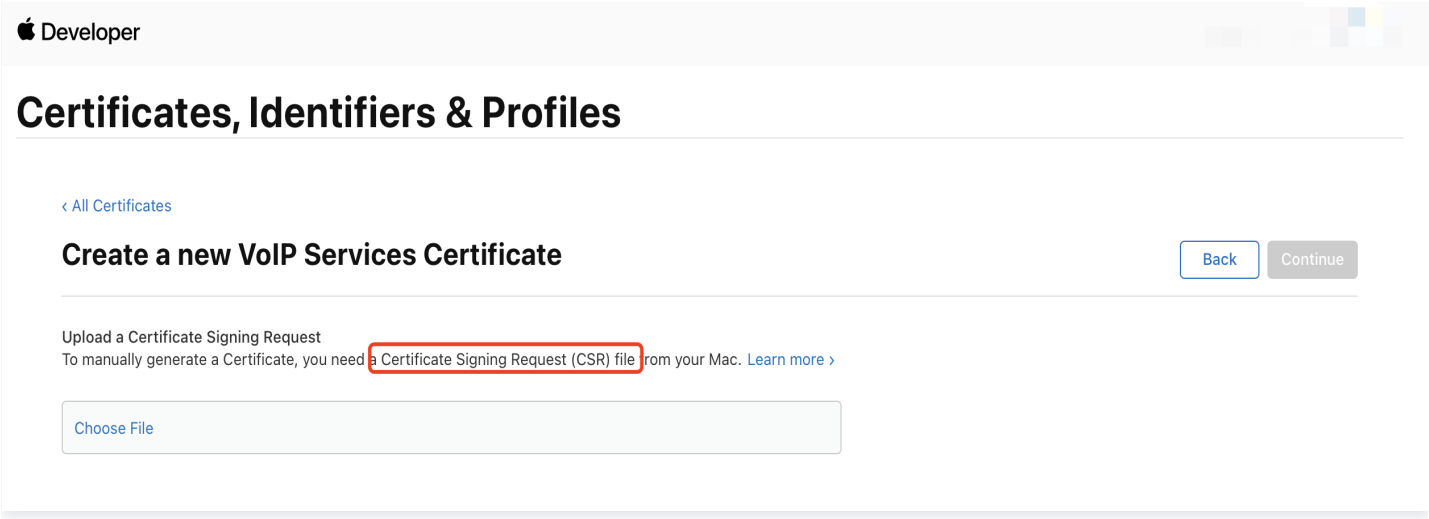
3. 在 Create a New Ceritificate 选项卡中，选择 VoIP Services Certificate，并点击 Continue。



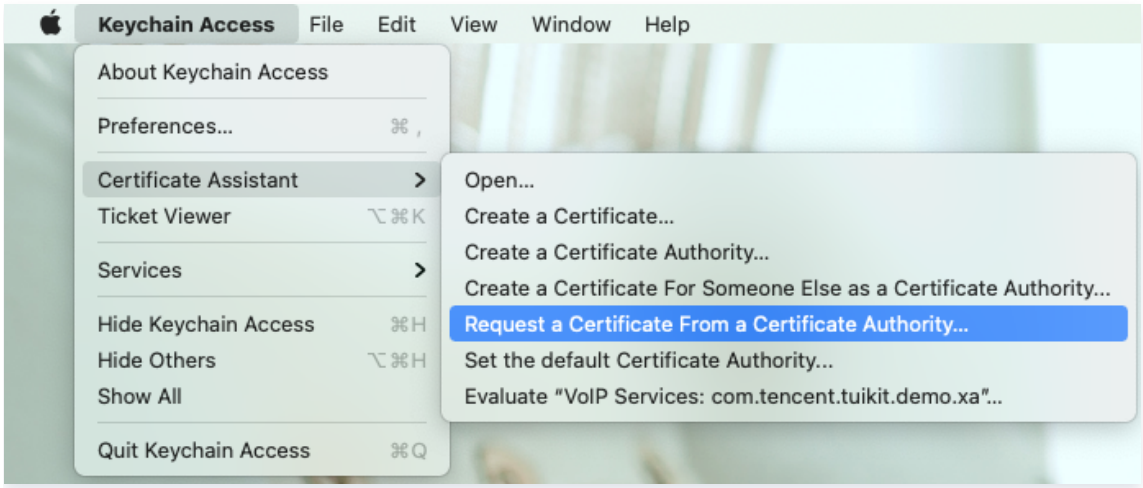
4. 在 Select an App ID for your VoIP Service Certificate 选项卡中，选择您当前的 App 的 BundleID，并点击 Continue。



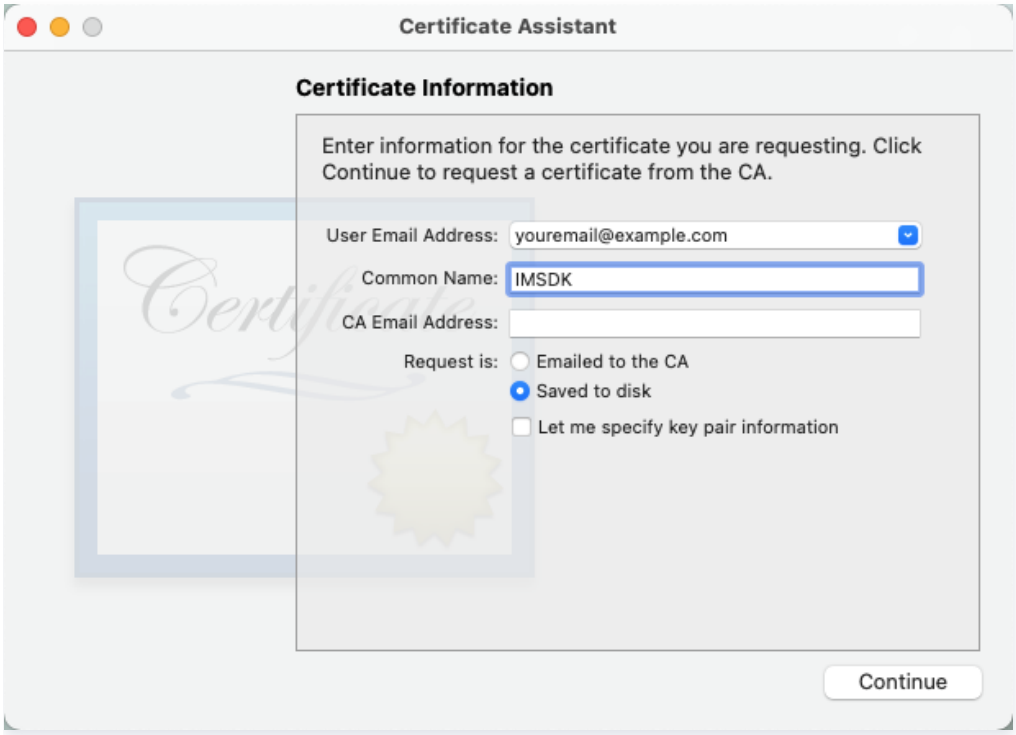
5. 紧接着，系统提示我们需要一个 Certificate Signing Request（CSR）。



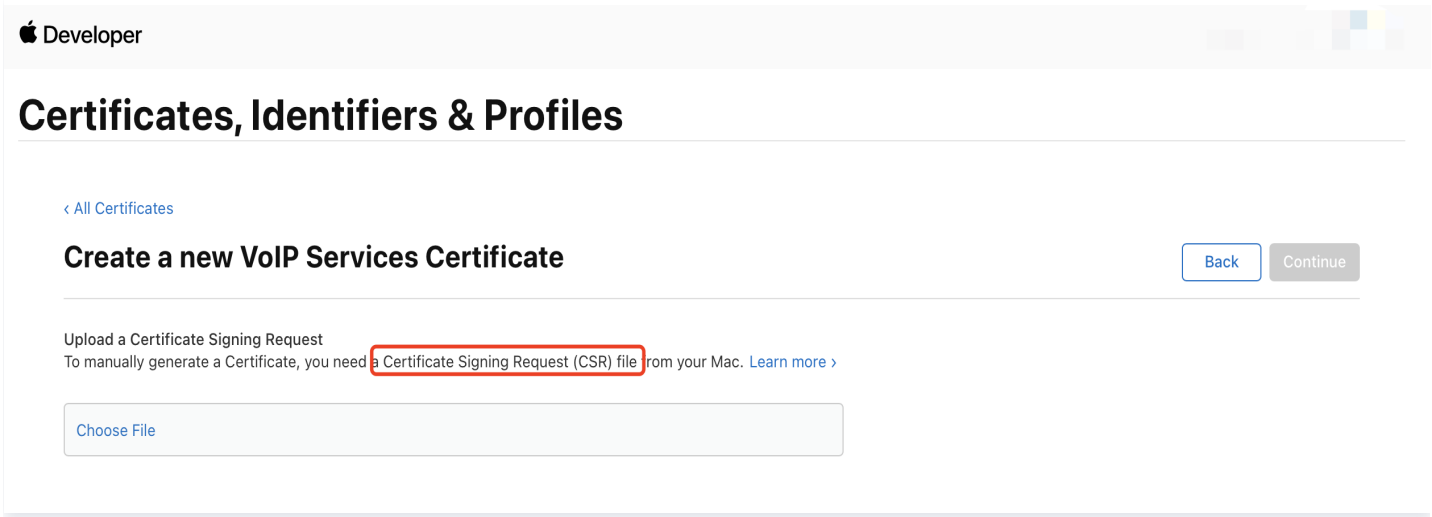
6. 我们接下来制作 CSR 文件。首先在 Mac 上打开 **钥匙串访问工具（Keychain Access）**，在菜单中选择 **钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（ `Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority` ）。



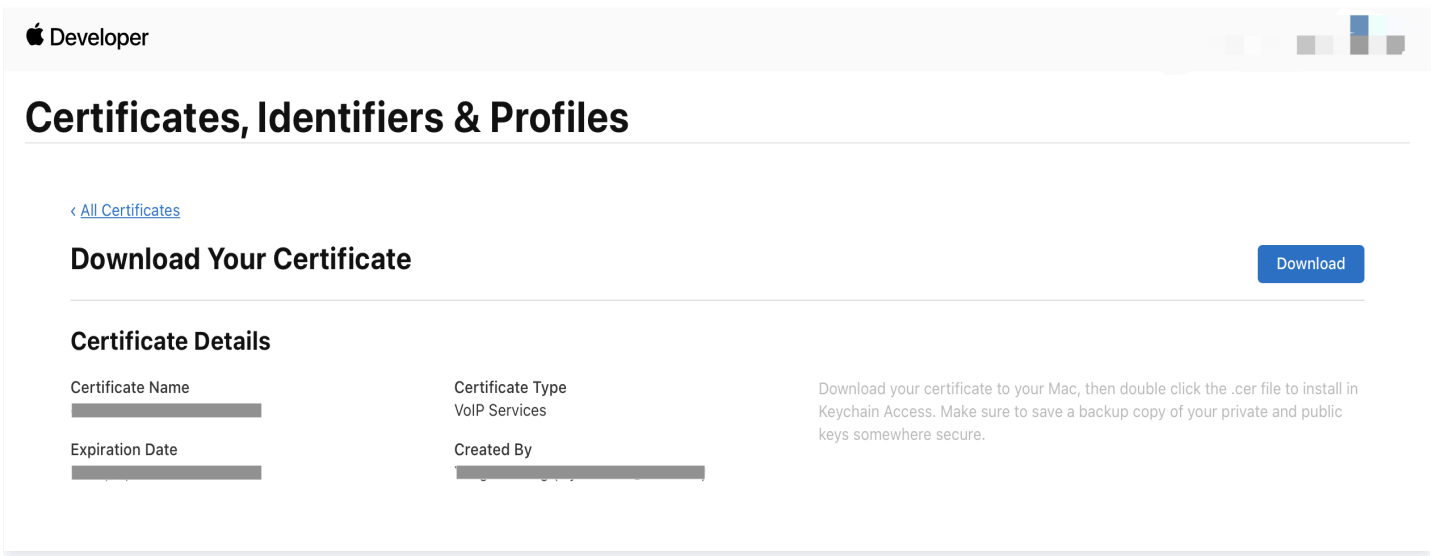
7. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择 **存储到磁盘**，单击继续，系统将生成一个 `*.certSigningRequest` 文件。



返回上述 **步骤5** 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。

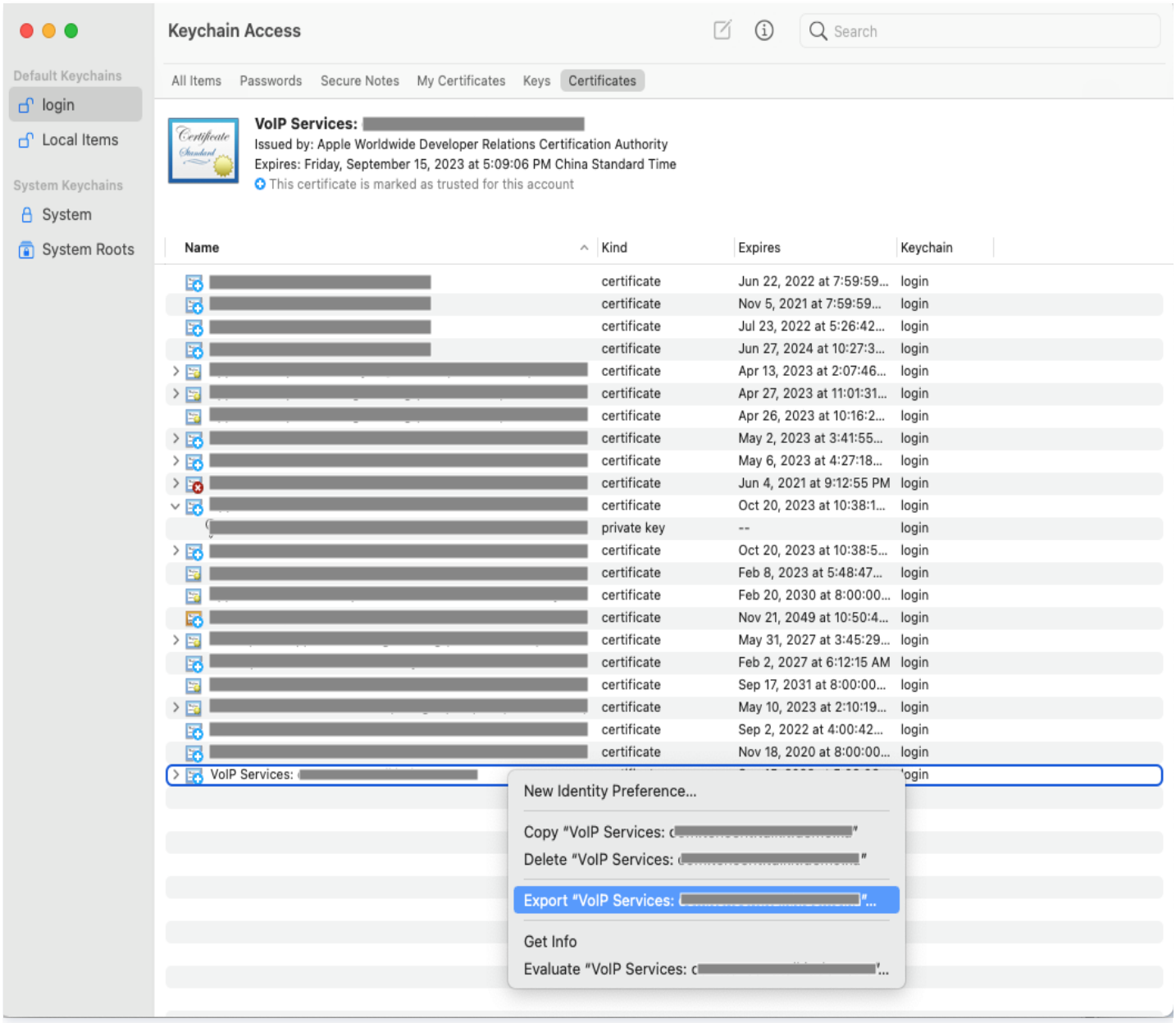


8. 单击 Continue 后生成证书，点击 Download 下载对应的证书到本地。



9. 双击打开刚才下载的 `voip_services.cer` ，系统会将其导入钥匙串中。
10. 打开钥匙串应用，在 **登录** > **我的证书**，右键导出刚创建的 VoIP Services 的 `P12` 文件。

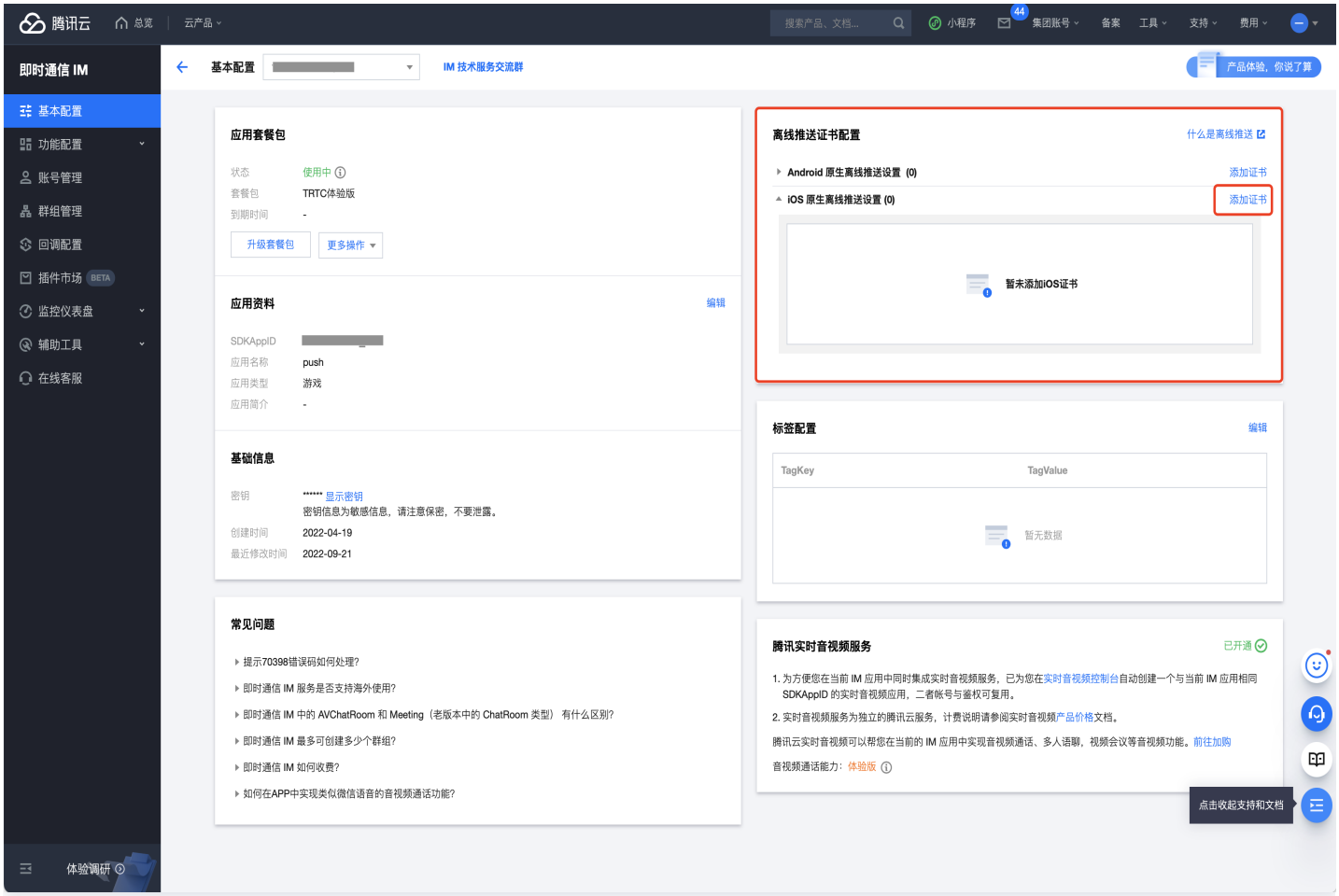
> **说明：**
> 保存 `P12` 文件时，请务必要为其设置密码。



步骤2：上传证书到 IM 控制台

打开 IM 控制台，选择您创建的 IM 应用，并按照如下步骤上传证书：

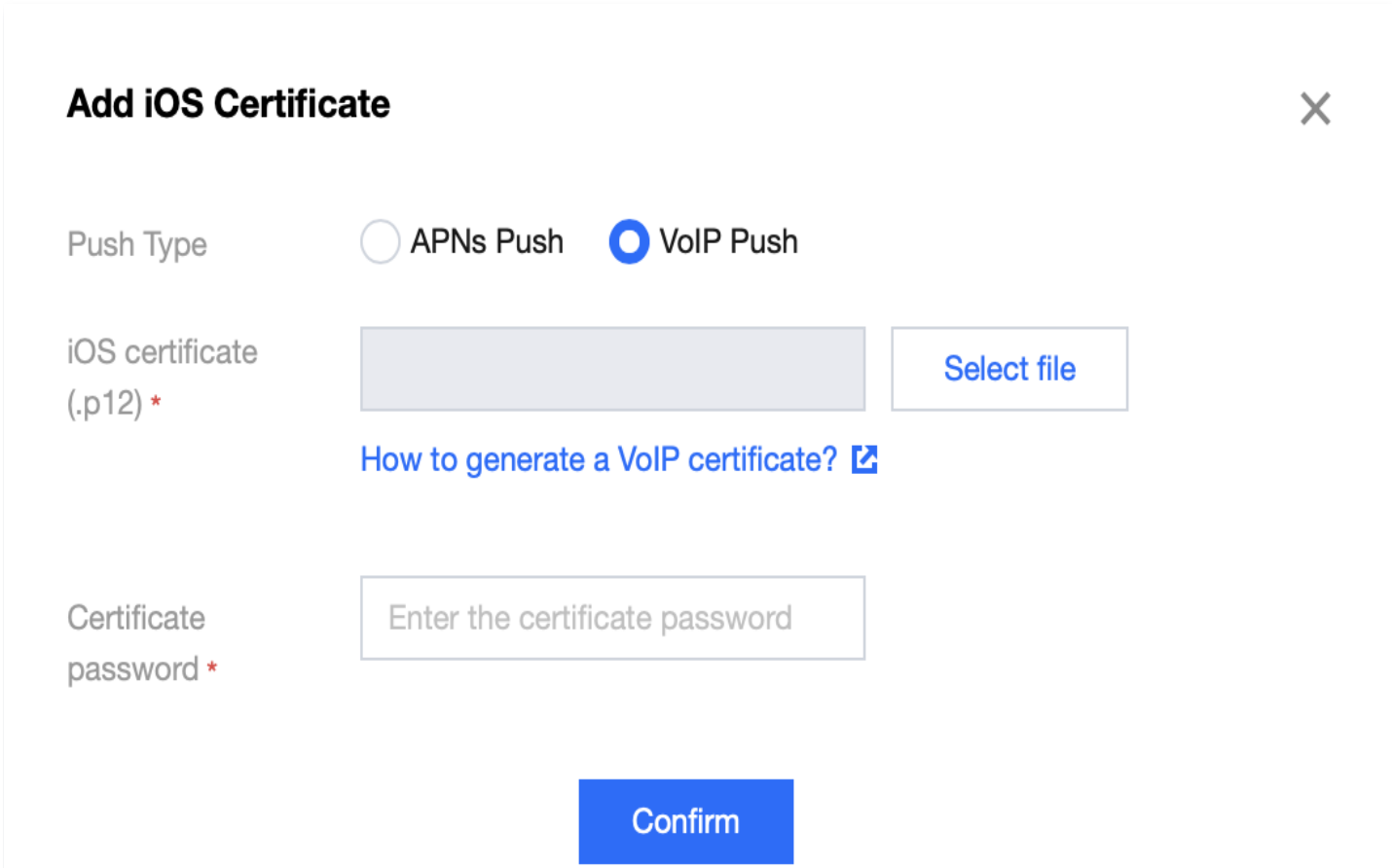
1. 选择您的 IM 应用，在**离线推送证书配置**选项卡中，点击**添加证书**。



2. 在弹出的**添加 iOS 证书**选项卡中，分别上传生产环境和开发环境的 VoIP 证书。

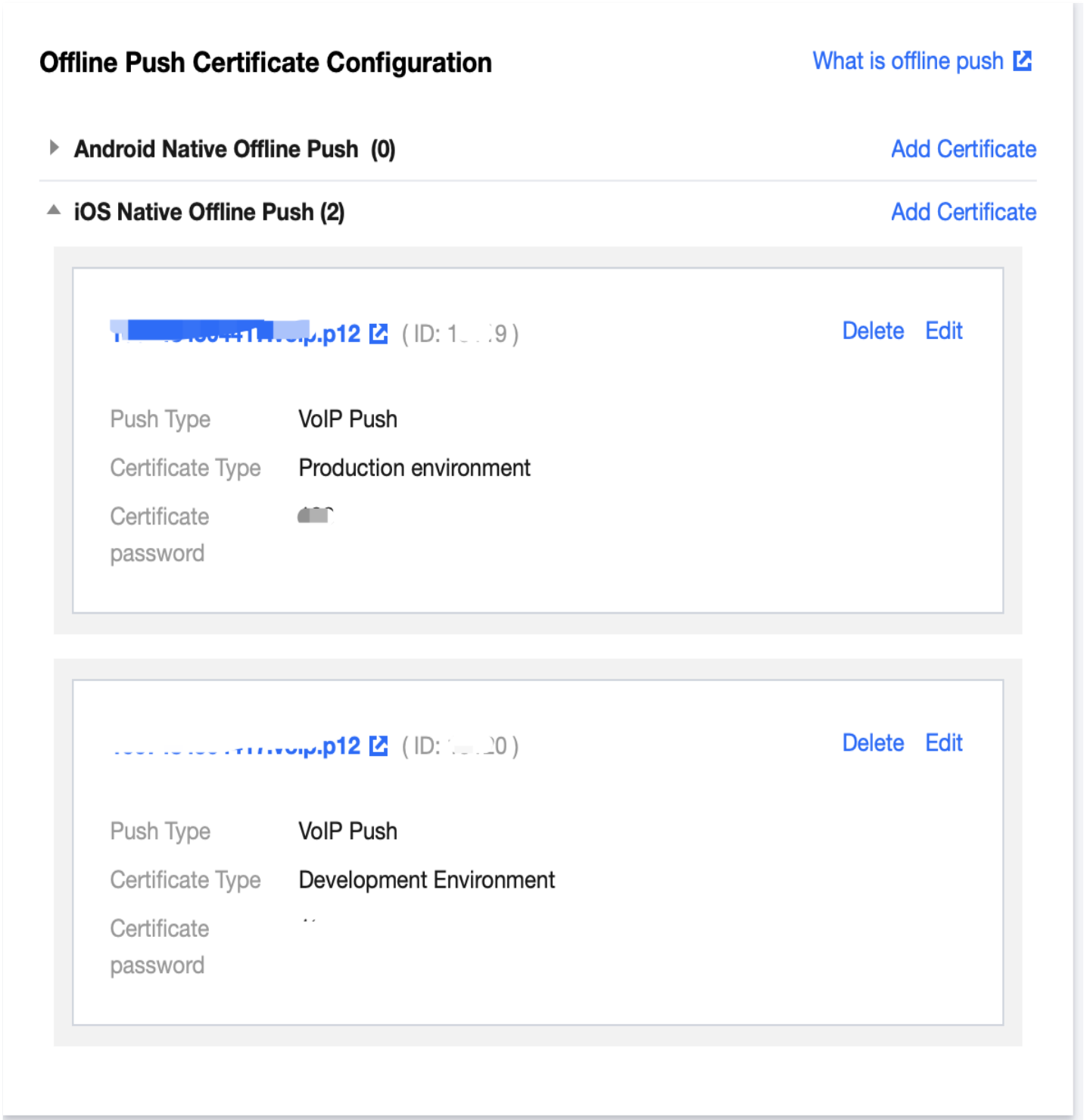
说明：

- VoIP Push 证书本身不区分生产环境和测试环境，生产环境和开发环境使用的是同一份 VoIP Push 证书，请分别上传。
- 上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。
- 上传证书需要设置密码，无密码收不到推送。
- 发布 App Store 的证书需要设置为生产环境，否则无法收到推送。
- 上传的 p12 证书必须是自己申请的真实有效的证书。



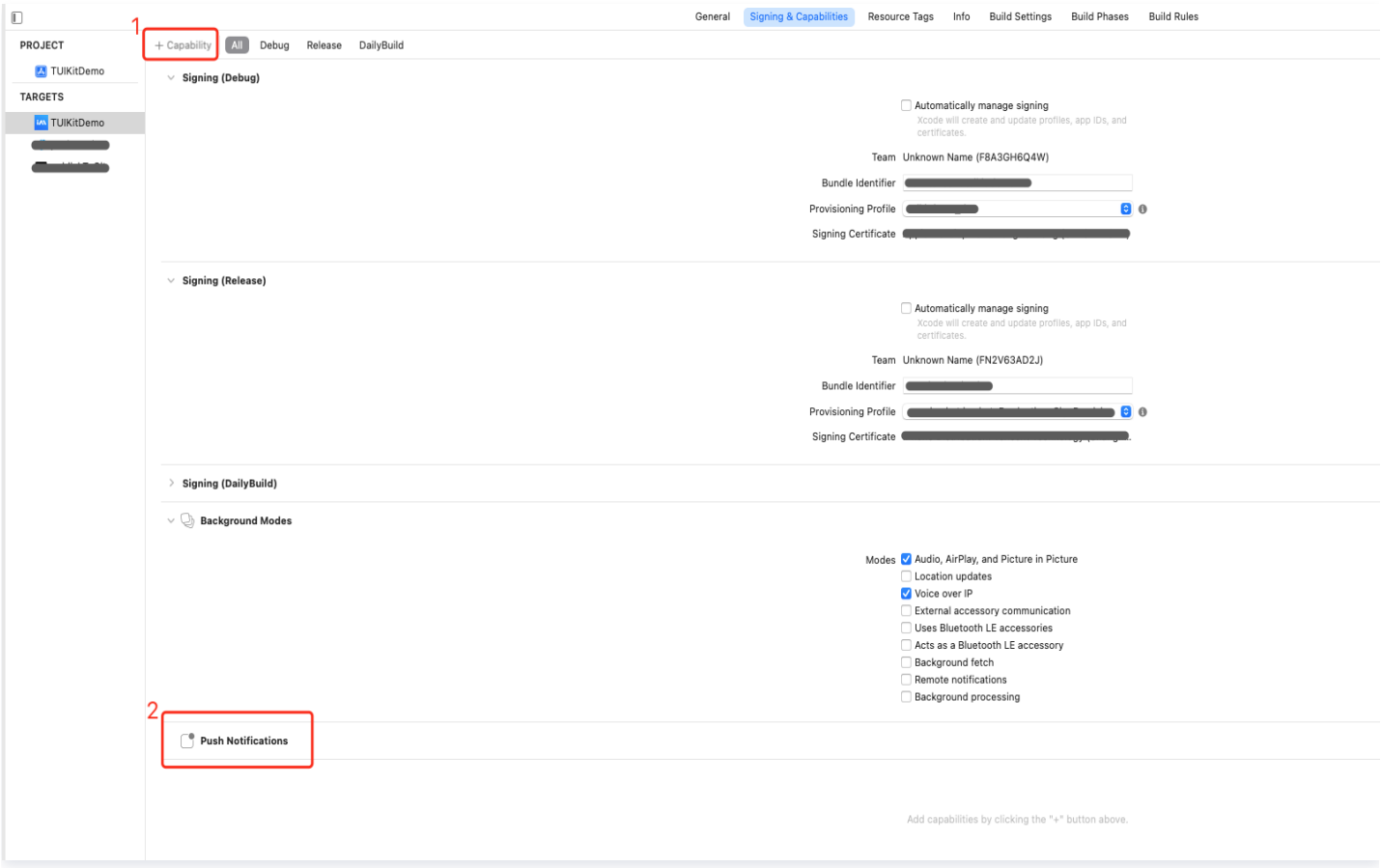
3. 上传完成后，记录不同环境下的证书 ID。

> **说明：**
> 开发环境和生产环境下的证书 ID 要严格区分，并在 步骤4：接入TUICallKitVoIPExtension 组件 时根据实际环境填写。

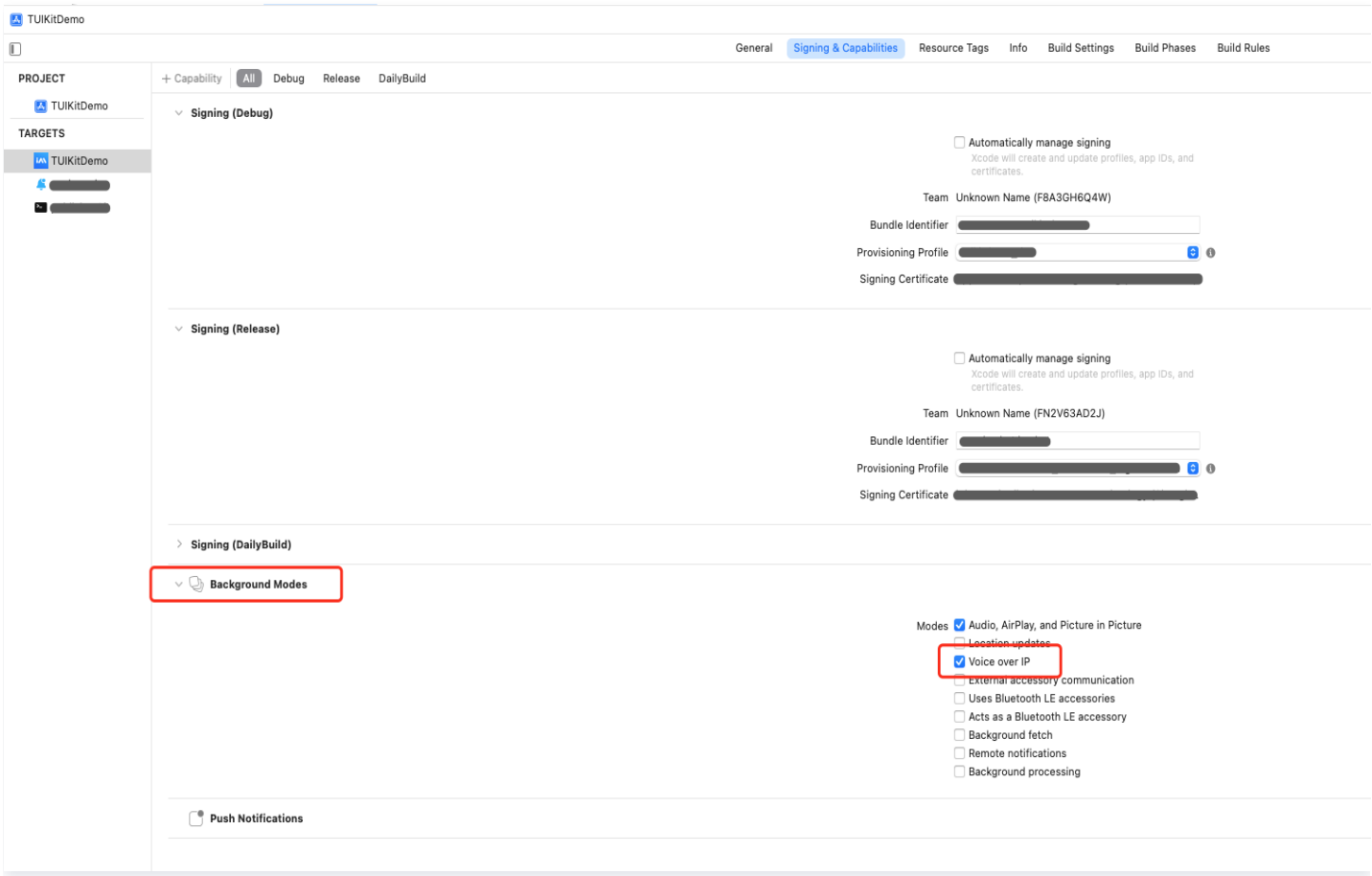


步骤3：完成工程配置

1. 如下图，确认您工程的 Capability 中是否添加 Push Notification 能力。



2. 如下图，请检查您工程 Capability 的 Backgrounds Modes 中，是否开启了 Voice over IP 选项。



步骤4：接入 TUICallKitVoIPExtension 组件

使用 CocoaPods 导入组件，具体步骤如下：

1. 在您的 Podfile 文件中添加以下依赖。

```
pod 'TUICallKitVoIPExtension'

//如果在Flutter平台集成VoIP，请使用：
pod 'TUICallKitVoIPExtension/Professional'
```

2. 执行以下命令，安装组件。

```
pod install
```

3. 上报 步骤 2：记录的证书 ID 。

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        // 上报证书 ID
        TUICallKitVoIPExtension.setCertificateID(CertificateID)
        return true
```

```
    }
```

说明：
如果无法安装 TUICallKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

拨打 VoIP 通话

如果您需要拨打 VoIP 通话，需要在调用 call 时候设置 OfflinePushInfo 中的 iOSPushType 字段为 TUICallIOSOfflinePushTypeVoIP，默认为 TUICallIOSOfflinePushTypeAPNs。

Objective-C

```
    let params = TUICallParams()
    let pushInfo: TUIOfflinePushInfo = TUIOfflinePushInfo()
    pushInfo.title = "TUICallKit"
    pushInfo.desc = "TUICallKit.have.new.invitation"
    // iOS push type: if you want user VoIP, please modify type to TUICallIOSOfflinePushTypeVoIP
    pushInfo.iOSPushType = .apns
    pushInfo.ignoreIOSBadge = false
    pushInfo.iOSSound = "phone_ringing.mp3"
    pushInfo.androidSound = "phone_ringing"
    // OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
    // OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
    pushInfo.androidOPPOChannelID = "tuikit"
    // FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
    pushInfo.androidFCMChannelID = "fcm_push_channel"
    // VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
    pushInfo.androidVIVOClassification = 1
    // HuaWei message type: https://developer.huawei.com/consumer/cn/doc/development/HMSCore-
Guides/message-classification-0000001149358835
    pushInfo.androidHuaWeiCategory = "IM"
    params.userData = "User Data"
    params.timeout = 30
    params.offlinePushInfo = pushInfo
    TUICallKit.createInstance().call(userId: "123456", callMediaType: .audio, params: params) {  } fail: { code,
message in }
```

Java

```
TUICallDefine.OfflinePushInfo offlinePushInfo = new TUICallDefine.OfflinePushInfo();
offlinePushInfo.setTitle("");
offlinePushInfo.setDesc("You have receive a new call");
// OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
// OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
```

```
offlinePushInfo.setAndroidOPPOChannelID("tuikit");
offlinePushInfo.setIgnoreIOSBadge(false);
offlinePushInfo.setIOSSound("phone_ringing.mp3");
offlinePushInfo.setAndroidSound("phone_ringing"); //Note：don't add suffix
//VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
offlinePushInfo.setAndroidVIVOClassification(1);
//FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
offlinePushInfo.setAndroidFCMChannelID("fcm_push_channel");
//Huawei message type
offlinePushInfo.setAndroidHuaWeiCategory("IM");
//IOS push type: if you want user VoIP, please modify type to TUICallDefine.IOSOfflinePushType.VoIP
offlinePushInfo.setIOSPushType(TUICallDefine.IOSOfflinePushType.VoIP);

TUICallDefine.CallParams params = new TUICallDefine.CallParams();
params.offlinePushInfo = offlinePushInfo;

TUICallKit.createInstance(context).call("mike", TUICallDefine.MediaType.Video, params, null);
```

Dart

```
TUIOfflinePushInfo offlinePushInfo = TUIOfflinePushInfo();
 offlinePushInfo.title = "Flutter TUICallKit";
 offlinePushInfo.desc = "This is an incoming call from Flutter TUICallkit";
 offlinePushInfo.ignoreIOSBadge = false;
 offlinePushInfo.iOSSound = "phone_ringing.mp3";
 offlinePushInfo.androidSound = "phone_ringing";
 offlinePushInfo.androidOPPOChannelID = "Flutter TUICallKit";
 offlinePushInfo.androidVIVOClassification = 1;
 offlinePushInfo.androidFCMChannelID = "fcm_push_channel";
 offlinePushInfo.androidHuaWeiCategory = "Flutter TUICallKit";
 offlinePushInfo.iOSPushType = TUICallIOSOfflinePushType.VoIP;

 TUICallParams params = TUICallParams(offlinePushInfo: offlinePushInfo);

 TUICallKit.instance.call(callUserId, TUICallMediaType.audio, params);
```

Swift

```
[[TUICallKit createInstance] call:@"mike 的 id" params:[self getCallParams]
callMediaType:TUICallMediaTypeVideo];

- (TUICallParams *)getCallParams {
    TUIOfflinePushInfo *offlinePushInfo = [self createOfflinePushInfo];
    TUICallParams *callParams = [TUICallParams new];
    callParams.offlinePushInfo = offlinePushInfo;
    callParams.timeout = 30;
    return callParams;
}
```

```
+ (TUIOfflinePushInfo *)createOfflinePushInfo {
    TUIOfflinePushInfo *pushInfo = [TUIOfflinePushInfo new];
    pushInfo.title = @"";
    pushInfo.desc = TUICallingLocalize(@"TUICallKit.have.new.invitation");
    // iOS push type: if you want user VoIP, please modify type to TUICallIOSOfflinePushTypeVoIP
    pushInfo.iOSPushType = TUICallIOSOfflinePushTypeAPNs;
    pushInfo.ignoreIOSBadge = NO;
    pushInfo.iOSSound = @"phone_ringing.mp3";
    pushInfo.AndroidSound = @"phone_ringing";
    // OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
    // OPPO must set a ChannelID to receive push messages. This channelID needs to be the same as the
console.
    pushInfo.AndroidOPPOChannelID = @"tuikit";
    // FCM channel ID, you need change PrivateConstants.java and set "fcmPushChannelId"
    pushInfo.AndroidFCMChannelID = @"fcm_push_channel";
    // VIVO message type: 0-push message, 1-System message(have a higher delivery rate)
    pushInfo.AndroidVIVOClassification = 1;
    // HuaWei message type: https://developer.huawei.com/consumer/cn/doc/development/HMSCore-
Guides/message-classification-0000001149358835
    pushInfo.AndroidHuaWeiCategory = @"IM";
    return pushInfo;
}
```

通过系统通话记录拨打通话

如果需要实现点击**系统电话 - 最近通话**列表中的通话记录，直接发起单人音视频通话，您需要在 Application 的生命周期回调函数使用 **TUICallKitVoIPExtension** 组件中的 **callWith** 接口，具体示例如下：

> **说明：**
> - 仅支持单人音视频通话的直接拨打。
> - 登录账户必须为同一个账户。

1. 在iOS 13（及以后版本）上，使用了 **SceneDelegate**，且最低兼容版本到iOS 13以下，需分别在 **AppDelegate** 和 **SceneDelegate**中实现如下方法。

```
/// AppDelegate 中的实现，iOS 13 以前版本的处理
- (BOOL)application:(UIApplication *)application continueUserActivity:(nonnull NSUserActivity *)userActivity
restorationHandler:(nonnull void (^)(NSArray<id<UIUserActivityRestoring>> * _Nullable))restorationHandler
{
    [TUICallKitVoIPExtension callWith:userActivity];
    return YES;
}

/// SceneDelegate 中的实现
- (void)scene:(UIScene *)scene willConnectToSession:(UISceneSession *)session options:
(UISceneConnectionOptions *)connectionOptions {
    [connectionOptions.userActivities enumerateObjectsUsingBlock:^(NSUserActivity * _Nonnull userActivity,
BOOL * _Nonnull stop) {
        [TUICallKitVoIPExtension callWith:userActivity];
    }];
}
```

```
- (void)scene:(UIScene *)scene continueUserActivity:(NSUserActivity *)userActivity {
    [TUICallKitVoIPExtension callWith:userActivity];
}
```

2. 在iOS 13（及以后版本）上，未使用**SceneDelegate**，只需要在 **AppDelegate** 实现如下方法。

```
- (BOOL)application:(UIApplication *)application continueUserActivity:(nonnull NSUserActivity *)userActivity
restorationHandler:(nonnull void (^)(NSArray<id<UIUserActivityRestoring>> * _Nullable))restorationHandler
{
    [TUICallKitVoIPExtension callWith:userActivity];
    return YES;
}
```

常见问题

获取不到 VoIP Push

1. 首先检查下 App 的运行环境和证书的环境是否一致，证书 ID 是否匹配，如果不一致，无法收到推送。
2. 请确认当前您登录的账号是否处于离线状态：按 home 键切后台、登录后主动杀进程退出。VoIP Push 目前只支持离线状态下的推送。
3. 检查 步骤3：完成工程配置 是否正确。
4. 尝试重启测试手机来清除系统缓存和内存。

uni-app

最近更新时间：2023-09-27 16:26:02

离线唤醒功能，能够让您的应用在**后台运行**或者**离线状态**下依然能够收到音视频通话的响铃呼叫。TUICallKit 插件需要集成 TUIOfflinePush 组件实现离线唤醒功能，本文将介绍如何在音视频通话项目中集成 TUIOfflinePush 组件。

前期准备

注册应用到厂商推送平台

离线推送功能依赖厂商原始通道，您需要将自己的应用注册到各个厂商的推送平台，得到 APPID 和 APPKEY 等参数；并将得到的参数配置到 IM 控制台，具体请参见 离线推送（uniapp） 中的 Android 配置 和 iOS 配置。

集成 TUIOfflinePush 插件

1、音视频通话需集成 TUIOfflinePush 插件，具体集成步骤请参见 离线推送（uniapp）中的 集成 TUIOfflinePush 插件，完成步骤1、步骤2、步骤3、步骤6；
2、剩下的步骤4和步骤5不需要处理，TUICallKit 内部已封装了对应的消息格式。

> **注意：**
> TUICallKit 内部封装了消息格式，消息格式中没有携带各个厂商的 channelID。因此，在配置各个厂商的参数到 IM 控制台的时候必须在 IM 控制台填写厂商的 channelID。
> TUICallKit 内部封装了 VIVO 的消息类别为：系统消息类，具体可查看 厂商推送限制 中的 vivo 厂商描述。
> TUICallKit 内部封装了 华为的消息类别为：IM，具体可查看 厂商推送限制 中的 华为厂商描述。
> TUICallKit 内部封装了 苹果手机的推送方式为：APNs，暂不支持 VoIP。如有需求，可 进入 IM 社群 进行咨询和反馈。

以 OPPO 厂商为例，控制台配置界面如下所示：



完成上述步骤，您可以在音视频通话插件 TUICallKit 中使用离线唤醒功能。

条件说明

部分厂商要求必须上架应用市场才可以正常使用推送服务，详情参见下表：

厂商通道	是否需要上架	账号说明
小米	是	需要注册企业开发者账号
华为	否	个人开发者账号即可
魅族	否	个人开发者账号即可
荣耀	否	需要注册企业开发者账号
VIVO	是	需要注册企业开发者账号
OPPO	否	需要注册企业开发者账号

集成效果

完成上述步骤后，应用在后台或者离线的时候，可以收到推送通知消息，通知样式如下所示：
以下是谷歌手机效果，不同厂商效果可能不同，请以真机运行。



常见问题

1、收不到离线推送通知?

请检查控制台中各厂商的参数以及 channelID 是否配置，如果已配置，请参见 离线推送(uniapp) -常见问题 进行排查。

2、如何设置自定义铃声?

TUIOfflinePush 插件暂不支持自定义铃声。

技术咨询

如有其他问题，欢迎您 进入 IM 社群 进行咨询和反馈。

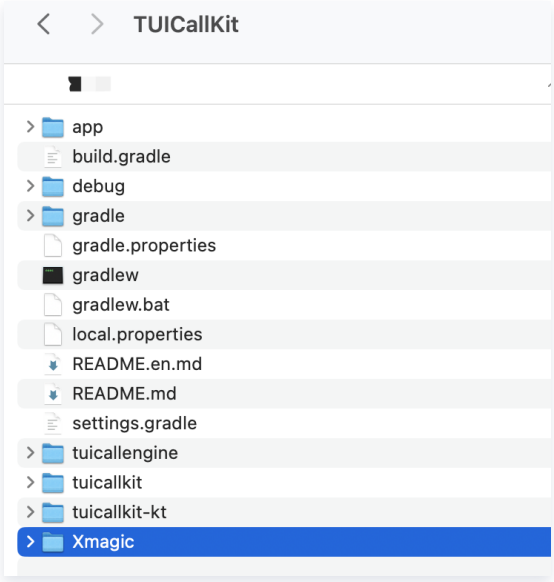
美颜特效（TUICallKit）
Android

最近更新时间：2023-07-13 15:41:42

TUICallEngine 中提供了 getTRTCCloudInstance() 接口，可以通过该接口调用 TRTC 的高级特性 setLocalVideoProcessListener()，实现接入第三方美颜。
本文将介绍如何在 TUICallKit 中接入 腾讯特效 SDK，**其他第三方美颜接入方法类似，请结合第三方 SDK 文档进行接入。**

接入准备

按照 腾讯特效 TRTC 接入指引，下载**场景化 Demo 的 zip 包**，按照文档替换资源，将 Xmagic 目录复制到自己的工程中，完成 步骤一；
导入后的目录结构如下：



导入 Xmagic

1. 在工程根目录下找到 settings.gradle 文件，在其中增加如下代码，将 Xmagic 组件导入到您当前的项目中。

```
include ':Xmagic'
```

2. 在 tuicallkit 目录下找到 build.gradle 文件，并在其中增加如下代码，声明 tuicallkit 对新加入的 Xmagic 组件的依赖。

```
api project(':Xmagic')
```

3. 修改 Xmagic 目录下 build.gradle 中的 compileSdkVersion、minSdkVersion、TRTC SDK 版本、Glide 版本等与您的工程保持一致；依赖的 Xmagic SDK 版本为您的套餐包版本。

```
api 'com.tencent.liteav:LiteAVSDK_TRTC:11.1.0.13111'
api 'com.github.bumptech.glide:glide:4.12.0'

api 'com.tencent.mediacloud:TencentEffect_S1-04:3.0.0.13'
```

4. 修改 XMagicImpl.java 文件，替换自己的 authKey 和 authLicenceUrl。

```
public static final String authKey = YOUR_LICENSE_KEY;
public static final String authLicenceUrl = YOUR_LICENSE_URL;
```

初始化 Xmagic

Xmagic 提供了美颜面板供使用者调用，TUICallKit 集成的时候，可以在 BaseCallActivity 文件中增加布局，每次拉起通话界面时展示，也可以在显示出通话界面后以 Dialog 的形式展示，注意生命周期即可。可参考 腾讯特效 TRTC 接入指引 中 Demo 程的 `ThirdBeautyActivity` 类。

1. 授权：

```
//鉴权注意事项及错误码详情，请参考
https://cloud.tencent.com/document/product/616/65891#.E6.AD.A5.E9.AA.A4.E4.B8.80.EF.BC.9A.E9.89.B4
.E6.9D.83
XMagicImpl.init(getApplicationContext());

if
(TUICallDefine.MediaType.Video.equals(TUICallingStatusManager.sharedInstance(getApplicationContext())
.getMediaType())) {
    if (XmagicLoadAssetsView.isCopyedRes) {
        initXMagic();
        return;
    }

    XMagicImpl.checkAuth((errorCode, msg) -> {
        if (errorCode == TELicenseCheck.ERROR_OK) {
            showLoadResourceView();
        } else {
            Log.e("Xmagic", "please check url and key" + errorCode + " , " + msg);
        }
    });
 }
```

2. 初始化素材：

```
private void showLoadResourceView() {
    if (XmagicLoadAssetsView.isCopyedRes) {
        XmagicResParser.parseRes(getApplicationContext());
        initXMagic();
    } else {
        loadAssetsView = new XmagicLoadAssetsView(this);
        loadAssetsView.setOnAssetsLoadFinishListener(() -> {
            XmagicResParser.parseRes(getApplicationContext());
            initXMagic();
        });
    }
}
```

设置纹理回调和添加美颜效果

```
//1. set TRTCVideoFrameListener, refer to：{https://liteav.sdk.qcloud.com/doc/api/zh-
cn/group__TRTCCloud__android.html#a0b565dc8c77df7fb826f0c45d8ad2d85}
TRTCCloud trtcCloud = TUICallEngine.createInstance(this).getTRTCCloudInstance();
```

```
trtcCloud.setLocalVideoProcessListener(TRTCCloudDef.TRTC_VIDEO_PIXEL_FORMAT_Texture_2D,
        TRTCCloudDef.TRTC_VIDEO_BUFFER_TYPE_TEXTURE, new
TRTCCloudListener.TRTCVideoFrameListener() {
            @Override
            public void onGLContextCreated() { //2. GLContext created
              if (isGLContextCreated) {
                 isGLContextCreated = false;
              } else {
                 runOnUiThread(() -> initXMagic());
              }
            }
            @Override
            public int onProcessVideoFrame(TRTCCloudDef.TRTCVideoFrame srcFrame,
TRTCCloudDef.TRTCVideoFrame dstFrame) {
            //3. invoke xmagic module, more details refer to {https://liteav.sdk.qcloud.com/doc/api/zh-
cn/group__TRTCCloudListener__android.html#a22afb08b2a1a18563c7be28c904b166a}
               if (mXMagic != null) {
                  dstFrame.texture.textureId = mXMagic.process(srcFrame.texture.textureId, srcFrame.width,
srcFrame.height);
               }
               return dstFrame.texture.textureId;
            }
            @Override
            public void onGLContextDestory() {
            //4. GLContext destroyed
               if (mXMagic != null) {
                 mXMagic.onDestroy();
               }
            }
});
```

1. **布局中添加 SDK 美颜面板：**

```
<RelativeLayout
    android:layout_above="@+id/ll_edit_info"
    android:id="@+id/livepusher_bp_beauty_pannel"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

2. **初始化面板：**

```
  mBeautyPanelView = findViewById(R.id.livepusher_bp_beauty_pannel);

  private void initXMagic() {
     if (mXMagic == null) {
        mXMagic = new XMagicImpl(this, mBeautyPanelView);
     }else {
        mXMagic.onResume();
     }
  }
```

完成上述步骤后，即可在音视频通话中使用第三方美颜。

集成效果

下图是 TUICallKit 含 UI 组件集成腾讯特效 SDK 面板的效果：



常见问题

集成 Xmagic 后，动效、美妆、分割下内容怎么为空?

请先检查套餐是否包含该特效；如果套餐包满足，请检查 /asset/下的资源文件是否包含 MotionRes/2dMotionRes 等目录及文件。

iOS

最近更新时间：2023-07-21 15:51:51

TUICallEngine 中提供了 getTRTCCloudInstance() 接口，可以通过该接口调用 TRTC 的高级特性 setLocalVideoProcessDelegete:pixelFormat:bufferType: 实现接入第三方美颜。
本文将介绍如何在 TUICallKit 中接入 腾讯特效 SDK，**其他第三方美颜接入方法类似，请结合第三方 SDK 文档进行接入。**

导入组件

您可以使用 CocoaPods 导入组件，具体如下：
1. 在您的 `Podfile` 文件中添加以下依赖。

```
Objective-C
pod 'XMagic'
```

2. 执行以下命令，安装组件。

```
pod install
```

如果无法安装 XMagic 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

3. 添加美颜资源到实际项目工程中：
   3.1 下载并解压对应套餐的 SDK 和美颜资源，将 **resources 文件夹下的除 LightCore.bundle、Light3DPlugin.bundle、LightBodyPlugin.bundle、LightHandPlugin.bundle、LightSegmentPlugin.bundle、audio2exp.bundle 以外的其它 bundle 资源**添加到实际工程中。
   3.2 在 Build Settings 中的 Other Linker Flags 添加 `-ObjC` 。
4. 将 `Bundle Identifier` 修改成与申请的测试授权一致。

授权

1. 申请授权，得到 LicenseURL 和 LicenseKEY，请参见 License 指引 。

> ⚠ **注意：**
> 正常情况下，只要 App 成功联网一次，就能完成鉴权流程，因此您**不需要**把 License 文件放到工程的工程目录里。但是如果您的 App 在从未联网的情况下也需要使用 SDK 相关功能，那么您可以把 License 文件下载下来放到工程目录，作为保底方案，此时 License 文件名必须是 `v_cube.license` 。

2. 在相关业务模块的初始化代码中设置 URL 和 KEY，触发 license 下载，避免在使用前才临时去下载。也可以在 AppDelegate 的 didFinishLaunchingWithOptions 方法里触发下载。其中，LicenseURL 和 LicenseKey 是控制台绑定 License 时生成的授权信息。

```
[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authresult, NSString *
_Nonnull errorMsg) {
    if (authresult == TELicenseCheckOk) {
        NSLog(@"鉴权成功");
```

```
    } else {
        NSLog(@"鉴权失败");
    }
}];
```

设置 SDK 素材资源路径

```
#import <XMagic/XMagic.h>

- (void)buildBeautySDK:(int)width and:(int)height texture:(unsigned)textureID {
    NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) lastObject];
    beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config.json"];
    NSFileManager *localFileManager=[[NSFileManager alloc] init];
    BOOL isDir = YES;
    NSDictionary * beautyConfigJson = @{};

    if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isDir) {
        NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfigPath
encoding:NSUTF8StringEncoding error:nil];
        NSError *jsonError;
        NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncoding];
        beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                                           options:NSJSONReadingMutableContainers
                                                           error:&jsonError];
    }

    NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                                 @"root_path":[[NSBundle mainBundle] bundlePath],
                                 @"tnn_"
                                 @"beauty_config":beautyConfigJson
    };

    // 初始化SDK：width 和 height 分别是 texture 的宽高
    self.xMagicKit = [[XMagic alloc] initWithRenderSize:CGSizeMake(width, height) assetsDict:assetsDict];
}
```

配置美颜各种效果（详细美颜效果配置请参考 美颜参数说明 ）

```
- (int)configPropertyWithType:(NSString *_Nonnull)propertyType withName:(NSString
*_Nonnull)propertyName withData:(NSString*_Nonnull)propertyValue withExtraInfo:(id _Nullable)extraInfo;
```

进行渲染处理

TRTC SDK 设置第三方美颜的视频数据回调：设置该回调之后，TRTC 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。

```
#import <TUICallEngine/TUICallEngine.h>

[[[TUICallEngine createInstance] getTRTCCloudInstance] setLocalVideoProcessDelegete:self
pixelFormat:TRTCVideoPixelFormat_Texture_2D bufferType:TRTCVideoBufferType_Texture];
```

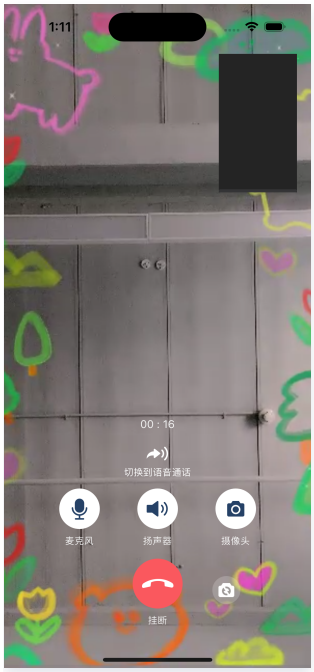
版权所有：腾讯云计算（北京）有限责任公司

在视频帧回调接口 onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVideoFrame *_Nonnull)dstFrame
中，构造 YTProcessInput 传入到 SDK 内做渲染处理。

```
#pragma mark - TRTCVideoFrameDelegate

- (uint32_t)onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVideoFrame *_Nonnull)dstFrame {
    if (!self.xMagicKit) {
        [self buildBeautySDK:srcFrame.width and:srcFrame.height texture:srcFrame.textureId];
        self.heightF = srcFrame.height;
        self.widthF = srcFrame.width;
    }

    if(self.xMagicKit!=nil && (self.heightF!=srcFrame.height || self.widthF!=srcFrame.width)){
      self.heightF = srcFrame.height;
      self.widthF = srcFrame.width;
      [self.xMagicKit setRenderSize:CGSizeMake(srcFrame.width, srcFrame.height)];
    }

    YTProcessInput *input = [[YTProcessInput alloc] init];
    input.textureData = [[YTTextureData alloc] init];
    input.textureData.texture = srcFrame.textureId;
    input.textureData.textureWidth = srcFrame.width;
    input.textureData.textureHeight = srcFrame.height;
    input.dataType = kYTTextureData;
    YTProcessOutput *output = [self.xMagicKit process:input withOrigin:YtLightImageOriginTopLeft withOrientation:YtLightCameraRotation0];
    dstFrame.textureId = output.textureData.texture;
    return 0;
}
```

集成效果

下图是 TUICallKit 含 UI 组件集成腾讯特效 SDK 的效果：


1:11
00 : 16
切换到语音通话
麦克风
扬声器
摄像头
挂断

云端录制（TUICallKit）

最近更新时间：2023-07-11 18:04:02

本文将介绍如何开启 TUICallKit 的云端录制，方便重要通话的存档、审核等，我们提供了两个方案供您选择：自动录制方案 和 REST API 录制方案 。

> **说明：**
> TUICallKit 整合有多个腾讯云基础的 PaaS 服务，其中音视频相关能力依赖于 实时音视频 TRTC ，因为 TUICallKit 的云端录制功能需要进入 实时音视频控制台 进行配置。

方案一：自动录制方案（推荐）

我们推荐您使用自动录制的方案，录制不需业务方来启动和停止，由腾讯云实时音视频后台来管理录制任务，通话过程中有音视频流上行时就自动录制，**接入过程快速、简单**，您可以通过如下几步完成：

1. 在 实时音视频控制台 > 应用管理 找到对应 SDKAppId 的应用，进入功能配置页，详见下图：



2. 在功能配置页，您可以看到云端录制配置的卡片，**启用云端录制**功能后，单击**创建全局自动录制模板**。



3. 根据音视频通话的业务场景（1v1通话、群组通话），我们推荐如下配置参数，当然您也可以根据自己的业务需求配置自定义录制模板。


全局自动录制模板
录制模式 *
单流录制
合流录制
同时勾选单流和合流，将会产生两份录制文件，产生两份录制费用
录制格式 *
音视频格式
纯音频格式
文件格式 *
MP4
HLS
音视频-MP4格式
基本参数
单个录制文件时长
1440
分钟
文件大小超过 2GB 将会被拆分
续录等待时长
5
s
续录等待时长会直接影响录制文件生成的时间
移除音频
合流输出参数
视频参数
推荐参数：
标清
高清
超高清
画面分辨率
960
px
视频宽度限制在[16，2560]且视频宽度*视频高度不得超过2560*1440
540
px
视频高度限制在[16，2560]且视频宽度*视频高度不得超过2560*1440
视频码率
1000
kbps
视频帧率
15
fps
GOP
2
s
音频参数
采样率
48000 Hz
44100 Hz
16000 Hz
音频码率
48
kbps
声道
单声道
双声道
布局参数
混流布局模板
九宫格模板
屏幕分享模板
悬浮模板
混流用户数
8
录制文件存储 *
云点播
指定点播应用 *
主应用
保存时间 *
永久保存
指定时间
高级设置

> **注意：**
> - 全局录制支持的最大混流人数为8人。如果通话人数超过8人（包括自己），则最后一个用户的流无法被录制。
> - 开启全局自动录制功能后，在通话接听后且有音视频上行，就会触发自动启动录制任务，在通话结束后会自动停止录制，如果因为网络或者其他情况异常退房，我们的录制后台会根据您设置的 MaxIdleTime 值（空闲等待时间，默认5s）自动停止录制任务，避免造成额外的计费损失。

4. **模板停止进程创建完成后，勾选全局自动录制即可完成。**



方案二：REST API 录制方案

如果自动录制方案不满足您的业务需求，您也可以采用更加灵活的 REST API 录制方案，可以指定录制订阅房间内的主播，自定义合流布局，录制中途更新布局和订阅等，但是**需要搭配业务后台服务，接入复杂，功能强大，关键步骤如下：**

1. 在 实时音视频控制台 > 应用管理 找到对应 SDKAppId 的应用，进入功能配置页，详见下图：



2. 在功能配置页，您可以看到云端录制配置的卡片，**启用云端录制**功能，此时默认勾选**手动自定义录制**，即 REST API 模式。



3. 然后您就可以调用 REST API（CreateCloudRecording）来启动云端的录制，这里建议您可以通过监听 TUICallObserver 的通知事件，在音视频通话开始的时候开启录制，以 Java 代码为例：

```
TUICallEngine.createInstance(context).addObserver(new TUICallObserver() {
    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole) {
        // 通知您的业务后台，使用REST API 启动录制任务。
    }
});
```

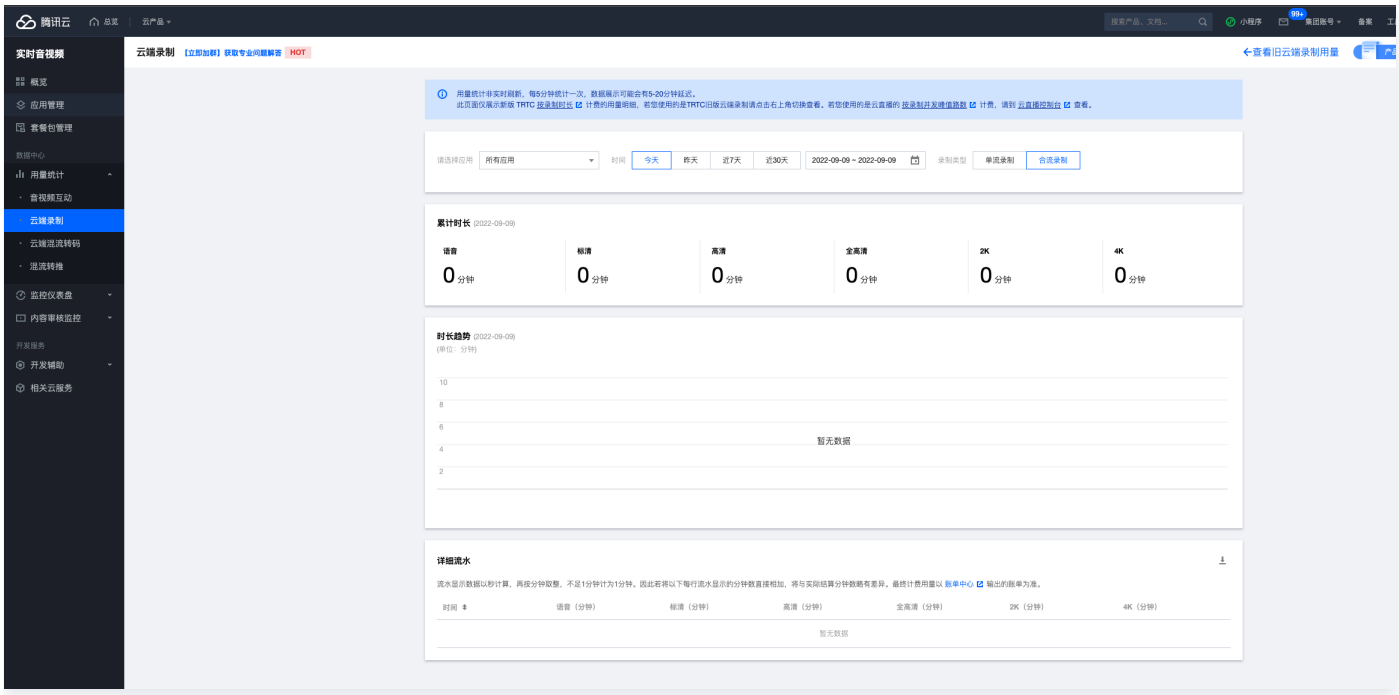
4. 考虑到客户端有可能会存在网络差、停止进程的情况造成的通话异常挂断，所以如何结束录制，建议您可以通过订阅 TRTC 房间状态的回调（详情请参见 监听服务端事件回调 ），在收到 TRTC 房间状态解散的回调后，调用 REST API （ DeleteCloudRecording ）来停止云端的录制任务。



常见问题

1. 如何查看录制时长明细?

您可以在 实时音视频控制台 > 云端录制 查看录制的一些时长明细：



2. 如何查看录制的文件?

您可以登录 云点播控制台，在左侧导航栏选择**媒资管理**，单击列表上方的**前缀搜索**，选择**前缀搜索**，在搜索框输入关键词，录制文件命名规则如下：

- **单流录制 MP4 文件名规则：** `<SdkAppId>_<RoomId>_UserId_s_<UserId>_UserId_e_<MediaId>_<Index>.mp4`
- **合流录制 MP4 文件名规则：** `<SdkAppId>_<RoomId>_<Index>.mp4`

更多文档

- 实现云端录制与回放
- 监听服务端事件回调
- 云端录制计费说明

通话记录（TUICallKit）
通话记录简介

最近更新时间：2023-08-15 18:02:12

通话记录是音视频通话 SDK（TUICallKit）的配套服务，提供 TUICallKit 产生的通话记录数据查看、查询等功能，具体包含 通话记录看板、通话状态回调 和 REST API 三种服务内容，您可按需取用。

服务内容

服务类型	描述	版本要求
通话记录看板	提供 TUICallKit 的通话数据统计及通话记录可视化看板，登录控制台即可查看相关数据。	开通**群组通话版**的应用（SDKAppId）可使用
通话状态回调	提供实时通话结果回调	开通**群组通话版**的应用（SDKAppId）可使用
REST API	提供一种便捷获取通话记录的方法	**目前属于内测期**，内测期间开通**群组通话版**的应用（SDKAppId）可使用

使用条件

目前各平台仅在特定版本的 TUICallKit 可使用通话记录服务，详见下表：

平台/框架	版本号
Android/iOS/Flutter/uni-app（客户端）	>= 1.7.1
Web	>= 1.4.6
微信小程序	>= 1.5.1

注意：
参与通话的所有平台/框架升级到上述新的版本后，才可查看到对应的通话信息。

版权所有：腾讯云计算（北京）有限责任公司

通话记录看板

最近更新时间：2023-12-29 17:31:22

通话记录看板是音视频通话能力（TUICallKit）的配套服务，提供 TUICallKit 的通话数据统计及通话记录可视化看板。服务包含通话次数、通话人数、最大并发通话数等统计数据，以及通话发起时间、通话参与人数、通话持续时间等单条通话记录信息。

使用条件

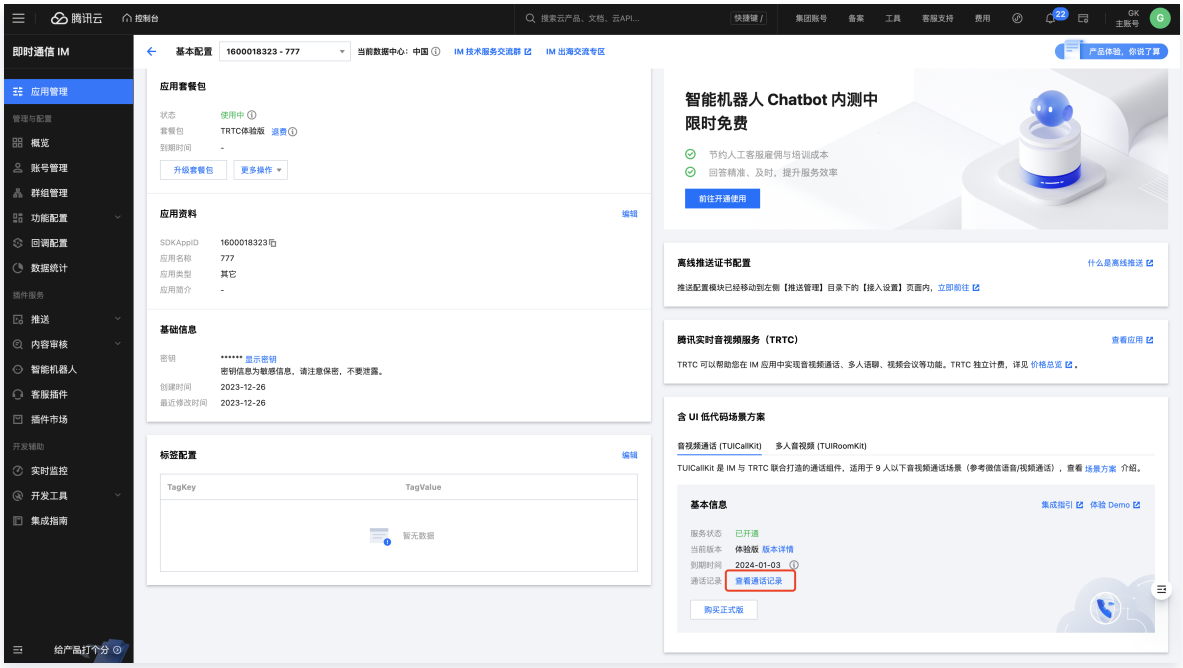
- 仅开通了音视频通话能力（TUICallKit）**群组通话版**的应用（SDKAppId）可以使用通话记录功能，您也可以开通体验版免费试用，版本说明及开通指引参考 音视频通话能力版本 。
- 目前各平台仅在特定版本的 TUICallKit 可使用通话记录看板，详见下表：

平台/框架	版本号
Android/iOS/Flutter/uni-app（客户端）	⩾ 1.7.1
Web	⩾ 1.4.6
微信小程序	⩾ 1.5.1

> **注意：**
> 参与通话的所有平台/框架升级到上述新的版本后，才可在控制台上查看到对应的通话信息。

使用指引

1. 登录 即时通信IM 控制台，单击目标应用卡片，进入应用的基础配置页面。
2. 在页面的右下角找到**含 UI 低代码场景方案 > 音视频通话（TUICallKit）**功能区。
3. 您可通过点击**查看通话记录**，进入通话记录页面。



4. 若您为首次使用通话记录，需阅读并确认服务开通界面说明，勾选**我已知晓并同意《腾讯云视立方 · 音视频通话 SDK 个人信息保护规则》**，随后单击 **立即开通**。



数据指标及功能

统计数据



数据指标

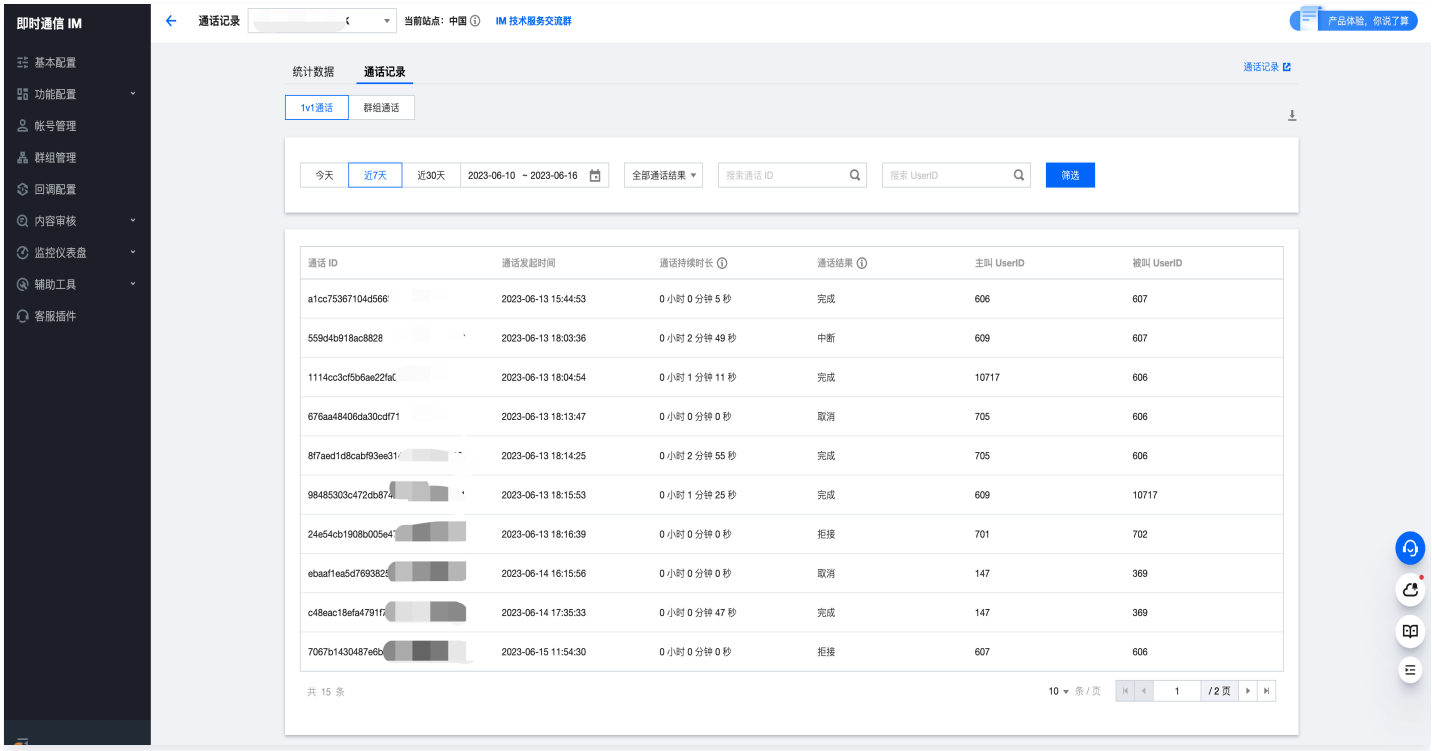
指标	定义
通话次数	发起通话的总次数，不论是否接通均统计在内。
通话人数	成功加入通话的 UserID 去重后的总数。 如收到了通话邀请但未接听的 UserID 不会统计在内；如一个 UserID 在统计时间范围内参与多场通话，在通话人数中也只记录为1。
最大并发通话数	同时进行通话的最大通话数量。 在今日通话总览和本月通话总览中，最大并发通话数是1v1通话和群组通话2种方式的最大值，如统计时间范围内1v1通话的最大并发通话数是20，群组通话的最大并发通话数是10，那么对应数据将为20。

数据筛选条件

筛选条件	定义

时间	最多可查询近一年统计数据，最小时间粒度为天。
通话类型	分为1v1通话、群组通话和全部通话三个统计维度。 今日通话总览和本月通话总览均为全部通话统计结果。

通话记录



数据指标

通话类型	指标	定义
1v1通话	通话 ID	每一条通话记录的唯一标识。
	通话发起时间	发起通话的时刻，时间粒度到秒，如：2023-5-20 10:28:08。
	通话持续时长	通话接通后的持续时长。
	通话结果	1. 取消：主叫方在接通前取消通话。 2. 拒接：被叫方拒接。 3. 未接听：被叫方超时未接听。 4. 完成：通话接通并正常结束。 5. 忙线：通话忙线。 6. 中断：网络等原因导致通话中断。 未知原因：未知原因。如本地网络、进程异常中断等原因导致的数据上报失败。
	主叫 UserID	发起通话的 UserID。
	被叫 UserID	接听通话的 UserID。

群组通话	通话 ID	每一条通话记录的唯一标识。
	通话发起时间	发起通话的时刻，时间粒度到秒，如：2023-5-20 10:28:08。
	通话持续时长	通话参与人数大于1人的持续时长。
	同时通话人数峰值	同时在线通话人数的峰值。
	参与人数	成功加入过群组通话的人数，以 UserID 去重，点击对应数字可查看详情，包含： 1. UserID：成功加入通话的 UserID。 2. 通话时长：通话参与人数大于1人时，对应 UserID 的通话持续时长。 3. 平台：接入通话的平台，分为iOS、Android、Web 和微信小程序。

数据筛选条件

筛选条件	指标
通话类型	1v1通话和群组通话。
时间	最多可查询近90天通话记录，最小查询时间粒度为天。
通话结果	仅1v1通话支持，通话结果分类及定义参见 通话记录 > 数据指标 。
通话 ID	输入通话 ID 进行查询。
UserID	输入 UserID 进行查询。

通话状态回调
通话状态回调

最近更新时间：2024-01-11 15:42:52

为了方便您精细化控制您的通话业务，音视频通话 SDK（TUICallKit）提供了通话状态回调。您的业务后台可以通过该回调实时查看用户的通话结果，如未接听、拒接等，您可以据此进行实时数据统计等操作。回调配置方式参见 回调配置 API 列表 。

使用条件

- 仅开通了音视频通话 SDK（TUICallKit）**群组通话版**的应用（SDKAppId）可以使用通话状态回调，您也可以开通体验版免费试用，版本说明及开通指引参考 音视频通话能力版本 。
- 目前各平台仅在特定版本的 TUICallKit 可使用通话状态回调，详见下表：

平台/框架	版本号
Android/iOS/Flutter/uni-app（客户端）	>= 1.7.1
Web	>= 1.4.6
微信小程序	>= 1.5.1

> **注意：**
> 参与通话的所有平台/框架升级到上述新的版本后，才可在控制台上查看到对应的通话信息。

注意事项

- 要启用回调，必须配置回调 URL，并打开本条命令字的开关，请参考 创建回调配置 。
- 回调的方向是 Callkit 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

可能触发该回调的场景

App 用户通过客户端进行通话所产生的动作，例如挂断等。

回调的发生时机

通话结束后。

可能发生的回调结果

回调状态	结果表示
未接听：被叫方超时未接听	not_answer
拒接：被叫方拒接	reject
忙线：通话忙线	call_busy
取消：主叫方在接通前取消通话	cancel
完成：通话接通并正常结束	normal_end
中断：网络等原因导致通话中断	interrupt

接口说明

请求URL

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
$http://www.example.com?
sdkappid=$sdkappid&command=$command&contenttype=json&clientip=$clientip&optplatform=$optplatfo
rm
```

请求参数说明

参数	说明
http	请求协议为 HTTPS 或 HTTP，请求方式为 POST
www.example.com	回调 URL
sdkappid	创建应用时在即时通信 IM 控制台分配的 SDKAppID
command	请参见：回调命令字列表
contenttype	固定值为 json
clientip	客户端 IP，格式如：127.0.0.1
optplatform	客户端平台，可能是 iOS、Android、Web、miniProgram

具体的回调内容在 HTTP 请求包体中，参见下文回调示例。

回调示例

回调请求示例：

```
POST /?sdkappid=8888888&command=call_end&contenttype=json&clientip=127.0.0.1&optplatform=iOS
HTTP/1.1
Host: www.example.com
Content-Length: 337
{
   "UserId": "Alice",
   "RoomId": "Alice's Room",
   "TotalNum": 2,
   "MediaType": "audio",
   "CallType": "single",
   "CallId": "aheahfo-eqwnknoihfsd-qweqf",
   "Role": "caller",
   "CallResult": "normal_end",
   "EventTime": 170485688,
   "StartCallTs": 1704856873,
   "AcceptTs": 1704856876,
   "EndTs": 1704856885
}
```

请求包字段说明

字段	类型	说明
UserId	String	操作的用户 ID
RoomId	String	操作的房间 ID
TotalNum	Integer	参与通话人数
MediaType	String	媒体类型： • audio 音频通话 • video 视频通话
CallType	String	通话类型： • single 音频通话 • group 视频通话
CallId	String	通话唯一 ID
Role	String	角色： • caller 主叫 userId • callee 被叫 userId
CallResult	String	通话结果，仅在结束事件中填写，其余为空： • cancel 取消：主叫方在接通前取消通话 • reject 拒接：被叫方拒接 • not_answer 未接听：被叫方超时未接听 • normal_end 完成：通话接通并正常结束 • call_busy 忙线：通话忙线 • interrupt 中断：网络等原因导致通话中断
EventTime	Integer	操作发生的时间戳（秒级）
StartCallTs	Integer	通话发起的时间戳（秒级）只有在normal_end才会回调此值
AcceptTs	Integer	通话接听的时间戳（秒级）只有在normal_end才会回调此值
EndTs	Integer	通话结束的时间戳（秒级）只有在normal_end才会回调此值

说明：
CallResult 字段仅一对一通话可能显示所有类型，群聊仅有 normal_end 。

回调应答示例

App 后台同步数据后，发送回调应答包。

```
{
  "ErrorCode": 0,
  "ErrorMessage": "Success"
}
```

应答包字段说明

字段	类型	说明
ErrorCode	Integer	0 为成功，其余为失败
ErrorMessage	String	您的服务器响应可以自定义携带错误信息

> **注意：**
> 您的服务器在正确接收到请求包时，建议您及时回复正确的应答包，否则会触发系统重试，重试机制如下。

重试机制

- 触发重试的时机

  ErrorCode = 0 视为回调成功，否则会触发重试。
  向您的 appServer 发 http 请求报错也会触发重试。
- 重试间隔

  重试间隔为 `0s, 1s, 1s, 2s, 3s, 5s` 若均为成功则放弃。

错误码

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
999	回调不存在
70001	请求参数无效，请检查必填参数是否缺失或填错
70002	UserSig 无效
70003	UserSig 过期
70004	请求用户非超级管理员
70005	请求被限频
未知错误码	系统内部错误，请 提交工单 联系技术人员

通话事件回调

最近更新时间：2024-01-11 15:42:51

为了方便您精细化控制您的通话业务，音视频通话 SDK（TUICallKit）提供了通话状态回调。您的业务后台可以通过该回调实时查看用户的通话结果，如未接听、拒接等，您可以据此进行实时数据统计等操作。回调配置方式参见 回调配置 API 列表 。

使用条件

- 仅开通了音视频通话 SDK（TUICallKit）**群组通话版**的应用（SDKAppId）可以使用通话状态回调，您也可以开通体验版免费试用，版本说明及开通指引参见 音视频通话能力版本 。
- 目前各平台仅在特定版本的 TUICallKit 可使用通话状态回调，详见下表：

平台/框架	版本号
Android/iOS/Flutter/uni-app（客户端）	⩾ 1.7.1
Web	⩾ 1.4.6
微信小程序	⩾ 1.5.1

> **注意：**
> 参与通话的所有平台/框架升级到上述新的版本后，才可在控制台上查看到对应的通话信息。

注意事项

- 要启用回调，必须配置回调 URL，并打开本条命令字的开关，请参见 创建回调配置 。
- 回调的方向是 Callkit 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。

可能触发该回调的场景

App 用户通过客户端进行通话所产生的动作，例如接听、挂断等。

回调的发生时机

接听、挂断等用户操作后。

可能发生的回调结果

请参见 回调命令字-通话事件回调 。

接口说明

请求URL

以下示例中 App 配置的回调 URL 为 `https://www.example.com` 。
示例：

```
$http://www.example.com?
sdkappid=$sdkappid&command=$command&contenttype=json&clientip=$clientip&optplatform=$optplatfo
rm
```

请求参数说明

参数	说明
http	请求协议为 HTTPS 或 HTTP，请求方式为 POST
www.example.com	回调 URL
sdkappid	创建应用时在即时通信 IM 控制台分配的 SDKAppID
command	请参见：回调命令字列表
contenttype	固定值为 json
clientip	客户端 IP，格式如：127.0.0.1
optplatform	客户端平台，可能是 iOS、Android、Web、miniProgram

具体的回调内容在 HTTP 请求包体中，参见下文回调示例。

回调示例

回调请求示例：

```
POST /?
sdkappid=8888888&command=caller_start_call&contenttype=json&clientip=127.0.0.1&optplatform=iOS
HTTP/1.1
Host: www.example.com
Content-Length: 337
{
   "UserId": "Alice",
   "RoomId": "Alice's Room",
   "TotalNum": 2,
   "MediaType": "audio",
   "CallType": "single",
   "CallId": "aheahfo-eqwnknoihfsd-qweqf",
   "Role": "caller",
   "Event": "start_call",
   "CallResult": "",
   "EventTime": 1704695566,
   "StartCallTs": 1704856873,
   "AcceptTs": 1704856876,
   "EndTs": 1704856885
}
```

请求包字段说明

字段	类型	说明
UserId	String	操作的用户 ID
RoomId	String	操作的房间 ID
TotalNum	Integer	参与通话人数
CallType	String	通话类型：

		• single 音频通话 • group 视频通话
CallId	String	通话唯一 ID
Role	String	角色： • caller 主叫 userId • callee 被叫 userId
Event	String	通话事件： • start_call 主叫发起通话 • call_accepted 主叫接听通话 • call_missed 主叫未接听 • call_rejected 主叫拒绝通话 • call_busy 主叫通话忙线 • cancel_call 主叫取消通话 • call_failed 主叫发起通话失败 • call_end 主叫通话正常结束 • call_interrupted 主叫通话中断 • receive_call 被叫接收通话 • accept_call 被叫接听通话 • not_answer_call 被叫未应答 • reject_call 被叫拒接 • ignore_call 被叫忽略通话 • call_canceled 被叫取消通话 • call_end 被叫通话正常结束 • call_interrupted 被叫通话异常中断 • invite_user 中途邀请用户 • join_in_group_call 中途加入通话
StartCallTs	Integer	通话发起的时间戳（秒级）只有在normal_end才会回调此值
AcceptTs	Integer	通话接听的时间戳（秒级）只有在normal_end才会回调此值
EndTs	Integer	通话结束的时间戳（秒级）只有在normal_end才会回调此值

说明：
CallResult 字段仅一对一通话可能显示所有类型，群聊仅有 `normal_end` 。
Event 字段中列举了所有可能，但部分事件概率出现较小。
TotalNum 通话总人数可能在事件同步时有一定的误差。

回调应答示例

App 后台同步数据后，发送回调应答包。

```
{
  "ErrorCode": 0,
```

```
  "ErrorMessage": "Success"
}
```

应答包字段说明

字段	类型	说明
ErrorCode	Integer	0 为成功，其余为失败
ErrorMessage	String	您的服务器响应可以自定义携带错误信息

> **注意：**
> 您的服务器在正确接收到请求包时，建议您及时回复正确的应答包，否则会触发系统重试，重试机制如下。

重试机制

- 触发重试的时机

  ErrorCode = 0 视为回调成功，否则会触发重试。
  向您的 appServer 发 http 请求报错也会触发重试。
- 重试间隔

  重试间隔为 `0s, 1s, 1s, 2s, 3s, 5s` 若均为成功则放弃。

错误码

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
999	回调不存在
70001	请求参数无效，请检查必填参数是否缺失或填错
70002	UserSig 无效
70003	UserSig 过期
70004	请求用户非超级管理员
70005	请求被限频
未知错误码	系统内部错误，请 提交工单 联系技术人员

回调配置
回调配置 API 列表

最近更新时间：2024-01-11 15:42:52

回调信息配置

功能说明	接口
创建回调	v1/callback/set
查询回调	v1/callback/get
更新回调	v1/callback/update
删除回调	v1/callback/delete

回调命令字列表

通话状态回调	通话事件回调
• cancel 取消：主叫方在接通前取消通话 • reject 拒接：被叫方拒接 • not_answer 未接听：被叫方超时未接听 • normal_end 完成：通话接通并正常结束 • call_busy 忙线：通话忙线 • interrupt 中断：网络等原因导致通话中断	• caller_start_call 主叫发起通话 • caller_call_accepted 主叫接听通话 • caller_call_missed 主叫未接听 • caller_call_rejected 主叫拒绝通话 • caller_call_busy 主叫通话忙线 • caller_cancel_call 主叫取消通话 • caller_call_failed 主叫发起通话失败 • caller_call_end 主叫通话正常结束 • caller_call_interrupted 主叫通话中断 • callee_receive_call 被叫接收通话 • callee_accept_call 被叫接听通话 • callee_not_answer_call 被叫未应答 • callee_reject_call 被叫拒接 • callee_ignore_call 被叫忽略通话 • callee_call_canceled 被叫取消通话 • callee_call_end 被叫通话正常结束 • callee_call_interrupted 被叫通话异常中断 • invite_user 中途邀请用户 • join_in_group_call 中途加入通话

版权所有：腾讯云计算（北京）有限责任公司

创建回调配置

最近更新时间：2024-01-11 15:42:52

功能说明

管理员可通过该接口创建回调。

接口说明

> ⚠ **注意：**
> 接口多次调用以最后一次的结果为准。

请求URL示列

```
https://xxxxxx/v1/callback/set?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国：`callkit.trtc.tencent-cloud.com`
v1/callback/set	请求接口
sdkappid	创建应用时控制台分配的 sdkappid
identifier	必须为 IM App 管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围 0 - 4294967295
contenttype	请求格式固定值为 `json`

最高调用频率

10次/秒。

请求包示例

```
{
  "address":"http://www.example.com/callback",
  "actions":[
    "call_busy",
    "normal_end",
    "caller_start_call",
    "invite_user",
    "callee_reject_call"
  ]
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

请求字段说明

字段	类型	属性	说明
address	String	必填	回调的地址，需要以 http/https 开头，建议使用更为安全的 https
actions	Array	必填	需要触发回调的情景，列表请参见 回调命令字列表

应答包体示例

```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "a1d8543a9b1daef5d0f0c21517a4bc0a",
  "data": "http://www.example.com/callback"
}
```

应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
data	String	配置成功的回调地址

错误码说明

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
70001	回调地址必须 http 或 https 开头

版权所有：腾讯云计算（北京）有限责任公司

查询回调配置

最近更新时间：2023-08-15 17:39:52

功能说明

管理员可通过该接口查询已配置的回调。

接口说明

请求URL示列

```
https://xxxxxx/v1/callback/get?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国： callkit.trtc.tencent-cloud.com
v1/callback/get	请求接口
sdkappid	创建应用时控制台分配的 sdkappid
identifier	必须为 IM App 管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围0 - 4294967295
contenttype	请求格式固定值为 json

最高调用频率

10次/秒。

请求包示例

传空即可。

```
{

}
```

应答包体示例

```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "355c3c394f0602ed81a13c34999abebb",
```

```
  "data": {
    "actions": [
      "call_busy",
      "normal_end"
    ],
    "address": "http://www.example.com/callback"
  }
}
```

应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
actions	Array	已配置的回调 actions
address	String	已配置的回调地址

错误码说明

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
70001	回调地址必须 http 或 https 开头

更新回调配置

最近更新时间：2023-08-15 17:39:52

功能说明

管理员可通过该接口更新回调。

接口说明

请求URL示列

```
https://xxxxxx/v1/callback/update?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国：callkit.trtc.tencent-cloud.com
v1/callback/update	请求接口
sdkappid	创建应用时控制台分配的 sdkappid
identifier	必须为 IM App 管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围0 - 4294967295
contenttype	请求格式固定值为 json

最高调用频率

10次/秒。

请求包示例

```
{
  "address": "https://www.example2.com/callback",
  "actions": [
    "call_busy",
    "cancel",
    "normal_end"
  ]
}
```

请求字段说明

字段	类型	属性	说明

address	String	必填	回调的地址，需要以 http/https 开头，建议使用更为安全的 https
actions	Array	必填	需要触发回调的情景，列表请参考 回调命令字列表

应答包体示例

```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "5b0fa500064397cad3554506e27e18e1",
  "data": "https://www.example2.com/callback"
}
```

应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
data	String	更新成功的回调地址

错误码说明

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
999	没有配置信息

版权所有：腾讯云计算（北京）有限责任公司

删除回调配置

最近更新时间：2023-08-15 17:39:53

功能说明

管理员可通过该接口删除回调。

接口说明

请求URL示列

```
https://xxxxxx/v1/callback/delete?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国： callkit.trtc.tencent-cloud.com
v1/callback/delete	请求接口
sdkappid	创建应用时控制台分配的 sdkappid
identifier	必须为 IM App 管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围 0 - 4294967295
contenttype	请求格式固定值为 json

最高调用频率

10次/秒。

请求包示例

传空即可。

```
{

}
```

应答包体示例

```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "5b0fa500064397cad3554506e27e18e1",
```

```
  "data": ""
}
```

应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
data	String	无数据

错误码说明

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用

REST API
REST API 简介

最近更新时间：2023-08-18 10:16:22

REST API 是音视频通话 SDK 的后台 HTTP 管理接口，可以为开发者提供一套简单的管理入口。目前音视频通话 SDK 支持的 REST API 请参见 REST API 接口列表 。
为了安全性，REST API 仅提供 HTTPS 接口。

使用条件

- **REST API 目前处于内测期**，内测期间开通了音视频通话 SDK（TUICallKit）**群组通话版**的应用（SDKAppId）可以使用REST API，您也可以开通体验版免费试用，版本说明及开通指引参考 音视频通话能力版本 。
- 目前各平台仅在特定版本的 TUICallKit 可使用 REST API，详见下表：

平台/框架	版本号
Android/iOS/Flutter/uni-app（客户端）	>= 1.7.1
Web	>= 1.4.6
微信小程序	>= 1.5.1

注意：
- 参与通话的所有平台/框架升级到上述新的版本后，才可在控制台上查看到对应的通话信息。
- 内测期 REST API 支持查询近7天数据。

REST API 接口列表

功能说明	接口
通过 callId 获取记录	v1/records/get_record_by_callId
通过条件获取记录	v1/records/get_records_by_filter

调用方法

请求 URL

REST API 的 URL 格式如下：

```
https://xxxxxx/$version/$kind/$command?
sdkappid=$SDKAppID&identifier=$identifier&usersig=$usersig&random=99999999&contenttype=json
```

其中各个参数的含义以及取值如下（参数名称及其取值均区分大小写）：

参数	含义	取值
https	请求协议	请求协议为 HTTPS，请求方式为 POST
xxxxxx	专属域名	中国 callkit.trtc.tencent-cloud.com

version	协议版本号	固定为 `v1`
kind	管理分类	示列：`v1/records/get_record_by_callId`，其中 records 为 kind
command	命令字，与 kind 组合用来表示具体的业务功能	示列：`v1/records/get_record_by_callId`，其中 get_record_by_callId 为 command
sdkappid	在控制台获取的应用标识	在申请接入时获得
identifier	用户名，调用 REST API 时必须为 App 管理员账号	使用 IM 的管理员账号
usersig	用户名对应的密码	参见 生成 UserSig
random	标识当前请求的随机数参数	32位无符号整数随机数，取值范围 0 – 4294967295
contenttype	请求格式	固定值为 json

注意：

- 领取或购买套餐包后会在 IM 账号系统中创建 administrator 管理员账号，在请求中的 `identifier` 参数中请使用 administrator，如果您在 IM 账号管理 中取消或删除管理员，请正确指定管理员账号以及对应的 userSig。
- App 可以在每次调用 REST API 时都生成管理员账号的 UserSig，亦可生成一个固定的 UserSig 重复使用，但需特别注意 UserSig 的有效期。
- 在创建房间以及进入房间等操作时，系统会在 IM 系统中自动导入 IM 账号，敬请知悉。

HTTP 请求包体格式

REST API 仅支持 POST 方法，其请求包体为 JSON 格式，具体的包体格式参见每个 API 的详细描述。

注意：

POST 包体不能为空，即使某条协议包体中不用携带任何信息，也必须要携带一个空的 JSON 对象，即 `{}` 。

HTTP 返回码

除非发生网络错误（例如502错误），否则 REST API 的调用结果均为200，实际 API 调用的错误码与错误信息在 HTTP 应答包体中返回。

HTTP 应答包体格式

REST API 的应答包体也是 JSON 格式，其格式符合如下特征：

```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "1c8960ac38d61be38b6fb219db0182d1",
  "data": {}
}
```

应答包体中必然包含 errorCode、errorMessage、requestId 这三个属性，其含义如下：

字段	类型	说明
errorCode	String	错误码，0为成功，其余为失败
errorMessage	String	错误信息
requestId	Integer	请求唯一标识符

调用示例

以下为通过 REST API 获取指定 callId 的通话记录示例。
HTTPS 请求：

```
POST /records/get_record_by_callId?
usersig=xxx&identifier=admin&sdkappid=88888888&random=99999999&contenttype=json HTTP/2
Host: callkit.trtc.tencent-cloud.com
Content-Length: 36
{
   "callId": "2ae7d549-c441-4a9b-87c0-61810fe19582"
}
```

HTTPS 应答：

```
HTTP/2 200 OK
Date: Fri, 21 Apr 2023 06:06:16 GMT
Content-Length: 112
Connection: keep-alive

{
   "errorCode": 0,
   "errorMessage": "Success",
   "requestId": "9f01db503f2006ad10c45f4f4609e38d",
   "data": {
      "callId": "2ae7d549-c441-4a9b-87c0-61810fe19582",
      "sdkAppId": 88888888,
      "mediaType": "video",
      "roomId": "123456",
      "startCallTs": 1688705638,
      "acceptTs": 1688705641,
      "endTs": 1688705668,
      "caller": "alice",
      "totalUserNumber": 2,
      "callType": "single",
      "callResult": "normal_end",
      "callees": [
         "bob1",
         "bob2"
      ]
   }
}
```

REST API 公共错误码

错误码	含义说明

0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
70001	请求参数无效，请检查必填参数是否缺失或填错
70002	UserSig 无效
70003	UserSig 过期
70004	请求用户非超级管理员
70005	请求被限频
70009	解析请求 body 出错，请检查请求参数类型是否正确
未知错误码	系统内部错误，请 提交工单 联系技术人员

常见问题

REST API 请求有概率超时，收不到任何响应?

您可以从以下几个方面确认：

1. 后台 REST 接口设置的超时时间是 3s，调用方设置的超时时间应该长于 3s。
2. `telnet callkit.trtc.tencent-cloud.com 443` 确认能否连接服务端口。
3. 使用 `curl -I https://callkit.trtc.tencent-cloud.com` 简单测试看状态码是否为200。
4. 确认机器的 dns server 配置是内网 dns server，还是公共 dns server。如果是内网 dns server，请确保 dns server 网络出口和本机器网络出口 IP 所在地域运营商一致。
5. 建议业务调用方使用**长连接+连接池**模式。

通过 callId 获取记录

最近更新时间：2023-08-18 10:16:22

功能说明

管理员可通过该接口获取指定 callId 的通话记录。

> **说明：**
> REST API 目前处于内测期，您可以查询创建时间在7天内的通话数据。

接口调用说明

请求URL示列

```
https://xxxxxx/v1/records/get_record_by_callId?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国站：`callkit.trtc.tencent-cloud.com`
v1/records/get_record_by_callId	请求接口
sdkappid	创建应用时控制台分配的 SDKAppID
identifier	使用 IM 的管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围 0 - 4294967295
contenttype	请求格式固定值为 `json`

最高调用频率

20次/秒。

请求包示例

```
{
  "callId": "2ae7d549-c441-4a9b-87c0-61810fe19582"
}
```

请求包字段说明

字段	类型	属性	说明
callId	String	必填	通话唯一 ID

应答包体示例


```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "3fdc344c7a67461c6af3cfc3e77744b5",
  "data": {
    "callId": "2ae7d549-c441-4a9b-87c0-61810fe19582",
    "sdkAppId": 88888888,
    "mediaType": "video",
    "roomId": "123456",
    "startCallTs": 1688705638,
    "acceptTs": 1688705641,
    "endTs": 1688705668,
    "caller": "alice",
    "totalUserNumber": 2,
    "callType": "single",
    "callResult": "normal_end",
    "callees": [
      "bob1",
      "bob2"
    ]
  }
}
```


应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
callId	String	通话的唯一 ID
sdkAppId	Integer	您的 sdkAppId
mediaType	String	媒体类型 • video 视频通话 • audio 音频通话
roomId	String	通话的房间 ID
startCallTs	Integer	通话发起的时间戳（秒级）
acceptTs	Integer	通话接通的时间戳（秒级）

版权所有：腾讯云计算（北京）有限责任公司

endTs	Integer	通话结束的时间戳（秒级）
caller	String	主叫 userId
totalUserNumber	Integer	参与通话的总人数
callType	String	通话类型： • single 一对一通话 • group 群组通话
callResult	String	通话结果： • cancel 取消：主叫方在接通前取消通话。 • reject 拒接：被叫方拒接。 • not_answer 未接听：被叫方超时未接听。 • normal_end 完成：通话接通并正常结束。 • call_busy 忙线：通话忙线。 • interrupt 中断：网络等原因导致通话中断。
callees	Array	被叫且参与通话的 userId 列表

说明：
callResult 字段仅一对一通话可能显示所有类型，群聊仅有 normal_end 。

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。实际的错误码、错误信息是通过应答包体中的 errorCode、errorMessage 来表示的。
公共错误码（70000到79999）请参见 错误码 。

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
70011	未找到任何记录
未知错误码	未知错误，请 提交工单 联系技术人员

通过条件获取记录

最近更新时间：2023-08-18 10:16:22

管理员可通过该接口获取指定条件的通话记录。

> **说明：**
> REST API 目前处于内测期，您可以查询创建时间在七天内的通话数据。

接口说明

请求URL示列

```
https://xxxxxx/v1/records/get_records_by_filter?
sdkappid=88888888&identifier=admin&usersig=xxx&random=99999999&contenttype=json
```

请求参数说明

下表仅列出调用本接口时涉及修改的参数及其说明，更多参数详情请参考 REST API 简介 。

参数	说明
xxxxxx	SDKAppID 所在国家/地区对应的专属域名： 中国：`callkit.trtc.tencent-cloud.com`
v1/records/get_records_by_filter	请求接口
sdkappid	创建应用时控制台分配的 SDKAppID
identifier	使用 IM 的管理员账号
usersig	App 管理员账号生成的签名，具体操作请参见 生成 UserSig
random	请输入随机的32位无符号整数，取值范围 0 - 4294967295
contenttype	请求格式固定值为 `json`

最高调用频率

200次/秒。

请求包示例

基础形式
创建房间

```
{
  "startTimestamp": 1618705638,
  "endTimestamp": 1618705738,
  "callResult": "normal_end",
  "callType": "single",
  "numberPerPage": 20,
  "page": 2
```

版权所有：腾讯云计算（北京）有限责任公司

```
}
```

请求包字段说明

字段	类型	属性	说明
startTimestamp	Integer	必填	通话开始时间戳（秒级）
endTimestamp	Integer	选填	通话结束时间戳（秒级），不填默认为7天
callResult	String	选填	通话结果，不填默认为所有结果
callType	String	选填	通话类型，不填默认为所有类型
numberPerPage	Integer	选填	每页查询数量，默认为：100
page	Integer	选填	查询页数，不填默认为第一页

应答包字段示列


```
{
  "errorCode": 0,
  "errorMessage": "Success",
  "requestId": "8ba14f64a1d92d38705eec775e1f3deb",
  "data": {
    "totalNum": 3,
    "page": 2,
    "callRecordList": [
      {
        "callId": "2ae7d549-c441-4a9b-87c0-61810fe19581",
        "sdkAppId": 88888888,
        "mediaType": "video",
        "roomId": "456",
        "startCallTs": 1688709003,
        "acceptTs": 1688709003,
        "endTs": 1689150030,
        "caller": "123",
        "totalUserNumber": 0,
        "callType": "single",
        "callResult": "offline",
        "callees": [
          "111",
          "123"
        ]
      },
      {
        "callId": "2ae7d549-c441-4a9b-87c0-61810fe19582",
        "sdkAppId": 88888888,
        "mediaType": "video",
        "roomId": "456",
        "startCallTs": 1688709303,
        "acceptTs": 1688709303,
        "endTs": 1689150030,
        "caller": "123",
        "totalUserNumber": 0,
```

```
        "callType": "single",
        "callResult": "offline",
        "callees": [
          "111",
          "123"
        ]
      },
      {
        "callId": "2ae7d549-c441-4a9b-87c0-61810fe19583",
        "sdkAppId": 88888888,
        "mediaType": "video",
        "roomId": "456",
        "startCallTs": 1688709903,
        "acceptTs": 1688709903,
        "endTs": 1689150030,
        "caller": "123",
        "totalUserNumber": 0,
        "callType": "single",
        "callResult": "offline",
        "callees": [
          "111",
          "123"
        ]
      }
    ]
  }
}
```


应答包字段说明

字段	类型	说明
errorCode	Integer	错误码，0为成功
errorMessage	String	错误信息
requestId	String	请求的唯一 ID
totalNum	Integer	此次查询的总数量
page	Integer	当 page > 0 时 说明还有数据，将 page+1 可继续请求之后的数据。
callRecordList	Array	请参见：单条通话记录说明

单条通话记录说明

字段	类型	说明
callId	String	通话的唯一 ID
sdkAppId	Integer	您的 sdkAppId
mediaType	String	媒体类型 • video 视频通话 • audio 音频通话

roomId	String	通话的房间 ID
startCallTs	Integer	通话发起的时间戳（秒级）
acceptTs	Integer	通话接通的时间戳（秒级）
endTs	Integer	通话结束的时间戳（秒级）
caller	String	主叫 userId
totalUserNumber	Integer	参与通话的总人数
callType	String	通话类型 • single 一对一通话 • group 群组通话
callResult	String	通话结果： • cancel 取消：主叫方在接通前取消通话。 • reject 拒接：被叫方拒接。 • not_answer 未接听：被叫方超时未接听。 • normal_end 完成：通话接通并正常结束。 • call_busy 忙线：通话忙线。 • interrupt 中断：网络等原因导致通话中断。
callees	Array	被叫且参与通话的 userId 列表

说明：
callResult 字段仅一对一通话可能显示所有类型，群聊仅有 normal_end 。

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。实际的错误码、错误信息是通过应答包体中的 errorCode、errorMessage 来表示的。
公共错误码（70000到79999）请参见 错误码 。

错误码	含义说明
0	请求成功
50001	当前应用需要购买 TUICallKit 群组通话版套餐包方可使用
70011	未找到任何记录
未知错误码	未知错误，请 提交工单 联系技术人员

API 参考（TUICallKit）
Android
API 概览

最近更新时间：2023-05-22 10:03:17

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的头像、昵称
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话

版权所有：腾讯云计算（北京）有限责任公司

hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调

onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

关键类型定义

API	描述
TUICallDefine.MediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallDefine.Role	通话的角色，枚举类型：主叫、被叫
TUICallDefine.Status	通话的状态，枚举类型：空闲、待接听、接听中
TUICallDefine.CallParams	通话扩展参数：离线推送消息显示内容
TUICommonDefine.RoomId	音视频房间 ID，支持数字、字符串两种类型
TUICommonDefine.Camera	摄像头Id参数，枚举类型：前摄、后摄
TUICommonDefine.AudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUICommonDefine.NetworkQualityInfo	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2023-12-29 17:31:22

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入TUICallKit 。

API 概览

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

API 详情

createInstance

创建 TUICallKit 的单例。

Kotlin

```
fun createInstance(context: Context): TUICallKit
```

Java

```
TUICallKit createInstance(Context context)
```

setSelfInfo

设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是 URL 格式。

Kotlin

```
fun setSelfInfo(nickname: String?, avatar: String?, callback: TUICommonDefine.Callback?)
```

Java

```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback)
```

参数如下表所示：

参数	类型	含义
nickname	String	目标用户的昵称
avatar	String	目标用户的头像

call

拨打电话（1v1通话）。

Kotlin

```
fun call(userId: String, callMediaType: TUICallDefine.MediaType)
```

Java

```
void call(String userId, TUICallDefine.MediaType callMediaType)
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

call

拨打电话（1v1通话），支持自定义房间号、通话邀请超时时间，离线推送内容等。

Kotlin

```
fun call(
    userId: String, callMediaType: TUICallDefine.MediaType,
    params: CallParams?, callback: TUICommonDefine.Callback?
)
```

Java

```
void call(String userId, TUICallDefine.MediaType callMediaType,
        TUICallDefine.CallParams params, TUICommonDefine.Callback callback)
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallDefine.CallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话。

> ⚠ 注意
> 使用群组通话前需要创建IM 群组，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Kotlin

```
fun groupCall(groupId: String, userIdList: List<String?>?, callMediaType: TUICallDefine.MediaType)
```

Java

```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType callMediaType);
```

参数如下表所示：

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

groupCall

发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。

> ⚠ 注意
> 使用群组通话前需要创建IM 群组，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Kotlin

```
fun groupCall(
```

```
    groupId: String, userIdList: List<String?>?,
    callMediaType: TUICallDefine.MediaType, params: CallParams?,
    callback: TUICommonDefine.Callback?
)
```

Java

```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType callMediaType,
        TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallDefine.CallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等

joinInGroupCall

加入群组中已有的音视频通话。

> ⚠ **注意**
> **加入群组中已有的音视频通话前，需要提前创建或加入IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。**
> **群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。**

Kotlin

```
fun joinInGroupCall(roomId: RoomId?, groupId: String?, callMediaType: TUICallDefine.MediaType?)
```

Java

```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId, TUICallDefine.MediaType
callMediaType);
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaT	TUICallDefine.Media	通话的媒体类型，比如视频通话、语音通话

ype	Type	

setCallingBell

设置自定义来电铃音。
这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

- 铃声设置后与设备绑定，更换用户，铃声依旧会生效。
- 如需恢复默认铃声，`filePath` 传空即可。

Kotlin

```
fun setCallingBell(filePath: String?)
```

Java

```
void setCallingBell(String filePath);
```

enableMuteMode

开启/关闭静音模式。
开启后，收到通话请求，不会播放来电铃声。

Kotlin

```
fun enableMuteMode(enable: Boolean)
```

Java

```
void enableMuteMode(boolean enable);
```

enableFloatWindow

开启/关闭悬浮窗功能。
默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

Kotlin

```
fun enableFloatWindow(enable: Boolean)
```

Java

```
void enableFloatWindow(boolean enable);
```

版权所有：腾讯云计算（北京）有限责任公司

TUICallEngine

最近更新时间：2023-09-04 16:59:01

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）

版权所有：腾讯云计算（北京）有限责任公司

setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

API 详情

createInstance

创建 TUICallEngine 的单例。

```
TUICallEngine createInstance(Context context);
```

destroyInstance

销毁 TUICallEngine 的单例。

```
void destroyInstance();
```

init

初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。

```
void init(int sdkAppId, String userId, String userSig, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在**实时音视频控制台** > **应用管理** > **应用信息**中查看 SDKAppID
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig
callback	TUICommonDefine.Callback	初始化回调，`onSuccess` 表示初始化成功

addObserver

添加回调接口，您可以通过这个接口，监听 `TUICallObserver` 相关的事件回调。

```
void addObserver(TUICallObserver observer);
```

removeObserver

移除回调接口。

```
void removeObserver(TUICallObserver observer);
```

call

拨打电话（1v1通话）。

> ⚠ **注意：**
> 字符串房间号目前只支持 Android、iOS、Flutter、Uniapp 平台，Web、小程序等其他平台后续支持，敬请期待！

```
void call(String userId, TUICallDefine.MediaType callMediaType,
        TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如：视频通话、语音通话
params	TUICallDefine.CallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

发起群组通话。

> ⚠ **注意：**
> 使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。
> 字符串房间号目前只支持 Android、iOS、Flutter、Uniapp 平台，Web、小程序等其他平台后续支持，敬请期待！

```
void groupCall(String groupId, List<String> userIdList, TUICallDefine.MediaType callMediaType,
            TUICallDefine.CallParams params, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List	目标用户的 userId 列表
callMediaType	TUICallDefine.MediaType	通话的媒体类型，例如：视频通话、语音通话
params	TUICallDefine.CallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

accept

接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。

版权所有：腾讯云计算（北京）有限责任公司

```
void accept(TUICommonDefine.Callback callback);
```

reject

拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。

```
void reject(TUICommonDefine.Callback callback);
```

ignore

忽略当前通话，当您作为被叫收到 onCallReceived() 的回调时，可以调用该函数忽略来电，此时主叫会收到 onUserLineBusy 的回调。
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。

```
void ignore(TUICommonDefine.Callback callback);
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

```
void hangup(TUICommonDefine.Callback callback);
```

inviteUser

邀请用户加入此次群组通话。
使用场景：一个群组通话中的用户主动邀请其他人时使用。

```
void inviteUser(List<String> userIdList, TUICallDefine.CallParams params,
                TUICommonDefine.ValueCallback callback);
```

参数如下表所示：

参数	类型	含义
userIdList	List	目标用户的 userId 列表
params	TUICallDefine.CallParams	通话参数扩展字段，例如：通话邀请超时时间，离线推送自定义内容等

> **注意：**
> 邀请他人进入当前通话，该场景下，自定义RoomId无效，SDK会邀请他人加入当前邀请者所在的房间。

joinInGroupCall

主动加入此次群组通话。
使用场景：群组内用户主动加入此次群组通话使用。

```
void joinInGroupCall(TUICommonDefine.RoomId roomId, String groupId,
                     TUICallDefine.MediaType callMediaType, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

switchCallMediaType

切换视频通话到语音通话。

```
void switchCallMediaType(TUICallDefine.MediaType callMediaType);
```

参数如下表所示：

参数	类型	含义
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话

startRemoteView

开始订阅远端用户的视频数据。

```
void startRemoteView(String userId, TUIVideoView videoView, TUICommonDefine.PlayCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
videoView	TUIVideoView	待渲染的视图

stopRemoteView

停止订阅远端用户的视频数据。

```
void stopRemoteView(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId

openCamera

开启摄像头。

版权所有：腾讯云计算（北京）有限责任公司

```
void openCamera(TUICommonDefine.Camera camera, TUIVideoView videoView, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
camera	TUICommonDefine.Camera	前置/后置 摄像头
videoView	TUIVideoView	待渲染的视图

closeCamera

关闭摄像头。

```
void closeCamera();
```

switchCamera

切换前后摄像头。

```
void switchCamera(TUICommonDefine.Camera camera);
```

参数如下表所示：

参数	类型	含义
camera	TUICommonDefine.Camera	前置/后置 摄像头

openMicrophone

打开麦克风。

```
void openMicrophone(TUICommonDefine.Callback callback);
```

closeMicrophone

关闭麦克风。

```
void closeMicrophone();
```

selectAudioPlaybackDevice

选择音频播放设备。
目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。

```
void selectAudioPlaybackDevice(TUICommonDefine.AudioPlaybackDevice device);
```

参数如下表所示：

参数	类型	含义
device	TUICommonDefine.AudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。
用户昵称不能超过500字节，用户头像必须是 URL 格式。

```
void setSelfInfo(String nickname, String avatar, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
nickname	String	用户昵称
avatar	String	用户头像（格式为 URL）

enableMultiDeviceAbility

开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）。

```
void enableMultiDeviceAbility(boolean enable, TUICommonDefine.Callback callback);
```

setVideoRenderParams

设置用户视频画面的渲染模式。

```
void setVideoRenderParams(String userId, TUICommonDefine.VideoRenderParams params,
TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
params	TUICommonDefine.VideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。
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。

```
void setVideoEncoderParams(TUICommonDefine.VideoEncoderParams params, TUICommonDefine.Callback
callback);
```

参数如下表所示：

参数	类型	含义

params	TUICommonDefine.VideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高比

getTRTCCloudInstance

高级接口。
用于接入第三方美颜等。

```
TRTCCloud getTRTCCloudInstance();
```

setBeautyLevel

设置美颜级别，支持关闭默认美颜。

```
void setBeautyLevel(float level, TUICommonDefine.Callback callback);
```

参数如下表所示：

参数	类型	含义
level	float	美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显

TUICallObserver

最近更新时间：2023-12-29 17:31:22

TUICallObserver API 简介

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

回调事件详情

onError

错误回调。

> **说明**
> 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。

```
void onError(int code, String msg);
```

参数如下表所示：

参数	类型	含义
code	int	错误码
msg	String	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

```
void onCallReceived(String callerId, List<String> calleeIdList, String groupId,
                    TUICallDefine.MediaType callMediaType, String userData);
```

参数如下表所示：

参数	类型	含义
callerId	String	主叫 ID（邀请方）
calleeIdList	List	被叫 ID 列表（被邀请方）
groupId	String	群组通话 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，比如视频通话、语音通话
userData	String	用户增加的扩展字段，详见 TUICallDefine.CallParams

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

- 主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）。
- 被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）。
- 异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）。

```
void onCallCancelled(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

```
void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole);
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallDefine.Role	角色，枚举类型：主叫、被叫

onCallEnd

表示通话挂断，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole, long totalTime);
```

参数如下表所示：

参数	类型	含义
roomId	TUICommonDefine.RoomId	此次通话的音视频房间 ID
callMediaType	TUICallDefine.MediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallDefine.Role	角色，枚举类型：主叫、被叫
totalTime	long	此次通话的时长，单位：秒

> **注意**
> 客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用REST API来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。

```
void onCallMediaTypeChanged(TUICallDefine.MediaType oldCallMediaType,TUICallDefine.MediaType
newCallMediaType);
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallDefine.MediaType	旧的通话类型

newCallMediaType	TUICallDefine.MediaType	新的通话类型

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。

```
void onUserReject(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。

```
void onUserNoResponse(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	无响应用户的 ID

onUserLineBusy

通话忙线回调。

```
void onUserLineBusy(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID

onUserJoin

有用户进入此次通话的回调。

```
void onUserJoin(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	加入当前通话的用户 ID

onUserLeave

有用户离开此次通话的回调。

版权所有：腾讯云计算（北京）有限责任公司

```
void onUserLeave(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	离开当前通话的用户 ID

onUserVideoAvailable

用户是否开启视频上行回调。

```
void onUserVideoAvailable(String userId, boolean isVideoAvailable);
```

参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID
isVideoAvailable	boolean	用户视频是否可用

onUserAudioAvailable

用户是否开启音频上行回调。

```
void onUserAudioAvailable(String userId, boolean isAudioAvailable);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID
isAudioAvailable	boolean	用户音频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

```
void onUserVoiceVolumeChanged(Map<String, Integer> volumeMap);
```

参数如下表所示：

参数	类型	含义
volumeMap	Map<String, Integer>	音量表，根据每个 userId 可以获取对应用户的音量大小，音量最小值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。

```
```

```
void onUserNetworkQualityChanged(List<TUICallDefine.NetworkQualityInfo> networkQualityList);
```

参数如下表所示：

参数	类型	含义
networkQualityList	List	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。

```
void onKickedOffline();
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。

```
void onUserSigExpired();
```

类型定义

最近更新时间：2023-09-04 16:59:02

常用结构

TUICallDefine

类型	描述
CallParams	通话参数
OfflinePushInfo	离线推送厂商配置信息

TUICommonDefine

类型	描述
RoomId	通话中音视频房间 ID
NetworkQualityInfo	网络质量
VideoRenderParams	视频画面的渲染参数
VideoEncoderParams	视频编码参数

枚举定义

TUICallDefine

类型	说明
MediaType	通话类型
Role	通话角色
Status	通话状态
Scene	通话场景
IOSOfflinePushType	iOS 离线推送类型

TUICommonDefine

类型	说明
AudioPlaybackDevice	音频设备
Camera	摄像头类型
NetworkQuality	网络质量
FillMode	视频画面填充模式
Rotation	视频画面旋转方向
ResolutionMode	视频宽高比模式

版权所有：腾讯云计算（北京）有限责任公司

Resolution	视频分辨率

CallParams

通话参数

类型	类型	描述
roomId	TUICommonDefine.RoomId	通话中音视频房间ID
offlinePushInfo	OfflinePushInfo	厂商离线推送配置信息
timeout	int	通话超时时间，默认：30s，单位：秒
userData	String	发起通话时自定义扩展字段，被叫端在 onCallReceived 中回调

OfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

类型	类型	描述
title	String	离线推送展示通知栏标题
desc	String	离线推送展示通知栏内容
ignoreIOSBadge	boolean	离线推送忽略 badge 计数（仅对 iOS 生效），如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加。
iOSSound	String	离线推送声音设置（仅对 iOS 生效）。当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀）设置给 iOSSound
androidSound	String	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声。另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	int	VIVO 推送消息分类 (待废弃接口，VIVO 推送服务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	String	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	String	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	String	华为推送消息分类，详见：华为消息分类标准

isDisablePush	boolean	是否关闭推送（默认开启推送）
iOSPushType	IOSOfflinePushType	iOS 离线推送类型，默认：APNs

RoomId

通话中音视频房间ID

- intRoomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 intRoomId 需要填写为 0。若两者都填，SDK 将优先选用 intRoomId。
- 不要混用 intRoomId 和 strRoomId，因为它们之间是不互通的，比如数字123和字符串123是两个完全不同的房间。

类型	类型	描述
intRoomId	int	数字房间号 **取值范围** 1 - 2147483647(2^31-1)
strRoomId	String	字符串房间号 **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、!、#、$、%、&、(、)、+、-、:、;、<、=、.、>、?、@、[、]、^、_、{、}、|、~、,。

> ⚠ **注意：**
> 字符串房间号目前只支持 Android、iOS、Flutter、Uniapp 平台，Web、小程序等其他平台后续支持，敬请期待！

NetworkQualityInfo

用户网络质量

类型	类型	描述
userId	String	用户 ID
quality	NetworkQuality	网络质量

VideoRenderParams

视频画面的渲染参数

类型	类型	描述
fillMode	VideoRenderParams.FillMode	视频画面填充模式
rotation	VideoRenderParams.Rotation	视频画面旋转方向

VideoEncoderParams

视频编码参数

类型	类型	描述

resolution	VideoEncoderParams.Resolution	视频分辨率
resolutionMode	VideoEncoderParams.ResolutionMode	视频宽高比

MediaType

通话类型

类型	取值	描述
Unknown	0	未知类型
Audio	1	语音通话
Video	2	视频通话

Role

通话角色

类型	取值	描述
None	0	未知类型
Caller	1	主叫（邀请方）
Called	2	被叫（被邀请方）

Status

通话状态

类型	取值	描述
None	0	未定义
Waiting	1	通话等待中
Accept	2	通话已接听

Scene

通话场景

类型	取值	描述
GROUP_CALL	0	群组通话
MULTI_CALL	1	匿名群组通话（暂不支持，敬请期待）
SINGLE_CALL	2	单人通话

IOSOfflinePushType

iOS离线推送类型

类型	取值	描述

版权所有：腾讯云计算（北京）有限责任公司

APNs	0	普通推送
VoIP	1	VoIP 推送

AudioPlaybackDevice

音频设备

类型	取值	描述
Earpiece	0	听筒
Speakerphone	1	扬声器

Camera

前置/后置 摄像头

类型	取值	描述
Front	0	前置摄像头
Back	1	后置摄像头

NetworkQuality

网络质量

类型	取值	描述
Unknown	0	未定义
Excellent	1	当前网络非常好
Good	2	当前网络比较好
Poor	3	当前网络一般
Bad	4	当前网络较差
Vbad	5	当前网络很差
Down	6	当前网络不满足通话的最低要求

FillMode

视频画面填充模式

类型	取值	描述
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。
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。

Rotation

视频画面旋转方向

类型	取值	描述
Rotation_0	0	不旋转
Rotation_90	1	顺时针旋转90度
Rotation_180	2	顺时针旋转180度
Rotation_270	3	顺时针旋转270度

ResolutionMode

视频宽高比模式

类型	取值	描述
Landscape	0	横屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360。
Portrait	1	竖屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640。

Resolution

视频分辨率

类型	取值	描述
Resolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps
Resolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
Resolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
Resolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
Resolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
Resolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

版权所有：腾讯云计算（北京）有限责任公司

iOS
API 概览

最近更新时间：2023-05-24 14:46:47

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的头像、昵称
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话

版权所有：腾讯云计算（北京）有限责任公司

ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等

TUICallObserver

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话的媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调

版权所有：腾讯云计算（北京）有限责任公司

onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

关键类型定义

API	描述
TUICallMediaType	通话的媒体类型，枚举类型：视频通话、语音通话
TUICallRole	通话的角色，枚举类型：主叫、被叫
TUICallStatus	通话的状态，枚举类型：空闲、待接听、接听中
TUIRoomId	音视频房间 ID，支持数字、字符串两种类型
TUICamera	摄像头 ID 参数，枚举类型：前摄、后摄
TUIAudioPlaybackDevice	声音的播放设备，枚举类型：扬声器、听筒
TUINetworkQuality	当前的网络质量信息

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2023-12-29 17:31:22

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入TUICallKit 。

API 概览

API	描述
createInstance	创建 TUICallKit 实例（单例模式）
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
call	发起 1v1 通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
groupCall	发起群组通话
groupCall	发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

API 详情

createInstance

创建 TUICallKit 的单例。

Objective-C

```
- (instancetype)createInstance;
```

Swift

```
public static func createInstance() -> TUICallKit
```

setSelfInfo

设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是 URL 格式。

Objective-C

```
- (void)setSelfInfo:(NSString * _Nullable)nickname avatar:(NSString * _Nullable)avatar succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

Swift

```
public func setSelfInfo(nickname: String, avatar: String, succ:@escaping TUICallSucc, fail: @escaping TUICallFail)
```

参数	类型	含义
nickname	NSString	用户昵称
avatar	NSString	用户头像（格式为 URL）
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

call

拨打电话（1v1通话）

Objective-C

```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType;
```

Swift

```
public func call(userId: String, callMediaType: TUICallMediaType)
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

call

拨打电话（1v1通话），支持自定义房间号、通话邀请超时时间，离线推送内容等。

Objective-C

```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType params:(TUICallParams *)params succ:(TUICallSucc __nullable)succ fail:(TUICallFail __nullable)fail;
```

Swift

```
public func call(userId: String, callMediaType: TUICallMediaType, params: TUICallParams,
                 succ: @escaping TUICallSucc, fail: @escaping TUICallFail)
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

groupCall

发起群组通话

> ⚠ **注意：**
> 使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Objective-C

```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray<NSString *> *)userIdList callMediaType:
(TUICallMediaType)callMediaType;
```

Swift

```
public func groupCall(groupId: String, userIdList: [String], callMediaType: TUICallMediaType)
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

groupCall

发起群组通话，支持自定义房间号、通话邀请超时时间，离线推送内容等。

> ⚠ **注意：**
> 使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

Objective-C

```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray<NSString *> *)userIdList callMediaType:(TUICallMediaType)callMediaType params:(TUICallParams *)params succ:(TUICallSucc __nullable)succ fail:(TUICallFail __nullable)fail;
```

Swift

```
public func groupCall(groupId: String, userIdList: [String], callMediaType: TUICallMediaType, params: TUICallParams, succ: @escaping TUICallSucc, fail: @escaping TUICallFail)
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话扩展参数，例如：房间号、通话邀请超时时间，离线推送自定义内容等
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

joinInGroupCall

发起群组通话

> ⚠ **注意：**
> **加入群组中已有的音视频通话前，需要提前创建或加入 IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。**
> **群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。**

Objective-C

```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaType:(TUICallMediaType)callMediaType;
```

Swift

```
public func joinInGroupCall(roomId: TUIRoomId, groupId: String, callMediaType: TUICallMediaType)
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
groupId	NSString	此次群组通话的群 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

setCallingBell

设置自定义来电铃音。
这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

- 铃声设置后与设备绑定，更换用户，铃声依旧会生效。
- 如需恢复默认铃声， `filePath` 传空即可。

Objective-C

```
- (void)setCallingBell:(NSString *)filePath;
```

Swift

```
public func setCallingBell(filePath: String)
```

enableMuteMode

开启后，收到通话请求，不会播放来电铃声。

Objective-C

```
- (void)enableMuteMode:(BOOL)enable;
```

Swift

```
public func enableMuteMode(enable: Bool)
```

enableFloatWindow

开启/关闭悬浮窗功能。
默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

Objective-C

```
- (void)enableFloatWindow:(BOOL)enable;
```

Swift

```
public func enableFloatWindow(enable: Bool)
```

版权所有：腾讯云计算（北京）有限责任公司

TUICallEngine

最近更新时间：2023-09-12 17:36:41

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
init	完成音视频通话基础能力的鉴权
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的昵称、头像
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）

版权所有：腾讯云计算（北京）有限责任公司

setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
getTRTCCloudInstance	高级接口，用于接入第三方美颜等
setBeautyLevel	设置美颜级别，支持关闭默认美颜

API 详情

createInstance

创建 TUICallEngine 的单例。

```
- (TUICallEngine *)createInstance;
```

destroyInstance

销毁 TUICallEngine 的单例。

```
- (void)destroyInstance;
```

init

初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。

```
- (void)init:(NSString *)sdkAppID userId:(NSString *)userId userSig:(NSString *)userSig succ:
(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数如下表所示：

参数	类型	含义
sdkAppID	NSString	您可以在**实时音视频控制台** > **应用管理** > **应用信息**中查看 SDKAppID
userId	NSString	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）
userSig	NSString	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

addObserver

添加回调接口，您可以通过这个接听，监听 TUICallObserver 相关的事件回调。

```
- (void)addObserver:(id<TUICallObserver>)observer;
```

removeObserver

移除回调接口。

```
- (void)removeObserver:(id<TUICallObserver>)observer;
```

call

拨打电话（1v1通话）。

> ⚠ **注意：**
> 字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！

```
- (void)call:(NSString *)userId callMediaType:(TUICallMediaType)callMediaType params:(TUICallParams *)params succ:(TUICallSucc)succ fail:(TUICallFail)fail
```

参数如下表所示：

参数	类型	含义
userId	NSString	目标用户的 userId
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

groupCall

发起群组通话。

> ⚠ **注意：**
> 使用群组通话前需要创建 IM 群组，如果已经创建，请忽略；
> 字符串房间号目前只支持 Android 和 iOS 平台，Web、小程序、Flutter、Uniapp 等其他平台后续支持，敬请期待！

```
- (void)groupCall:(NSString *)groupId userIdList:(NSArray <NSString *> *)userIdList callMediaType:(TUICallMediaType)callMediaType params:(TUICallParams *)params succ:(TUICallSucc)succ fail:(TUICallFail)fail
```

参数	类型	含义
groupId	NSString	此次群组通话的群 ID
userIdList	NSArray	目标用户的 userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

版权所有：腾讯云计算（北京）有限责任公司

succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

accept

接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。

```
- (void)accept:(TUICallSucc)succ fail:(TUICallFail)fail;
```

reject

拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。

```
- (void)reject:(TUICallSucc)succ fail:(TUICallFail)fail;
```

ignore

忽略当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数忽略来电，此时主叫会收到 `onUserLineBusy` 的回调。
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。

```
- (void)ignore:(TUICallSucc)succ fail:(TUICallFail)fail;
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

```
- (void)hangup:(TUICallSucc)succ fail:(TUICallFail)fail;
```

inviteUser

邀请用户加入此次群组通话。
使用场景：一个群组通话中的用户主动邀请其他人时使用。

```
- (void)inviteUser:(NSArray<NSString *> *)userIdList params:(TUICallParams *)params succ:(void(^)
(NSArray <NSString *> *userIdList))succ fail:(TUICallFail)fail
```

参数	类型	含义
userIdList	NSArray	目标用户的 userId 列表
params	TUICallParams	通话参数扩展字段，例如：离线推送自定义内容
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

joinInGroupCall

主动加入此次群组通话。
使用场景：群组内用户主动加入此次群组通话使用。

```
```

版权所有：腾讯云计算（北京）有限责任公司

```
- (void)joinInGroupCall:(TUIRoomId *)roomId groupId:(NSString *)groupId callMediaType:
(TUICallMediaType)callMediaType succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
groupId	NSString	此次群组通话的群 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

switchCallMediaType

切换视频通话到语音通话。

```
- (void)switchCallMediaType:(TUICallMediaType)newType;
```

参数	类型	含义
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

startRemoteView

设置显示视频画面的 View 对象

```
- (void)startRemoteView:(NSString *)userId videoView:(TUIVideoView *)videoView onPlaying:(void(^)
(NSString *userId))onPlaying onLoading:(void(^)(NSString *userId))onLoading onError:(void(^)(NSString
*userId, int code, NSString *errMsg))onError;
```

参数	类型	含义
userId	NSString	目标用户的 userId
videoView	TUIVideoView	待渲染的视图

stopRemoteview

停止订阅远端用户的视频数据。

```
- (void)stopRemoteView:(NSString *)userId;
```

参数	类型	含义
userId	NSString	目标用户的 userId

openCamera

开启摄像头。

```
- (void)openCamera:(TUICamera)camera videoView:(TUIVideoView *)videoView succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头
videoView	TUIVideoView	待渲染的视图
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

closeCamera

关闭摄像头。

```
- (void)closeCamera;
```

switchCamera

切换前后摄像头。

```
- (void)switchCamera:(TUICamera)camera;
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头

openMicrophone

打开麦克风。

```
- (void)openMicrophone:(TUICallSucc)succ fail:(TUICallFail)fail;
```

closeMicrophone

关闭麦克风。

```
- (void)closeMicrophone;
```

selectAudioPlaybackDevice

选择音频播放设备。
目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。

```
- (void)selectAudioPlaybackDevice:(TUIAudioPlaybackDevice)device;
```

参数	类型	含义

device	TUIAudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。
用户昵称不能超过500字节，用户头像必须是 URL 格式。

```
- (void)setSelfInfo:(NSString *_Nullable)nickname avatar:(NSString *_Nullable)avatar succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数	类型	含义
nickname	NSString	用户昵称
avatar	NSString	用户头像（格式为 URL）
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

enableMultiDeviceAbility

开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）

```
- (void)enableMultiDeviceAbility:(BOOL)enable succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

setVideoRenderParams

设置用户视频画面的渲染模式。

```
- (void)setVideoRenderParams:(NSString *)userId params:(TUIVideoRenderParams *)params succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数	类型	含义
userId	NSString	目标用户的 userId
params	TUIVideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

setVideoEncoderParams

设置视频编码的编码参数。
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。

```
- (void)setVideoEncoderParams:(TUIVideoEncoderParams *)params succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
params	TUIVideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高比
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

getTRTCCloudInstance

高级接口。
用于接入第三方美颜等。

```
- (TRTCCloud *)getTRTCCloudInstance;
```

setBeautyLevel

设置美颜级别，支持关闭默认美颜。

```
- (void)setBeautyLevel:(CGFloat)level succ:(TUICallSucc)succ fail:(TUICallFail)fail;
```

参数如下表所示：

参数	类型	含义
level	CGFloat	美颜级别，美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显。
succ	TUICallSucc	成功回调
fail	TUICallFail	失败回调

TUICallObserver

最近更新时间：2023-12-29 17:31:22

TUICallObserver API 简介

TUICallObserver 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

回调事件详情

onError

错误回调。

> **说明**
> 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。

```
- (void)onError:(int)code message:(NSString * _Nullable)message;
```

参数如下表所示：

参数	类型	含义
code	int	错误码
message	NSString	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

```
- (void)onCallReceived:(NSString *)callerId calleeIdList:(NSArray<NSString *> *)calleeIdList groupId:(NSString * _Nullable)groupId callMediaType:(TUICallMediaType)callMediaType userData:(NSString * _Nullable)userData
```

参数如下表所示：

参数	类型	含义
callerId	NSString	主叫 ID（邀请方）
calleeIdList	NSArray	被叫 ID 列表（被邀请方）
groupId	NSString	群组通话 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
userData	NSString	用户增加的扩展字段，详见 TUICallParams

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

- 主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）。
- 被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）。
- 异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）。

```
- (void)onCallCancelled:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

```
- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole;
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
callMediaType	TUICallMediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallRole	角色，枚举类型：主叫、被叫

onCallEnd

表示通话挂断，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole totalTime:(float)totalTime;
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
callMediaType	TUICallMediaType	通话的媒体类型，视频通话、语音通话
callRole	TUICallRole	角色，枚举类型：主叫、被叫
totalTime	float	此次通话的时长，单位：秒

> **注意**
> 客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用 REST API 来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。

```
- (void)onCallMediaTypeChanged:(TUICallMediaType)oldCallMediaType newCallMediaType:
(TUICallMediaType)newCallMediaType;
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallMediaType	旧的通话类型

版权所有：腾讯云计算（北京）有限责任公司

newCallMediaType	TUICallMediaType	新的通话类型

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。

```
- (void)onUserReject:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。

```
- (void)onUserNoResponse:(NSString *)userId;
```

参数如下表所示：

参数	类型	含义
userId	NSString	无响应用户的 ID

onUserLineBusy

通话忙线回调。

```
- (void)onUserLineBusy:(NSString *)userId;
```

onUserJoin

有用户进入此次通话的回调。

```
- (void)onUserJoin:(NSString *)userId;
```

onUserLeave

有用户离开此次通话的回调。

```
- (void)onUserLeave:(NSString *)userId;
```

onUserAudioAvailable

用户是否开启音频上行回调。

```
- (void)onUserAudioAvailable:(NSString *)userId isAudioAvailable:(BOOL)isAudioAvailable;
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID
isAudioAvailable	BOOL	用户音频是否可用

onUserVideoAvailable

用户是否开启视频上行回调。

```
- (void)onUserVideoAvailable:(NSString *)userId isVideoAvailable:(BOOL)isVideoAvailable;
```

参数如下表所示：

参数	类型	含义
userId	NSString	通话用户 ID
isVideoAvailable	BOOL	用户视频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

```
- (void)onUserVoiceVolumeChanged:(NSDictionary <NSString *, NSNumber *> *)volumeMap;
```

参数如下表所示：

参数	类型	含义
volumeMap	NSDictionary <NSString *, NSNumber *>	音量表，根据每个 userid 可以获取对应的音量大小，音量最小值为0，音量最大值为100
volumeMap	NSDictionary	音量表，根据每个 userId 可以获取对应的音量大小，音量最小值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。

```
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQualityList;
```

参数如下表所示：

参数	类型	含义
networkQualityList	NSArray	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。

```
- (void)onKickedOffline;
```

版权所有：腾讯云计算（北京）有限责任公司

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。

```
- (void)onUserSigExpired;
```

类型定义

最近更新时间：2023-09-11 17:51:52

常用结构

TUICallDefine

类型	描述
TUICallParams	通话参数
TUIOfflinePushInfo	离线推送厂商配置信息

TUICommonDefine

类型	描述
TUIRoomId	通话中音视频房间 ID
TUIVideoRenderParams	视频画面的渲染参数
TUIVideoEncoderParams	视频编码参数

枚举定义

TUICallDefine

类型	说明
TUICallMediaType	通话类型
TUICallRole	通话角色
TUICallStatus	通话状态
TUICallScene	通话场景
TUICallIOSOfflinePushType	iOS 离线推送类型

TUICommonDefine

类型	说明
TUIAudioPlaybackDevice	音频设备
TUICamera	摄像头类型
TUINetworkQuality	网络质量
TUIVideoRenderParamsFillMode	视频画面填充模式
TUIVideoRenderParamsRotation	视频画面旋转方向
TUIVideoEncoderParamsResolutionMode	视频宽高比模式

版权所有：腾讯云计算（北京）有限责任公司

TUIVideoEncoderParamsResolution	视频分辨率

TUICallParams

通话参数

参数	类型	描述
offlinePushInfo	TUIOfflinePushInfo	厂商离线推送配置信息
timeout	int	通话超时时间，默认：30s，单位：秒
userData	NSString	发起通话时自定义扩展字段，被叫端在 onCallReceived 中回调

TUIOfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

参数	类型	描述
title	NSString	离线推送展示通知栏标题
desc	NSString	离线推送展示通知栏内容
ignoreIOSBadge	BOOL	离线推送忽略 badge 计数（仅对 iOS 生效），如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加
iOSSound	NSString	离线推送声音设置（仅对 iOS 生效）。当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀）设置给 iOSSound
androidSound	NSString	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声 另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	NSString	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	NSInteger	VIVO 推送消息分类 (待废弃接口，VIVO 推送服务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	NSString	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	NSString	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	NSString	华为推送消息分类，详见：华为消息分类标准
isDisablePush	BOOL	是否关闭推送（默认开启推送）

iOSPushType	TUICallIOSOfflinePushType	iOS 离线推送类型，默认：APNs

TUIRoomId

通话中音视频房间ID

- intRoomId 与 strRoomId 是互斥的，若您选用 strRoomId，则 intRoomId 需要填写为 0。若两者都填，SDK 将优先选用 intRoomId。
- 不要混用 intRoomId 和 strRoomId，因为它们之间是不互通的，比如数字 123 和字符串 123 两个完全不同的房间。

参数	类型	描述
intRoomId	UInt32	数字房间号，**取值范围**：1 – 2147483647(2^31-1)
strRoomId	NSString	字符串房间号，**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、!、#、$、%、&、(、)、+、-、:、;、<、=、.、>、?、@、[、]、^、_、{、}、|、~、,。

> **注意：**
> 字符串房间号目前只支持 Android、iOS、Flutter、Uniapp 平台，Web、小程序等其他平台后续支持，敬请期待！

TUIVideoRenderParams

视频画面的渲染参数

参数	类型	描述
fillMode	TUIVideoRenderParamsFillMode	视频画面填充模式
rotation	TUIVideoRenderParamsRotation	视频画面旋转方向

TUIVideoEncoderParams

视频编码参数

参数	类型	描述
resolution	TUIVideoEncoderParamsResolution	视频分辨率
resolutionMode	TUIVideoEncoderParamsResolutionMode	视频宽高比

TUICallMediaType

通话类型

类型	取值	描述
TUICallMediaTypeUnknown	0	未知类型

版权所有：腾讯云计算（北京）有限责任公司

TUICallMediaTypeAudio	1	语音通话
TUICallMediaTypeVideo	2	视频通话

TUICallRole

通话角色

类型	取值	描述
TUICallRoleNone	0	未知类型
TUICallRoleCall	1	主叫（邀请方）
TUICallRoleCalled	2	被叫（被邀请方）

TUICallStatus

通话状态

类型	取值	描述
TUICallStatusNone	0	未定义
TUICallStatusWaiting	1	通话等待中
TUICallStatusAccept	2	通话已接听

TUICallScene

通话场景

类型	取值	描述
TUICallSceneGroup	0	群组通话
TUICallSceneMulti	1	匿名群组通话（暂不支持，敬请期待）
TUICallSceneSingle	2	单人通话

TUICallIOSOfflinePushType

iOS离线推送类型

类型	取值	描述
TUICallIOSOfflinePushTypeAPNs	0	普通推送
TUICallIOSOfflinePushTypeVoIP	1	VoIP 推送

TUIAudioPlaybackDevice

音频设备

类型	取值	描述

版权所有：腾讯云计算（北京）有限责任公司

TUIAudioPlaybackDeviceSpeakerphone	0	扬声器
TUIAudioPlaybackDeviceEarpiece	1	听筒

TUICamera

前置/后置 摄像头

类型	取值	描述
TUICameraFront	0	前置摄像头
TUICameraBack	1	后置摄像头

TUINetworkQuality

网络质量

类型	取值	描述
TUINetworkQualityUnknown	0	未定义
TUINetworkQualityExcellent	1	当前网络非常好
TUINetworkQualityGood	2	当前网络比较好
TUINetworkQualityPoor	3	当前网络一般
TUINetworkQualityBad	4	当前网络较差
TUINetworkQualityVbad	5	当前网络很差
TUINetworkQualityDown	6	当前网络不满足通话的最低要求

TUIVideoRenderParamsFillMode

视频画面填充模式

类型	取值	描述
TUIVideoRenderParamsFillModeFill	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
TUIVideoRenderParamsFillModeFit	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

TUIVideoRenderParamsRotation

视频画面旋转方向

类型	取值	描述
TUIVideoRenderParamsRotation_0	0	不旋转
TUIVideoRenderParamsRotati	1	顺时针旋转90度

版权所有：腾讯云计算（北京）有限责任公司

on_90		
TUIVideoRenderParamsRotation_180	2	顺时针旋转180度
TUIVideoRenderParamsRotation_270	3	顺时针旋转270度

TUIVideoEncoderParamsResolutionMode

视频宽高比模式

类型	取值	描述
TUIVideoEncoderParamsResolutionModeLandscape	0	横屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360
TUIVideoEncoderParamsResolutionModePortrait	1	竖屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640

TUIVideoEncoderParamsResolution

视频分辨率

类型	取值	描述
TUIVideoEncoderParamsResolution_640_360	108	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps
TUIVideoEncoderParamsResolution_640_480	62	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
TUIVideoEncoderParamsResolution_960_540	110	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
TUIVideoEncoderParamsResolution_960_720	64	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
TUIVideoEncoderParamsResolution_1280_720	112	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
TUIVideoEncoderParamsResolution_1920_1080	114	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

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：2024-01-11 15:42:52

TUICallKit（含 UI 组件）

TUICallKit 是音视频通话组件的**含 UI 组件**，您可以通过该组件快速实现一个类微信的音视频通话场景。

- <TUICallKit/>：UI 通话组件主体。
- <TUICallKitMini/>：UI 通话悬浮窗，若开启最小化功能，则 <TUICallKitMini/> 必须被放置在页面中。**注意：v2.3.2+ 已废弃。**

TUICallKitServer 是通话实例，提供的 API 接口如下。

API	描述
init	初始化 TUICallKit。
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
setSelfInfo	设置用户昵称和头像
enableMuteMode	开启/关闭来电铃声
enableFloatWindow	开启/关闭悬浮窗功能
setLanguage	设置 TUICallKit 组件通话语言
destroyed	销毁 TUICallKit

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
on	监听事件
off	取消监听事件
login	登录接口
logout	登出接口
setSelfInfo	设置用户昵称和头像
call	发起 1v1 通话

版权所有：腾讯云计算（北京）有限责任公司

groupCall	群聊邀请通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
switchCallMediaType	音视频通话切换
startRemoteView	启动远端画面渲染
stopRemoteView	停止远端画面渲染
startLocalView	启动本地画面渲染，**注意：后续将废弃，建议使用 openCamera**
stopLocalView	停止本地画面渲染，**注意：后续将废弃，建议使用 closeCamera**
openCamera	开启摄像头
closeCamara	关闭摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
setVideoQuality	设置视频质量
getDeviceList	获取设备列表
switchDevice	切换摄像头或麦克风设备
enableAIVoice	开启/关闭 AI 降噪

事件类型定义

TUICallEvent 是 TUICallEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

EVENT	描述
TUICallEvent.ERROR	SDK 内部发生了错误
TUICallEvent.SDK_READY	SDK 进入 ready 状态时收到该回调
TUICallEvent.KICKED_OUT	重复登录，收到该回调说明被踢出房间
TUICallEvent.USER_ACCEPT	如果有用户接听，那么会收到此回调
TUICallEvent.USER_ENTER	如果有用户同意进入通话，那么会收到此回调
TUICallEvent.USER_LEAVE	如果有用户同意离开通话，那么会收到此回调
TUICallEvent.REJECT	用户拒绝通话
TUICallEvent.NO_RESP	邀请用户无应答
TUICallEvent.LINE_BUSY	邀请方忙线
TUICallEvent.CALLING_TIMEOUT	作为被邀请方会收到，收到该回调说明本次通话超时未应答
TUICallEvent.USER_VIDEO_AVAILABLE	远端用户开启/关闭了摄像头，会收到该回调

TUICallEvent.USER_AUDIO_AVAILABLE	远端用户开启/关闭了麦克风，会收到该回调
TUICallEvent.USER_VOICE_VOLUME	远端用户说话音量调整，会收到该回调
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	群聊更新邀请列表收到该回调
TUICallEvent.INVITED	被邀请进行通话
TUICallEvent.CALLING_CANCEL	作为被邀请方会收到，收到该回调说明本次通话被取消了
TUICallEvent.CALLING_END	收到该回调说明本次通话结束了
TUICallEvent.DEVICED_UPDATED	设备列表更新收到该回调
TUICallEvent.CALL_TYPE_CHANGED	通话类型切换收到该回调

文档链接

- TUICallEngine
- TUICallEvent
- TUICallEngine API 文档

TUICallKit

最近更新时间：2024-01-26 15:06:11

API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详见：快速接入 TUICallKit。

API 概览

- `<TUICallKit />`：TUICallKit UI 通话组件主体。
- `<TUICallKitMini/>`：TUICallKit UI 通话悬浮窗，当 `<TUICallKit :allowedMinimized="true"/>` 时，`<TUICallKitMini/>` 必须被放置在页面中。

> **注意：**
> - `<TUICallKitMini/>` UI 通话悬浮窗，**≥ v2.3.2 废弃，不再提供该组件**。
> - 由 `<TUICallKit />` UI 通话组件主体，实现悬浮窗样式，减少使用复杂度。

API	描述
init	初始化 TUICallKit。
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
setSelfInfo	设置用户昵称和头像
enableMuteMode	开启/关闭来电铃声
enableFloatWindow	开启/关闭悬浮窗功能
setLanguage	设置 TUICallKit 组件通话语言
destroyed	销毁 TUICallKit

TUICallKit 属性介绍

属性概览

属性	描述	类型	是否必填	默认值
allowedMinimized	是否允许悬浮窗	boolean	否	false
allowedFullScreen	是否允许通话界面全屏	boolean	否	true

videoDisplayMode	通话界面显示模式	VideoDisplayMode	否	VideoDisplayMode.COVER
videoResolution	通话分辨率	VideoResolution	否	VideoResolution.RESOLUTION_480P
beforeCalling	拨打电话前与收到通话邀请前会执行此函数	function(type, error)	否	-
afterCalling	结束通话后会执行此函数	function()	否	-
onMinimized	组件切换最小化状态时会执行此函数，STATUS 值说明 ⚠ 注意：React TUICallKit 组件暂不支持	function(oldStatus, newStatus)	否	-
kickedOut	⚠ 注意：≥ v2.3.2 支持，React TUICallKit 组件暂不支持 组件抛出的事件，当前登录用户被踢出登录时会触发该事件，通话也会自动结束。	function()	否	-
statusChanged	⚠ 注意：≥ v2.3.2 支持，React TUICallKit 组件暂不支持 组件抛出的事件，当通话状态发生变化时，会触发该事件，通话状态种类详见，STATUS 值说明	function({oldStatus, newStatus})	否	-
@kicked-out	⚠ 注意：≥ v2.3.2 废弃 组件抛出的事件，当前登录用户被踢出登录时会触发该事件，通话也会自动结束	function()	否	-
@status-changed	⚠ 注意：≥ v2.3.2 废弃	function({oldStatus, newStatus})	否	-

	组件抛出的事件，当通话状态发生变化时，会触发该事件，通话状态种类详见，STATUS 值说明			

示例代码

Vue

```
<template>
  <TUICallKit
   :allowedMinimized="true"
   :allowedFullScreen="true"
   :videoDisplayMode="VideoDisplayMode.CONTAIN"
   :videoResolution="VideoResolution.RESOLUTION_1080P"
   :beforeCalling="beforeCalling"
   :afterCalling="afterCalling"
   :onMinimized="onMinimized"
   :kickedOut="handleKickedOut"
   :statusChanged="handleStatusChanged"
 />
</template>
<script lang="ts" setup>
import { TUICallKit, TUICallKitServer, VideoDisplayMode, VideoResolution, STATUS } from
"@tencentcloud/call-uikit-vue"; // vue2 要换成 '@tencentcloud/call-uikit-vue2'

function beforeCalling(type: string, error: any) {
 console.log("[Callkit Demo] beforeCalling:", type, error);
}
function afterCalling() {
 console.log("[Callkit Demo] afterCalling");
}
function onMinimized(oldStatus: string, newStatus: string) {
 console.log("[Callkit Demo] onMinimized: " + oldStatus + " -> " + newStatus);
}
function kickedOut() {
 console.log("[Callkit Demo] kickedOut");
}
function statusChanged(args: { oldStatus: string; newStatus: string; }) {
 const { oldStatus, newStatus } = args;
 if (newStatus === STATUS.CALLING_C2C_VIDEO) {
  console.log(`[Callkit Demo] statusChanged: ${oldStatus} -> ${newStatus}`);
 }
}
</script>
```

React

```
import React, { useState, useMemo } from "react";
```

```
// @ts-ignore
import { TUICallKit, TUICallKitServer, TUIGlobal, VideoDisplayMode, VideoResolution } from
"@tencentcloud/call-uikit-react";

function App() {
  const handleBeforeCalling = (type: string, error: any) => {
    console.log("[Callkit Demo] beforeCalling:", type, error);
  }
  const handleAfterCalling = () => {
    console.log("[Callkit Demo] afterCalling");
  }
  return (
    <>
      <TUICallKit
        :allowedMinimized="true"
        :allowedFullScreen="true"
        :videoDisplayMode="VideoDisplayMode.CONTAIN"
        :videoResolution="VideoResolution.RESOLUTION_1080P"
        beforeCalling={handleBeforeCalling}
        afterCalling={handleAfterCalling}
      />
    </>
  );
}

export default App;
```

TUICallKitServer API 详情

init

初始化 TUICallKit，必须在 call, groupCall 之前进行。

```
try {
  await TUICallKitServer.init({ SDKAppID, userID, userSig });
  // await TUICallKitServer.init({ tim, SDKAppID, userID, userSig}); // 如果工程中已有 tim 实例，需在此处传入
  alert("[TUICallKit] Initialization succeeds.");
} catch (error: any) {
  alert(`[TUICallKit] Initialization failed. Reason: ${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
SDKAppID	Number	是	云通信应用的 SDKAppID
userID	String	是	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）
userSig	String	是	腾讯云设计的一种安全保护签名，获取方式请参考如何计算 UserSig
tim	Any	否	tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性

版权所有：腾讯云计算（北京）有限责任公司

call

拨打 1v1 通话。

```
try {
  await TUICallKitServer.call({
    userID: callUserID,
    type: TUICallType.VIDEO_CALL
  });
} catch (error: any) {
  alert(`[TUICallKit] Call failed. Reason: ${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
userID	String	是	目标用户的 userId
type	TUICallType	是	通话的媒体类型，参数值说明参考 TUICallType 通话类型 兼容旧版本：语音通话(type = 1)、视频通话(type = 2)
roomID	Number	否	数字房间号，范围 [1, 2147483647]
timeout	Number	否	通话的超时时间，0 为不超时，单位 s(秒)（选填）- 默认 30s 被叫需要在 30s 内处于登录状态，timeout 值才会有效，具体参考 timeout 字段设置无效
userData	String	否	扩展字段: 用于在邀请信令中增加扩展信息
offlinePushInfo	Object	否	自定义离线消息推送（选填）-- 需 tsignaling 版本 >= 0.8.0

groupCall

发起群组通话。

```
try {
  await TUICallKitServer.groupCall({
    userIDList: ['jack', 'tom'],
    groupID: "xxx",
    type: TUICallType.VIDEO_CALL
  });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the groupCall API. Reason:${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
userIDList	Array<String>	是	邀请列表成员列表

type	TUICallType	是	通话的媒体类型，参数值说明参考 TUICallType 通话类型 兼容旧版本：语音通话(type = 1)、视频通话(type = 2)
groupID	String	是	通话群组 ID，groupID 的创建可参考 chat-createGroup API
roomID	Number	否	数字房间号，范围 [1, 2147483647]
timeout	Number	否	通话的超时时间，0 为不超时，单位 s(秒)（选填）- 默认 30s 被叫需要在 30s 内处于登录状态，timeout 值才会有效，具体参考 timeout 字段设置无效
userData	String	否	扩展字段：用于在邀请信令中增加扩展信息
offlinePushInfo	Object	否	自定义离线消息推送（选填）（tsignaling 版本 >= 0.8.0）

setLanguage

- 设置 TUICallKit 组件通话语言。
- 默认语言为浏览器的语言。

```
TUICallKitServer.setLanguage("zh-cn"); // "en" | "zh-cn"
```

参数如下表所示：

参数	类型	是否必填	含义
lang	String	是	语言类型 `en` 或 `zh-cn`

setSelfInfo

> ⚠ 注意：≥ v2.2.0 支持。通话中使用该接口修改用户信息，UI 不会立即更新，需要等到下次通话才能看到变化。

设置当前用户昵称和头像。

```
try {
  await TUICallKitServer.setSelfInfo({ nickName: "xxx", avatar: "http://xxx" });
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setSelfInfo API. Reason: ${error}`);
}
```

参数如下表所示：

参数	类型	是否必填	含义
nickName	String	是	用户昵称
avatar	String	是	用户头像地址

setCallingBell

> ⚠ 注意：≥ v3.0.0 支持

- 自定义用户的来电铃音。
- 仅限传入本地 MP3 格式的文件地址，需要确保该文件目录是应用可以访问的。
- 铃声媒体文件必须放在 public 目录下引入。

```
try {
  await TUICallKitServer.setCallingBell(filePath?: string)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the setCallingBell API. Reason: ${error}`);
}
```

enableFloatWindow

> ⚠ **注意：** ⩾ v3.1.0 支持

开启/关闭悬浮窗功能。
默认为 false ，通话界面左上角的悬浮窗按钮隐藏，设置为 true 后显示。

```
try {
  await TUICallKitServer.enableFloatWindow(enable: Boolean)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the enableFloatWindow API. Reason: ${error}`);
}
```

enableMuteMode

> ⚠ **注意：** ⩾ v3.1.2 支持

开启/关闭来电铃声。
开启后，收到通话请求时，不会播放来电铃声。

```
try {
  await TUICallKitServer.enableMuteMode(enable: boolean)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${error}`);
}
```

joinInGroupCall

> ⚠ **注意：** ⩾ v3.1.2 支持

加入群组中已有的音视频通话。

> ⓘ **说明：**
> 加入群组中已有的音视频通话前，需要提前创建或加入IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理 ，或者您也可以直接使用 IM TUIKit，一站式集成聊天、通话等场景。

```
try {
  const params = {
```

```
    type: TUICallType.VIDEO_CALL,
    groupID: "xxx",
    roomID: 0,
  };
  await TUICallKitServer.joinInGroupCall(params);
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${error}`);
}
```

参数列表：

参数	类型	是否必填	含义
type	TUICallType	是	通话的媒体类型，例如视频通话、语音通话
groupID	string	是	此次群组通话的群 ID
roomID	number	是	此次通话的音视频房间 ID

destroyed

销毁 TUICallKit 实例。
该方法不会自动退出 `tim` **，需要手动退出** `tim.logout();` **。**

```
try {
  await TUICallKitServer.destroyed();
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the destroyed API. Reason: ${error}`);
}
```

TUICallKit 类型定义

videoDisplayMode

画面显示模式 `videoDisplayMode` **属性有三个值：**

- `VideoDisplayMode.CONTAIN`
- `VideoDisplayMode.COVER`
- `VideoDisplayMode.FILL` **，默认值是** `VideoDisplayMode.COVER` **。**

属性	值	描述
videoDisplayMode	VideoDisplayMode.CONTAIN	• 优先保证视频内容全部显示。 • 视频尺寸等比缩放，直至视频窗口的一边与视窗边框对齐。 • 如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗，缩放后的视频四周会有一圈黑边。
	VideoDisplayMode.COVER	• 优先保证视窗被填满。 • 视频尺寸等比缩放，直至整个视窗被视频填满。 • 如果视频长宽与显示窗口不同，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
	VideoDisplayMode.FILL	• 保证视窗被填满的同时保证视频内容全部显示，但是不保证视频尺寸比例不变。

		• 视频的宽高会被拉伸至和视窗尺寸一致。

videoResolution

分辨率 videoResolution 有三个值：

- VideoResolution.RESOLUTION_480P
- VideoResolution.RESOLUTION_720P
- VideoResolution.RESOLUTION_1080P，默认值是 VideoResolution.RESOLUTION_480P 。

分辨率说明：

视频 Profile	分辨率（宽 × 高）	帧率（fps）	码率（kbps）
480p	640 × 480	15	900
720p	1280 × 720	15	1500
1080p	1920 × 1080	15	2000

常见问题：

- iOS 13&14不支持编码高于 720P 的视频，建议在这两个系统版本限制最高采集 720P。参考 iOS Safari 已知问题 case 12。
- Firefox 不支持自定义视频帧率（默认为 30fps）。
- 受系统性能占用，摄像头采集能力和浏览器限制等因素的影响，视频分辨率、帧率、码率的实际值不一定能够完全匹配设定值，在这种情况下，浏览器会自动调整 Profile 尽可能匹配设定值。

STATUS

在 onMinimized、statusChanged、 @status-changed 属性的回调事件中，会收到通话状态。

属性值	描述
STATUS.IDLE	闲置状态
STATUS.BE_INVITED	收到通话邀请
STATUS.DIALING_C2C	正在 1v1 呼叫
STATUS.DIALING_GROUP	正在群组呼叫
STATUS.CALLING_C2C_AUDIO	正在 1v1 语音通话
STATUS.CALLING_C2C_VIDEO	正在 1v1 视频通话
STATUS.CALLING_GROUP_AUDIO	正在群组语音通话
STATUS.CALLING_GROUP_VIDEO	正在群组视频通话

TUICallType

TUICallType 类型	描述
TUICallType.AUDIO_CALL	语音通话
TUICallType.VIDEO_CALL	视频通话

版权所有：腾讯云计算（北京）有限责任公司

offlinePushInfo

参数	类型	是否必填	含义
offlinePushInfo.title	String	否	离线推送标题（选填）
offlinePushInfo.description	String	否	离线推送内容（选填）
offlinePushInfo.androidOPPOChannelID	String	否	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
offlinePushInfo.extension	String	否	离线推送透传内容（选填）（tsignaling 版本 >= 0.9.0）

TUICallEngine

最近更新时间：2023-12-19 14:25:02

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**。

API 概览

API	描述
createInstance	创建 TUICallEngine 实例（单例模式）
destroyInstance	销毁 TUICallEngine 实例（单例模式）
on	监听事件
off	取消监听事件
login	登录接口
logout	登出接口
setSelfInfo	设置用户昵称和头像
call	C2C邀请通话
groupCall	群聊邀请通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	当前通话类型切换
startRemoteView	启动远端画面渲染
stopRemoteView	停止远端画面渲染
startLocalView	启动本地画面渲染
stopLocalView	停止本地画面渲染
openCamera	开启摄像头
closeCamera	关闭摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
setVideoQuality	设置视频质量

版权所有：腾讯云计算（北京）有限责任公司

setVideoRenderParams	设置用户视频画面的显示模式
getDeviceList	获取设备列表
switchDevice	切换摄像头或麦克风设备
enableAIVoice	开启/关闭 AI 降噪
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（**尊享版套餐支持**）

版权所有：腾讯云计算（北京）有限责任公司

TUICallEvent

最近更新时间：2023-07-05 10:51:41

TUICallEvent API 简介

TUICallEvent API 是音视频通话组件的**事件接口**。

事件列表

EVENT	描述
TUICallEvent.ERROR	SDK 内部发生了错误
TUICallEvent.SDK_READY	SDK 进入 ready 状态时收到该事件
TUICallEvent.KICKED_OUT	重复登录，收到该事件说明被移出房间
TUICallEvent.USER_ACCEPT	如果有用户接听，那么会收到此事件
TUICallEvent.USER_ENTER	如果有用户同意进入通话，那么会收到此事件
TUICallEvent.USER_LEAVE	如果有用户同意离开通话，那么会收到此事件
TUICallEvent.REJECT	用户拒绝通话，会收到该事件
TUICallEvent.NO_RESP	邀请用户无应答，会收到该事件
TUICallEvent.LINE_BUSY	邀请方忙线，会收到该事件
TUICallEvent.USER_VIDEO_AVAILABLE	远端用户开启/关闭了摄像头, 会收到该事件
TUICallEvent.USER_AUDIO_AVAILABLE	远端用户开启/关闭了麦克风, 会收到该事件
TUICallEvent.USER_VOICE_VOLUME	远端用户说话音量调整, 会收到该事件
TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE	群聊更新邀请列表收到该事件
TUICallEvent.INVITED	被邀请进行通话，会收到该事件
TUICallEvent.ON_CALL_RECEIVED	通话请求的事件
TUICallEvent.CALLING_CANCEL	通话未建立时，通话的各端都会抛出该事件
TUICallEvent.ON_CALL_BEGIN	通话接通时抛出的事件
TUICallEvent.CALLING_END	收到该事件说明本次通话结束了
TUICallEvent.DEVICED_UPDATED	设备列表更新收到该事件
TUICallEvent.CALL_TYPE_CHANGED	通话类型切换收到该事件

ERROR

SDK 内部发生了错误，可以通过监听该事件，捕获底层 SDK 的内部错误。

```
let onError = function(error) {
```

版权所有：腾讯云计算（北京）有限责任公司

```
  console.log(error.code, error.msg);
};
tuiCallEngine.on(TUICallEvent.ERROR, onError);
```

参数如下表所示：

参数	类型	含义
code	int	错误码
msg	String	错误信息

SDK_READY

SDK 进入 ready 状态时收到该事件。

```
let onSDKReady = function(event) {
  console.log(event);
};
tuiCallEngine.on(TUICallEvent.SDK_READY, onSDKReady);
```

KICKED_OUT

重复登录，收到该事件说明被踢出房间。

```
let handleOnKickedOut = function(event) {
  console.log(event);
};
tuiCallEngine.on(TUICallEvent.KICKED_OUT, handleOnKickedOut);
```

USER_ACCEPT

如果有用户接听，其他用户都会收到该事件，userID 就是接听的用户。

1. 1v1 通话下：被叫接通，主叫会抛出该事件。
2. 群组通话下：A 呼叫 B、C，B 接通，A、C 都会抛出该事件，事件的 userID 都是 B。同理 C 接通，A、B 都会抛出该事件，事件的 userID 都是 C。

```
let handleUserAccept = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.USER_ACCEPT, handleUserAccept);
```

参数如下表所示：

参数	类型	含义
userID	String	接听的用户 ID

USER_ENTER

如果有用户同意进入通话，其他用户都会收到该事件，userID 就是同意进入通话的用户。

```
let handleUserEnter = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.USER_ENTER, handleUserEnter);
```

参数如下表所示：

参数	类型	含义
userID	String	进房的用户 ID

USER_LEAVE

有用户离开此次通话时，**通话中**的其他用户都会收到该事件，userID 就是离开通话的用户。

```
let handleUserLeave = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.USER_LEAVE, handleUserLeave);
```

参数如下表所示：

参数	类型	含义
userID	String	退房的用户 ID

REJECT

通话被拒绝事件

1. 在 1v1 通话中，只有主叫方会收到拒绝事件，userID 就是被叫用户。
2. 在群组通话中，所有被邀请者都可以收到该事件，userID 就是拒绝通话的用户。

```
let handleInviteeReject = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.REJECT, handleInviteeReject);
```

参数如下表所示：

参数	类型	含义
userID	String	拒绝通话的用户 ID

NO_RESP

邀请用户无应答事件。

- 在 1v1 通话中，只有发起方会收到无人应答的事件。例如 A 邀请 B，B 不应答，A 可以收到该事件。
- 在群组通话中，所有被邀请人均能收到该事件。例如 A 邀请 B、C 进入通话，B 不应答，A、C 均能收到该事件。

```
let handleNoResponse = function(event) {
console.log(event.sponsor, event.userIDList);
};
```

版权所有：腾讯云计算（北京）有限责任公司

```
tuiCallEngine.on(TUICallEvent.NO_RESP, handleNoResponse);
```

参数如下表所示：

参数	类型	含义
sponsor	String	发起方的用户 ID
userIDList	Array<String>	触发超时无响应的用户列表

LINE_BUSY

通话忙线事件。

```
let handleLineBusy = function(event) {
  console.log(event)
};
tuiCallEngine.on(TUICallEvent.LINE_BUSY, handleLineBusy);
```

参数如下表所示：

参数	类型	含义
userID	String	忙线用户 ID

USER_VIDEO_AVAILABLE

远端用户开启/关闭了摄像头，会收到该事件。

```
let handleUserVideoChange = function(event) {
  console.log(event.userID, event.isVideoAvailable);
};
tuiCallEngine.on(TUICallEvent.USER_VIDEO_AVAILABLE, handleUserVideoChange);
```

参数如下表所示：

参数	类型	含义
userID	String	远端用户 ID
isVideoAvailable	Boolean	true 远端用户打开摄像头；false 远端用户关闭摄像头

USER_AUDIO_AVAILABLE

远端用户开启/关闭了麦克风，会收到该事件。

```
let handleUserAudioChange = function(event) {
  console.log(event.userID, event.isAudioAvailable);
};
tuiCallEngine.on(TUICallEvent.USER_AUDIO_AVAILABLE, handleUserAudioChange);
```

参数如下表所示：

参数	类型	含义
userID	String	远端用户 ID
isAudioAvailable	Boolean	true 远端用户打开麦克风；false 远端用户关闭麦克风

USER_VOICE_VOLUME

远端用户说话音量调整，会收到该事件。

```
let handleUserVoiceVolumeChange = function(event) {
  console.log(event.volumeMap);
};
tuiCallEngine.on(TUICallEvent.USER_VOICE_VOLUME, handleUserVoiceVolumeChange);
```

参数如下表所示：

参数	类型	含义
volumeMap	Array<Object>	音量表，根据每个 userid 可以获取对应的音量大小，音量最小值0，音量最大值100

GROUP_CALL_INVITEE_LIST_UPDATE

群聊更新邀请列表收到该事件。

```
let handleGroupInviteeListUpdate = function(event) {
  console.log(event.userIDList);
};
tuiCallEngine.on(TUICallEvent.GROUP_CALL_INVITEE_LIST_UPDATE, handleGroupInviteeListUpdate);
```

参数如下表所示：

参数	类型	含义
userIDList	Array<String>	群组更新邀请列表

INVITED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

> ⚠ **注意：**
> **计划后续版本废弃**

```
let handleNewInvitationReceived = function(event) {
  console.log(event.sponsor, event.userIDList, event.isFromGroup, event.inviteData, event.inviteDate,
event.inviteID);
};
tuiCallEngine.on(TUICallEvent.INVITED, handleNewInvitationReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
userIDList	Array<String>	同时还被邀请的人
isFromGroup	Boolean	是否是群组通话
inviteData	Object	呼叫数据
inviteID	String	邀请 ID，标识一次邀请
userData	String	扩展字段：用于在邀请信令中增加扩展信息
callId	String	本次通话的唯一 ID。**v1.4.6+ 版本支持**
roomID	Number	此次通话的音视频房间 ID。**v1.4.6+ 版本支持**
callMediaType	Number	通话的媒体类型，视频通话、语音通话。**v1.4.6+ 版本支持**
callRole	String	角色，枚举类型：主叫、被叫。**v1.4.6+ 版本支持**

ON_CALL_RECEIVED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

> ⚠ **注意：**
> **v1.4.6+ 版本支持**

```
let handleOnCallReceived = function(event) {
    console.log(event)
};
tuiCallEngine.on(TUICallEvent.ON_CALL_RECEIVED, handleOnCallReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
userIDList	Array<String>	同时还被邀请的人
isFromGroup	Boolean	是否是群组通话
inviteData	Object	呼叫数据
inviteID	String	邀请 ID，标识一次邀请
userData	String	扩展字段：用于在邀请信令中增加扩展信息
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫、被叫

版权所有：腾讯云计算（北京）有限责任公司

CALLING_CANCEL

如果通话未建立，都会抛出该事件。您可以通过监听该事件来实现类似未接来电、重置 UI 状态等显示逻辑。通话未建立的场景如下：

- 主叫取消：主叫抛出该事件，userID 为主叫；被叫也会抛出该事件，userID 为被叫；
- 被叫超时：主叫会同时抛出 NO_RESP 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫拒接：主叫会同时抛出 REJECT 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫忙线：主叫会同时抛出 LINE_BUSY 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

> ⚠ **注意：**
> **计划后续版本废弃**

```
let handleCallingCancel = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(TUICallEvent.CALLING_CANCEL, handleCallingCancel);
```

参数如下表所示：

参数	类型	含义
userID	String	通话取消的用户 ID
callId	String	本次通话的唯一 ID。**v1.4.6+ 版本支持**
roomID	Number	此次通话的音视频房间 ID。**v1.4.6+ 版本支持**
callMediaType	Number	通话的媒体类型，视频通话、语音通话。**v1.4.6+ 版本支持**
callRole	String	角色，枚举类型：主叫、被叫。**v1.4.6+ 版本支持**

ON_CALL_CANCELED

如果通话未建立，都会抛出该事件。您可以通过监听该事件来实现类似未接来电、重置 UI 状态等显示逻辑。通话未建立的场景如下：

- 主叫取消：主叫抛出该事件，userID 为主叫；被叫也会抛出该事件，userID 为被叫；
- 被叫超时：主叫会同时抛出 NO_RESP 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫拒接：主叫会同时抛出 REJECT 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫忙线：主叫会同时抛出 LINE_BUSY 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

> ⚠ **注意：**
> **v1.4.6+ 版本支持**

```
let handleOnCallCanceled = function(event) {
  console.log(event.userID);
};
```

版权所有：腾讯云计算（北京）有限责任公司

```
tuiCallEngine.on(TUICallEvent.ON_CALL_CANCELED, handleOnCallCanceled);
```

参数如下表所示：

参数	类型	含义
userID	String	通话取消的用户 ID
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫、被叫

ON_CALL_BEGIN

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

> ⚠ 注意：
> v1.4.6+ 版本支持

```
let handleOnCallBegin = function(event) {
    console.log(event)
};
tuiCallEngine.on(TUICallEvent.ON_CALL_BEGIN, handleOnCallBegin);
```

参数如下表所示：

参数	类型	含义
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，类型：主叫、被叫

CALLING_END

表示通话结束，主叫和被叫抛出该事件。您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
let handleCallingEnd = function(event) {
  console.log(event.userID, event.);
};
tuiCallEngine.on(TUICallEvent.CALLING_END, handleCallingEnd);
```

参数如下表所示：

参数	类型	含义

roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫（'inviter'）、被叫（'invitee'）、未知（''）
totalTime	Number	此次通话的时长，单位: 秒
userID	String	通话结束的 userID。
callId	String	本次通话的唯一 ID。**v1.4.6+ 版本支持**
callEnd	Number	此次通话的时长（**后续将废弃**），单位: 秒

DEVICED_UPDATED

设备列表更新收到该事件。

```
let handleDeviceUpdated = function({ microphoneList, cameraList, currentMicrophoneID, currentCameraID
}) {
  console.log(microphoneList, cameraList, currentMicrophoneID, currentCameraID)
};
tuiCallEngine.on(TUICallEvent.DEVICED_UPDATED, handleDeviceUpdated);
```

CALL_TYPE_CHANGED

通话类型切换收到该事件。

```
let handleCallTypeChanged = function({ oldCallType, newCallType }) {
  console.log(oldCallType, newCallType)
};
tuiCallEngine.on(TUICallEvent.CALL_TYPE_CHANGED, handleDeviceUpdated);
```

参数如下表所示：

参数	类型	含义
oldCallType	Number	旧通话类型
newCallType	Number	新通话类型

微信小程序
API 概览

最近更新时间：2023-11-21 15:31:31

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
init	初始化 TUICallKit
call	发起 1v1 通话
groupCall	发起群组通话
setSelfInfo	设置用户的头像、昵称
setCallingBell	设置自定义来电铃声
setLogLevel	设置日志级别
enableFloatWindow	开启/关闭悬浮窗功能
enableMuteMode	开启/关闭来电铃声
joinInGroupCall	加入群组中已有的音视频通话
destroyed	销毁 TUICallKit

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
createInstance	创建 TUICallEngine 实例（静态方法）
destroyInstance	销毁 TUICallEngine 实例（静态方法）
on	增加事件监听
off	取消事件监听
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话

版权所有：腾讯云计算（北京）有限责任公司

openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）
setSelfInfo	设置用户的头像、昵称

关键常量定义

常量	描述
EVENT	通话的事件表
CALL_STATUS	通话的状态 默认、呼叫中/被呼叫中 接通中
MEDIA_TYPE	通话的类型 音频 视频
AUDIO_PLAYBACK_DEVICE	声音的播放设备 扬声器、听筒

通话的事件表

事件名	说明
INVITED	被邀请通话
USER_ACCEPT	用户接听
USER_ENTER	用户进入通话
USER_LEAVE	用户离开通话
USER_UPDATE	用户更新
REJECT	用户拒绝通话
NO_RESP	用户无响应
LINE_BUSY	用户忙线
CALLING_CANCEL	通话被取消
CALL_END	通话结束
SDK_READY	SDK Ready 回调
KICKED_OUT	被踢下线
CALL_MODE	切换通话模式

通话的状态

CALL_STATUS	说明

IDLE	默认
CALLING	呼叫中/被呼叫中
CONNECTED	接通中

通话的类型

MEDIA_TYPE	说明
AUDIO	音频
VIDEO	视频

声音的播放设备

AUDIO_PLAYBACK_DEVICE	说明
EAR	听筒
SPEAKER	扬声器

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2024-01-26 15:06:11

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入TUICallKit 。

API 概览

API	描述
init	初始化 TUICallKit
call	发起 1v1 通话
groupCall	发起群组通话
setSelfInfo	设置用户的头像、昵称
setCallingBell	设置自定义来电铃声
setLogLevel	设置日志级别
enableFloatWindow	开启/关闭悬浮窗功能
enableMuteMode	开启/关闭来电铃声
joinInGroupCall	加入群组中已有的音视频通话
destroyed	销毁 TUICallKit

API 详情

init

初始化 TUICallKit。

```
init({
    sdkAppID: 0, // 替换为您自己账号下的 SDKAppId
    userID: 'jane', // 填写当前用的 userID
    userSig: 'xxxxxxxxxxx',
    type:1,
    tim: null, // 如果您不需要 TIM 实例，可忽略
})
```

参数如下表所示：

参数	类型	说明	是否必填
sdkAppID	Number	您可以在**实时音视频控制台** > 应用管理 > **应用信息**中查看 SDKAppID	是
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）	是

userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见：如何计算及使用 UserSig	是
tim	ChatSDK	TIM 实例	否

这里详细介绍一下 init 中的几个参数：

- **userSig**：使用步骤三中获取的 SecretKey 对 sdkAppID、userID 等信息进行加密，就可以得到 userSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务，更多信息请参见 如何计算及使用 UserSig 。
- **tim** 可以将外部的 tim 实例通过 init 透传给 callkit ，tim 参数适用于业务中已存在 TIM 实例，为保证 TIM 实例唯一性。

call

拨打电话（1v1 通话）。

```
call({
    userID: "jane",
    type: 1
})
```

参数如下表所示：

参数	类型	含义	是否必填
userID	String	被叫用户的 ID	是
type	Number	通话的媒体类型：1：语音通话　2：视频通话	是
roomID	Number	数字房间号, 范围 [1, 2147483647]	否
userData	String	扩展字段: 用于在邀请信令中增加扩展信息	否
timeout	Number	通话的超时时间，0 为不超时, 单位 s(秒)（选填） – 默认 30s 被叫需要在 30s 内处于登录状态，timeout 值才会有效，具体参考 timeout 字段设置无效	否
offlinePushInfo	Object	自定义离线消息推送（选填）	否

groupCall

发起群组通话。

```
groupCall({
    userIDList: ["jane", "mike", "tommy"],
    type: 1,
    groupID: "12345678"
})
```

参数如下表所示：

参数	类型	含义	是否必填
userIDList	Array<String>	目标用户的 userId 列表，示例：["jane", "mike", "tommy"]	是

版权所有：腾讯云计算（北京）有限责任公司

type	Number	通话的媒体类型：1：语音通话　2：视频通话	是
groupID	String	此次群组通话的群 ID	是
roomID	Number	数字房间号, 范围 [1, 2147483647]	否
userData	String	扩展字段: 用于在邀请信令中增加扩展信息	否
timeout	Number	通话的超时时间，0 为不超时, 单位 s(秒)（选填）– 默认 30s 被叫需要在 30s 内处于登录状态，timeout 值才会有效，具体参考 timeout 字段设置无效	否
offlinePushInfo	Object	自定义离线消息推送（选填）	否

setSelfInfo

设置用户头像、昵称。

> ⚠ **注意：通话中使用该接口修改用户信息，UI 不会立即更新，需要等到下次通话才能看到变化。**

> v1.4.4

```
setSelfInfo({ nickName: "xxx", avatar: "http://xxx" });
```

≤ v1.4.4

```
setSelfInfo('昵称', '头像地址');
```

参数如下表所示：

参数	类型	含义
nickName	String	设置昵称
avatar	String	头像地址

setCallingBell

设置自定义来电铃声。
这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

- 传入路径应为本地铃声文件的绝对地址。
- 如需恢复默认铃声，`filePath` 传空即可。
- 支持的铃声文件格式：

格式	iOS	Android

m4a	√	√
mp3	√	√
wav	√	√
aac	√	√

```
setCallingBell("filePath")
```

参数如下表所示：

参数	类型	含义
filePath	String	铃声地址

setLogLevel

设置日志级别，低于 level 的日志将不会输出。

```
setLogLevel(level)
```

参数如下表所示：

参数	值	含义
level	0	普通级别，日志量较多，接入时建议使用
	1	release 级别，SDK 输出关键信息，生产环境时建议使用
	2	告警级别，SDK 只输出告警和错误级别的日志
	3	错误级别，SDK 只输出错误级别的日志
	4	无日志级别，SDK 将不打印任何日志

enableFloatWindow

> ⚠ **注意：** ⩾ v3.1.0 支持

开启/关闭悬浮窗功能。
默认为 `false` ，通话界面左上角的悬浮窗按钮隐藏，设置为 `true` 后显示。

```
enableFloatWindow(enable?: Boolean)
```

enableMuteMode

> ⚠ **注意：** ⩾ v3.1.2 支持

开启/关闭来电铃声。
开启后，收到通话请求时，不会播放来电铃声。

```
try {
  await TUICallKitServer.enableMuteMode(enable: boolean)
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${error}`);
}
```

joinInGroupCall

> ⚠ **注意：** ≥v3.1.2 支持

加入群组中已有的音视频通话。

> **说明：**
> 加入群组中已有的音视频通话前，需要提前创建或加入IM 群组，并且群组中已有用户在通话中，如果已经创建，请忽略。
> 群组的创建详见： IM 群组管理，或者您也可以直接使用 IM TUIKIT，一站式集成聊天、通话等场景。

```
try {
  const params = {
    type: 2, // 视频通话
    groupID: "xxx",
    roomID: 0,
  };
  await TUICallKitServer.joinInGroupCall(params);
} catch (error: any) {
  alert(`[TUICallKit] Failed to call the enableMuteMode API. Reason: ${error}`);
}
```

参数列表：

参数	类型	是否必填	含义
type	number	是	通话的媒体类型：1：语音通话　2：视频通话
groupID	string	是	此次群组通话的群 ID
roomID	number	是	此次通话的音视频房间 ID

destroyed

销毁 TUICallKit。

```
destroyed()
```

offlinePushInfo

参数	类型	是否必填	含义
offlinePushInfo.title	String	否	离线推送标题（选填）
offlinePushInfo.description	String	否	离线推送内容（选填）

offlinePushInfo.androidOPPOChannelID	String	否	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID（选填）
offlinePushInfo.extension	String	否	离线推送透传内容（选填）（tsignaling 版本 ⩾ 0.9.0）

TUICallEngine

最近更新时间：2023-12-08 14:53:42

TUICallEngine API 简介

TUICallEngine 是无 UI 音视频通话 API 接口。如果 TUICallKit 的 UI 无法满足您的需求，您可以使用 TUICallEngine API 自己实现 UI 交互。
由于小程序开发的限制，需要先给 live-pusher 绑定事件，具体请参见 TUICallkit 。

API 概览

创建实例和事件回调

API	描述
createInstance	创建 TUICallEngine 实例（静态方法）
destroyInstance	销毁 TUICallEngine 实例（静态方法）
on	增加事件监听
off	取消事件监听

通话操作相关接口函数

API	描述
call	发起 1v1 通话
groupCall	发起群组通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
accept	接听通话
reject	拒绝通话
hangup	结束通话
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话

设备控制相关接口函数

API	描述
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风

版权所有：腾讯云计算（北京）有限责任公司

selectAudioPlaybackDevice	选择音频播放设备（听筒/扬声器）

其他接口函数

API	描述
setSelfInfo	设置用户的头像、昵称
setLogLevel	设置日志等级
setBeautyLevel	设置美颜级别，支持关闭默认美颜

API 详情

createInstance

创建 TUICallEngine 的单例。

```
import TUICallEngine from 'tuicall-engine-wx';
wx.$TUICallEngine = TUICallEngine.createInstance({
  tim,
  sdkAppID,
});
```

说明

参数	类型	说明	是否必传
sdkAppID	Number	SDKAppId 是腾讯云用于区分客户的唯一标识	是
tim	ChatSDK	TIM 实例	否

destroyInstance

销毁 TUICallEngine 的单例。

```
TUICallEngine.destroyInstance();
```

on

通过 on 方法可以监听 TUICallEngine 所有的事件。

```
import { EVENT } from 'tuicall-engine-wx';
let handleNewInvitationReceived = function(event) {
  // 收到邀请
};
wx.$TUICallEngine.on(EVENT.INVITED, handleNewInvitationReceived, this);
```

说明

参数	类型	说明	是否必传
eventName	EVENT	事件名	是

handler	Function	事件的回调函数	是
context	Any	当前执行上下文（this）	是

off

移除事件监听。

```
import { EVENT } from 'tuicall-engine-wx';
let handleNewInvitationReceived = function(event) {
    // 收到邀请
};
wx.$TUICallEngine.off(EVENT.INVITED, handleNewInvitationReceived);
```

说明

参数	类型	说明	是否必传
eventName	EVENT	事件名	是
handler	Function	事件的回调函数	是

login

登录接口。

```
const params = {
    userID: 'john', // 您的用户 ID
    userSig: 'xxxx', // 签名
};
let promise = wx.$TUICallEngine.login(params);
promise.then(() => {
    //success
}).catch(error => {
    console.warn('login error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）	是
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参考 如何计算UserSig 。	是

logout

登出接口。

```
let promise = wx.$TUICallEngine.logout();
promise.then(() => {
    // logout success
```

```
}).catch(error => {
  console.warn('logout error:', error);
});
```

call

拨打电话（1v1通话）。

> ⚠ **注意**
> **离线推送仅适用于终端（Android 或 iOS），Web 和 微信小程序不支持。**

```
import { MEDIA_TYPE } from 'tuicall-engine-wx';
let promise = wx.$TUICallEngine.call({
  userID: 'user1',
  type: MEDIA_TYPE.AUDIO,
});
promise.then(() => {
  // call success
}).catch(error => {
  console.warn('call error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
userID	String	被呼叫者的 userID	是
type	MEDIA_TYPE	通话的媒体类型 MEDIA_TYPE.AUDIO：语音通话 MEDIA_TYPE.VIDEO：视频通话	是
roomID	Number	数字房间号, 范围 [1, 2147483647]	否
userData	String	扩展字段: 用于在邀请信令中增加扩展信息	否

groupCall

发起群组通话。

> ⚠ **注意**
> **离线推送仅适用于终端（Android 或 iOS），Web 和 微信小程序不支持。**

```
import { MEDIA_TYPE } from 'tuicall-engine-wx';
let promise = wx.$TUICallEngine.groupCall({
  userIDList: ['user1', 'user2'],
  type: MEDIA_TYPE.AUDIO,
  groupID: '123345',
});
promise.then(() => {
  // groupCall success
}).catch(error => {
  console.warn('groupCall error:', error);
```

```
});
```

参数如下表所示：

参数	类型	含义	是否必传
userIDList	Array<String>	被呼叫者的 userID 列表	是
type	MEDIA_TYPE	通话的媒体类型 MEDIA_TYPE.AUDIO：语音通话 MEDIA_TYPE.VIDEO：视频通话	是
groupID	String	此次群组通话的群 ID	是
timeout	Number	超时时长，单位：秒	否
roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号	否
userData	String	扩展字段: 用于在邀请信令中增加扩展信息	否

inviteUser

邀请用户加入此次群组通话。
使用场景：一个群组通话中的用户主动邀请其他人时使用。

```
const userIDList = ['jack', 'john'];
const params = {
  userIDList
};
wx.$TUICallEngine.inviteUser(params).then(() => {
  // success
}).catch(error => {
  console.error('inviteUser error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
params	Object	邀请参数	是
params.userIDList	Array<String>	被叫用户的 userID 列表	是

joinInGroupCall

主动加入此次群组通话。
使用场景：群组内用户主动加入此次群组通话使用。

```
import { MEDIA_TYPE } from 'tuicall-engine-wx';
const params = {
  roomID: 123,
```

```
    type: MEDIA_TYPE.AUDIO,
    groupID: 111
};
wx.$TUICallEngine.joinInGroupCall(params).then(() => {
    // success
}).catch(error => {
    console.error('joinInGroupCall error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
params	Object	主动加入参数	是
roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号	是
groupID	String	此次群组通话的群 ID	是
type	MEDIA_TYPE	通话的媒体类型 MEDIA_TYPE.AUDIO：语音通话 MEDIA_TYPE.VIDEO：视频通话	是

accept

接受当前通话，当您作为被叫方收到 `EVENT.INVITED` 事件的回调时，可以调用该接口接听来电。

```
import { EVENT } from 'tuicall-engine-wx';
wx.$TUICallEngine.on(EVENT.INVITED, () => {
    wx.$TUICallEngine.accept().promise.then(() => {
        // accept success
    }).catch(error => {
        console.warn('accept error:', error);
    });
});
```

reject

拒绝当前通话，当您作为被叫收到 `EVENT.INVITED` 的回调时，可以调用该函数拒绝来电。

```
import { EVENT } from 'tuicall-engine-wx';
wx.$TUICallEngine.on(EVENT.INVITED, () => {
    wx.$TUICallEngine.reject().then(() => {
        // reject success
    }).catch(error => {
        console.warn('reject error:', error);
    });
});
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

版权所有：腾讯云计算（北京）有限责任公司

```
wx.$TUICallEngine.hangup().then(() => {
    // hangup success
}).catch(error => {
    console.warn('hangup error:', error);
});
```

switchCallMediaType

当前通话类型切换。

- 仅支持 1v1 通话过程中使用

```
import { MEDIA_TYPE } from 'tuicall-engine-wx';
wx.$TUICallEngine.switchCallMediaType(MEDIA_TYPE.VIDEO).then(() => {
  // switchCallMediaType success
}).catch(error => {
  console.warn('switchCallMediaType error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
type	MEDIA_TYPE	通话的媒体类型 MEDIA_TYPE.AUDIO：语音通话 MEDIA_TYPE.VIDEO：视频通话	是

openCamera

您可以调用该函数开启摄像头。

```
wx.$TUICallEngine.openCamera().then(() => {
    // openCamera success
}).catch(error => {
    console.warn('openCamera error:', error);
});
```

closeCamera

您可以调用该函数关闭摄像头，处于通话中的用户会收到回调。

```
wx.$TUICallEngine.closeCamera().then(() => {
    // closeCamera success
}).catch(error => {
    console.warn('closeCamera error:', error);
});
```

switchCamera

切换前后摄像头。

```
wx.$TUICallEngine.switchCamera().then(() => {
    // switchCamera success
```

版权所有：腾讯云计算（北京）有限责任公司

```
}).catch(error => {
    console.warn('openMicrophone error:', error);
});
```

openMicrophone

您可以调用该函数打开麦克风，处于通话中的用户会收到回调。

```
wx.$TUICallEngine.openMicrophone().then(() => {
    // openMicrophone success
}).catch(error => {
    console.warn('openMicrophone error:', error);
});
```

closeMicrophone

您可以调用该函数关闭麦克风，处于通话中的用户会收到回调。

```
wx.$TUICallEngine.closeMicrophone().then(() => {
    // closeMicrophone success
}).catch(error => {
    console.warn('closeMicrophone error:', error);
});
```

selectAudioPlaybackDevice

选择音频播放设备，目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。

```
wx.$TUICallEngine.selectAudioPlaybackDevice(AUDIO_PLAYBACK_DEVICE.EAR).then(() => {
    // selectAudioPlaybackDevice success
}).catch(error => {
    console.warn('selectAudioPlaybackDevice error:', error);
});
```

参数如下表所示：

参数	类型	含义	是否必传
type	AUDIO_PLAYBACK_DEVICE	扬声器：AUDIO_PLAYBACK_DEVICE.SPEAKER 听筒：AUDIO_PLAYBACK_DEVICE.EAR	是

setSelfInfo

设置用户头像、昵称。

```
wx.$TUICallEngine.setSelfInfo("昵称", "头像 URL").then(() => {
    // setSelfInfo success
}).catch(error => {
    console.warn('setSelfInfo error:', error);
});
```

参数如下表所示：

参数	类型	含义
nickName	String	设置昵称
avatar	String	头像地址

setLogLevel

设置日志级别，低于 level 的日志将不会输出。

```
wx.$TUICallEngine.setLogLevel(level)
```

参数如下表所示：

参数	值	含义
level	0	普通级别，日志量较多，接入时建议使用
	1	release 级别，SDK 输出关键信息，生产环境时建议使用
	2	告警级别，SDK 只输出告警和错误级别的日志
	3	错误级别，SDK 只输出错误级别的日志
	4	无日志级别，SDK 将不打印任何日志

setBeautyLevel

> ⚠ **注意：** ⩾ v2.1.0 支持

设置美颜级别，支持关闭默认美颜。

```
wx.$TUICallEngine.setBeautyLevel(beautyLevel: number);
```

参数如下表所示：

参数	类型	含义
beautyLevel	number	美颜级别，取值范围 0 - 9； 0 表示关闭，9 表示效果最明显

TUICallEvent

最近更新时间：2023-07-05 10:51:41

TUICallEvent API 简介

TUICallEvent API 是音视频通话组件的事件接口。

事件表

事件名	说明
INVITED	被邀请通话
ON_CALL_RECEIVED	通话请求的事件
USER_ACCEPT	用户接听
USER_ENTER	用户进入通话
USER_LEAVE	用户离开通话
USER_UPDATE	用户更新
REJECT	用户拒绝通话
NO_RESP	用户无响应
LINE_BUSY	用户忙线
CALLING_CANCEL	通话被取消
ON_CALL_CANCELED	通话未建立时，通话的各端都会抛出该事件
ON_CALL_BEGIN	通话接通时抛出的事件
CALL_END	通话结束
SDK_READY	SDK Ready 回调
KICKED_OUT	被踢下线
CALL_MODE	切换通话模式
ERROR	错误信息

INVITED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

> ⚠ **注意：**
> **计划后续废弃**

```
let handleNewInvitationReceived = function(event){
    console.log('被邀请通话')
  }
tuiCallEngine.on(EVENT.INVITED, this.handleNewInvitationReceived, this);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
inviteeList	Array<String>	同时还被邀请的人（**后续将废弃**）
userIDList	Array<String>	同时还被邀请的人
isFromGroup	Boolean	是否是群组通话
inviteData	Object	邀请数据
inviteID	String	邀请 ID，标识一次邀请
userData	String	扩展字段：用于在邀请信令中增加扩展信息
callId	String	本次通话的唯一 ID。**v1.5.1+ 版本支持**
roomID	Number	此次通话的音视频房间 ID。**v1.5.1+ 版本支持**
callMediaType	Number	通话的媒体类型，视频通话、语音通话。**v1.5.1+ 版本支持**
callRole	String	角色，枚举类型：主叫、被叫。**v1.5.1+ 版本支持**

ON_CALL_RECEIVED

收到一个新的来电请求事件，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

> ⚠ **注意：**
> **v1.5.1+ 版本支持**

```
let handleOnCallReceived = function(event) {
    console.log(event)
};
tuiCallEngine.on(EVENT.ON_CALL_RECEIVED, handleOnCallReceived);
```

参数如下表所示：

参数	类型	含义
sponsor	String	邀请者
inviteeList	Array<String>	同时还被邀请的人（**后续将废弃**）
userIDList	Array<String>	同时还被邀请的人
isFromGroup	Boolean	是否是群组通话
inviteData	Object	呼叫数据
inviteID	String	邀请 ID，标识一次邀请
userData	String	扩展字段：用于在邀请信令中增加扩展信息

callId	String	本次通话的唯一 ID。
roomID	Number	此次通话的音视频房间 ID。
callMediaType	Number	通话的媒体类型，视频通话、语音通话。
callRole	String	角色，枚举类型：主叫、被叫。

USER_ACCEPT

如果有用户接听，那么会收到该事件。

```
let handleUserAccept = function(event) {
    console.log('用户接听')
}
tuiCallEngine.on(EVENT.USER_ACCEPT, this.handleUserAccept, this);
```

参数如下表所示：

参数	类型	含义
userID	String	接听的用户 ID
userList	Array	未接听用户的信息（**后续将废弃**）

USER_ENTER

如果有用户同意进入通话，其他用户会收到该事件。

```
let handleUserEnter = function(event) {
    console.log('用户进入通话')
}
tuiCallEngine.on(EVENT.USER_ENTER, this.handleUserEnter, this);
```

参数如下表所示：

参数	类型	含义
userID	String	进房的用户 ID
playerList	Array	房间中用户的推流信息

USER_LEAVE

有用户离开此次通话时，通话中的其他用户会收到该事件。

```
let handleUserLeave = function(event) {
    console.log('用户离开通话')
}
tuiCallEngine.on(EVENT.USER_LEAVE, this.handleUserLeave, this);
```

参数如下表所示：

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
userID	String	退房的用户 ID
playerList	Array	房间中用户的推流信息

USER_UPDATE

用户更新。

```
let handleUserUpdate = function(event) {
    console.log('用户更新')
}
tuiCallEngine.on(EVENT.USER_UPDATE, this.handleUserUpdate, this)
```

参数如下表所示：

参数	类型	含义
pusher	String	更新后的自身推流信息
playerList	Array	更新后其他用户的推流信息

REJECT

通话被拒绝事件，在 1v1 通话中，只有主叫方会收到拒绝事件，在群组通话中，所有被邀请者都可以收到该事件。

```
let handleInviteeReject = function(event) {
    console.log('用户拒绝通话')
}
tuiCallEngine.on(EVENT.REJECT, this.handleInviteeReject, this);
```

参数如下表所示：

参数	类型	含义
userID	String	拒绝通话的用户 ID
invitee	String	拒绝通话的用户 ID（**后续将废弃**）
inviteID	String	邀请 ID, 标识一次邀请
reason	string	reject 表示拒绝

NO_RESP

邀请用户无应答。

- 在C2C通话中，只有发起方会收到无人应答的回调 例如 A 邀请 B、C 进入通话，B不应答，A可以收到该回调，但C不行
- 在IM群组通话中，所有被邀请人均能收到该回调 例如 A 邀请 B、C 进入通话，B不应答，A、C均能收到该回调

```
let handleNoResponse = function(event) {
    console.log('用户无响应')
}
tuiCallEngine.on(EVENT.NO_RESP, this.handleNoResponse, this);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
groupID	String	群ID, 唯一标识
sponsor	String	发起方的用户 ID
userIDList	Array<String>	触发超时无响应的用户列表
timeoutUserList	Array<String>	触发超时无响应的用户列表（**后续将废弃**）
inviteID	String	邀请 ID, 标识一次邀请

LINE_BUSY

用户忙线。

```
let handleLineBusy = function(event) {
    console.log('用户忙线')
}
tuiCallEngine.on(EVENT.LINE_BUSY, this.handleLineBusy, this);
```

参数如下表所示：

参数	类型	含义
userID	String	忙线用户 ID
invitee	String	忙线用户 ID（**后续将废弃**）
inviteID	String	邀请 ID, 标识一次邀请
reason	string	line busy 表示忙线

CALLING_CANCEL

表示此次通话被主叫取消（取消原因有可能是主叫主动取消、也有可能是来自于通话超时取消），被叫会收到，您可以通过监听这个事件来实现类似未接来电等显示逻辑。

1. A 呼叫 B，A 挂断，A 会抛出该事件，userID 就是 A
2. A 呼叫 B，B 一直接通超时，B 会抛出该事件，userID 是 B
3. A 呼叫 B、C、D，B 接通后，A 和 B 都挂断，此时 C 拒绝，此时 B 也会抛出该事件，事件里的 userID 为 C

> ⚠ **注意：**
> **计划后续废弃**

```
let handleCallingCancel = function(event) {
    console.log('通话被取消')
}
tuiCallEngine.on(EVENT.CALLING_CANCEL, this.handleCallingCancel, this);
```

参数如下表所示：

参数	类型	含义	
userID	String	通话取消的用户 ID	
invitee	String	通话取消的用户 ID（**后续将废弃**）	
inviteID	String	邀请 ID, 标识一次邀请	
callId	String	本次通话的唯一 ID。**v1.5.1+ 版本支持**	
roomID	Number	此次通话的音视频房间 ID。**v1.5.1+ 版本支持**	
callMediaType	Number	通话的媒体类型，视频通话、语音通话。**v1.5.1+ 版本支持**	
callRole	String	角色，枚举类型：主叫、被叫。**v1.5.1+ 版本支持**	

ON_CALL_CANCELED

如果通话未建立，都会抛出该事件。您可以通过监听该事件来实现类似未接来电、重置 UI 状态等显示逻辑。通话未建立的场景如下：

- 主叫取消：主叫抛出该事件，userID 为主叫；被叫也会抛出该事件，userID 为被叫；
- 被叫超时：主叫会同时抛出 NO_RESP 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫拒接：主叫会同时抛出 REJECT 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；
- 被叫忙线：主叫会同时抛出 LINE_BUSY 和 CALLING_CANCEL 事件，userID 为主叫；被叫抛出 CALLING_CANCEL 事件，userID 为被叫；

> ⚠ **注意：**
> **v1.5.1+ 版本支持**

```
let handleOnCallCanceled = function(event) {
  console.log(event.userID);
};
tuiCallEngine.on(EVENT.ON_CALL_CANCELED, handleOnCallCanceled);
```

参数如下表所示：

参数	类型	含义
userID	String	通话取消的用户 ID
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫、被叫

ON_CALL_BEGIN

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

> ⚠ 注意：
> v1.5.1+ 版本支持

```
let handleOnCallBegin = function(event) {
    console.log(event)
};
tuiCallEngine.on(EVENT.ON_CALL_BEGIN, handleOnCallBegin);
```

参数如下表所示：

参数	类型	含义
callId	String	本次通话的唯一 ID
roomID	Number	此次通话的音视频房间 ID
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，类型：主叫、被叫

CALL_END

表示通话挂断，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
let handleCallingEnd = function(event) {
    console.log('通话结束')
}
tuiCallEngine.on(EVENT.CALL_END, this.handleCallingEnd, this);
```

参数如下表所示：

参数	类型	含义
roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
callMediaType	Number	通话的媒体类型，视频通话、语音通话
callRole	String	角色，枚举类型：主叫（'inviter'）、被叫（'invitee'）、未知（''）
totalTime	Number	此次通话的时长，单位: 秒
userID	String	表示引起通话结束 userID
callEnd	Number	此次通话的时长（**后续将废弃**），单位: 秒
message	string	信息字段（**后续将废弃**）
callId	String	本次通话的唯一 ID。**v1.5.1+ 版本支持**

SDK_READY

SDK Ready 回调。

```
let handleSDKReady = function(event) {
```

```
    console.log('SDK Ready 回调')
}
tuiCallEngine.on(EVENT.SDK_READY, this.handleSDKReady, this);
```

KICKED_OUT

被踢下线。

```
let handleKickedOut = function(event) {
    console.log('被踢下线')
}
tuiCallEngine.on(EVENT.KICKED_OUT, this.handleKickedOut, this);
```

CALL_MODE

切换通话模式。

```
let handleCallMode = function(event) {
    console.log('切换通话模式')
}
tuiCallEngine.on(EVENT.CALL_MODE, this.handleCallMode, this);
```

ERROR

监听 SDK 的错误信息。

```
let handleError = function(event) {
    console.log('错误信息')
}
tuiCallEngine.on(EVENT.ERROR, this.handleError, this);
```

版权所有：腾讯云计算（北京）有限责任公司

uni-app（客户端）API 概览

最近更新时间：2024-01-16 09:44:21

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的头像、昵称
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

事件

目前 TUICallKit 抛出下述事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调

版权所有：腾讯云计算（北京）有限责任公司

onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，使用 TUICallEngine API，您可以做一些额外的操作（目前仅支持以下接口）。

API	描述
hangup	结束通话
accept	接听通话
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数

版权所有：腾讯云计算（北京）有限责任公司

TUICallKit

最近更新时间：2024-01-26 15:06:11

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入 TUICallKit 。

API 概览

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
setCallingBell	设置自定义来电铃音
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能

API 详情

login

登录

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  SDKAppID: 0,
  userID: 'mike',
  userSig: '',
};
TUICallKit.login(options, (res) => {
  if (res.code === 0) {
    console.log('login success');
  } else {
    console.log(`login failed, error message = ${res.msg}`);
  }
});
```

参数	类型	含义

options	Object	初始化参数
options.SDKAppID	Number	用户 SDKAppID
options.userID	String	用户 ID
options.userSig	String	用户签名 userSig
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

logout

登出

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
TUICallKit.logout((res) => {
  if (res.code === 0) {
    console.log('logout success');
  } else {
    console.log(`logout failed, error message = ${res.msg}`);
  }
});
```

参数	类型	含义
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

setSelfInfo

设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是URL格式。

> ⚠ **注意：通话中使用该接口修改用户信息，UI 不会立即更新，需要等到下次通话才能看到变化。**

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  nickName: 'jack',
  avatar: 'https:/****/user_avatar.png'
}
TUICallKit.setSelfInfo(options, (res) => {
  if (res.code === 0) {
    console.log('setSelfInfo success');
  } else {
    console.log(`setSelfInfo failed, error message = ${res.msg}`);
  }
});
```

参数	类型	含义
options	Object	初始化参数
options.nickName	String	目标用户的昵称，非必填

options.avatar	String	目标用户的头像，非必填
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

call

拨打电话（1v1通话）

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  userID: 'mike',
  callMediaType: 1, // 语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
  roomID: 0,
  strRoomID: '1223', // 使用字符串房间号
};
TUICallKit.call(options, (res) => {
  if (res.code === 0) {
    console.log('call success');
  } else {
    console.log(`call failed, error message = ${res.msg}`);
  }
});
```

参数如下表所示：

参数	类型	含义
options	Object	初始化参数
options.userID	String	目标用户的 userID
options.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
options.roomID	Number	自定义数字房间号。**只要有 roomID，就用的是数字房间号，即使 strRoomID 存在。**
options.strRoomID	String	自定义字符串房间号。**如果想使用字符串房间号，设置 strRoomID 后需要将 roomID = 0。**
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

groupCall

发起群组通话，注意：使用群组通话前需要创建IM 群组，如果已经创建，请忽略。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  groupID: 'myGroup',
  userIDList: ['mike', 'tom'],
  callMediaType: 1, // 语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
};
TUICallKit.groupCall(options, (res) => {
```

```
  if (res.code === 0) {
    console.log('call success');
  } else {
    console.log(`call failed, error message = ${res.msg}`);
  }
});
```

参数	类型	含义
options	Object	初始化参数
options.groupID	String	此次群组通话的群 ID
options.userIDList	List	目标用户的userId 列表
options.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
options.roomID	Number	自定义数字房间号。**只要有 roomID，就用的是数字房间号，即使 strRoomID 存在。**
options.strRoomID	String	自定义字符串房间号。**如果想使用字符串房间号，设置 strRoomID 后需要将 roomID = 0。**

joinInGroupCall

加入群组中已有的音视频通话。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  roomID: 9898,
  groupID: 'myGroup',
  callMediaType: 1, // 语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
};
TUICallKit.joinInGroupCall(options, (res) => {
  if (res.code === 0) {
    console.log('joinInGroupCall success');
  } else {
    console.log(`joinInGroupCall failed, error message = ${res.msg}`);
  }
});
```

参数	类型	含义
options	Object	初始化参数
options.roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
options.groupID	String	此次群组通话的群 ID
options.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

版权所有：腾讯云计算（北京）有限责任公司

setCallingBell

设置自定义来电铃音，这里仅限传入本地文件地址，需要确保该文件目录是应用可以访问的。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');

// 【1】通过 uni.saveFile 保存音频文件到本地，具体参考 saveFile 接口:
https://zh.uniapp.dcloud.io/api/file/file.html#savefile
const tempFilePath = './static/rain.mp3'; // 本地存放的音频文件
let musicFilePath = '';
uni.saveFile({
    tempFilePath: tempFilePath,
    success: (res) => {
        console.warn('保存文件成功 = ', JSON.stringify(res)); // 获取的是相对路径
        musicFilePath =  res.savedFilePath;

        // 【2】相对路径转绝对路径，否则访问不到
        musicFilePath = plus.io.convertLocalFileSystemURL(musicFilePath); // 转绝对路径

        // 【3】设置铃声
        TUICallKit.setCallingBell(musicFilePath, (res) => {
          if (res.code === 0) {
            console.log('setCallingBell success');
          } else {
            console.log(`setCallingBell failed, error message = ${res.msg}`);
          }
        });
    },
    fail: (err) => {
      console.error('保存文件失败');
    },
});
```

参数	类型	含义
filePath	String	来电铃音本地文件地址
callback	Function	回调函数，code = 0 表示调用成功；code != 0 表示调用失败，失败原因见 msg

enableMuteMode

开启/关闭静音模式。

开启后，收到通话请求，不会播放来电铃声。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const enable = true;
TUICallKit.enableMuteMode(enable);
```

参数	类型	含义
enable	Boolean	开启、关闭静音；true 表示开启静音

enableFloatWindow

版权所有：腾讯云计算（北京）有限责任公司

开启/关闭悬浮窗功能，设置为false后，通话界面左上角的悬浮窗按钮会隐藏。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const enable = true;
TUICallKit.enableFloatWindow(enable);
```

参数	类型	含义
enable	Boolean	开启、关闭悬浮窗功能；true 表示开启浮窗

TUICallEngine

最近更新时间：2024-01-17 10:47:12

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口实现交互。**目前仅提供如下接口，如有需求可到这里 反馈 。**

> **注意：**
> 需要结合 腾讯云音视频通话插件 TencentCloud-TUICallKit 插件使用，不能单独使用。下述接口也是结合客户有自动挂断、设置渲染模式的需求下，才支持的。

API 概览

API	描述
hangup	结束通话
accept	接听通话
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数

API 详情

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

```
const TUICallEngine = uni.requireNativePlugin('TencentCloud-TUICallKit-TUICallEngine');
TUICallEngine.hangup();
```

accept

接受当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。

```
const TUICallEngine = uni.requireNativePlugin('TencentCloud-TUICallKit-TUICallEngine');
TUICallEngine.accept();
```

setVideoRenderParams

设置用户视频画面的渲染模式。

```
const TUICallEngine = uni.requireNativePlugin('TencentCloud-TUICallKit-TUICallEngine');
const params = {
    userID: '234',
    fillMode: 0, // 0-填充模式，1—适应模式
    rotation: 1, // 0~3分别对应值：Rotation_0, Rotation_90, Rotation_180, Rotation_270;
};
TUICallEngine.setVideoRenderParams(params, (res) => {
```

版权所有：腾讯云计算（北京）有限责任公司

```
    console.warn('res = ', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
userID	String	目标用户的 userID
params	Object	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。
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。

```
const TUICallEngine = uni.requireNativePlugin('TencentCloud-TUICallKit-TUICallEngine');
const params = {
    resolution: 108,
    resolutionMode: 0, // 0--横屏，1--竖屏
};
TUICallEngine.setVideoEncoderParams(params, (res) => {
    console.warn('res = ', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
resolution	Number	视频分辨率 • 62 -- 宽高比 16:9；分辨率 640x360； • 64 -- 宽高比 4:3；分辨率 960x720； • 108 -- 宽高比 16:9；分辨率 640x360； • 110 -- 宽高比 16:9；分辨率 960x540； • 112 -- 宽高比 16:9；分辨率 1280x720； • 114 -- 宽高比 16:9；分辨率 1920x1080；
resolutionMode	Number	分辨率模式（横屏分辨率 or 竖屏分辨率） • 0-Landscape（横屏） • 1-Portrait（竖屏）

版权所有：腾讯云计算（北京）有限责任公司

事件

最近更新时间：2024-01-16 09:44:21

事件简介

可以监听 TUICallKit 对应的事件，进行提示或其它交互。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被踢下线
onUserSigExpired	在线时票据过期

回调事件详情

通过 globalEvent 监听原生插件抛出的事件。

```
const TUICallingEvent = uni.requireNativePlugin('globalEvent');
```

onError

错误回调。

> **说明**

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。

```
TUICallingEvent.addEventListener('onError', (res) => {
  console.log('onError', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	错误回调参数
res.code	Number	错误码
res.msg	String	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

```
TUICallingEvent.addEventListener('onCallReceived', (res) => {
  console.log('onCallReceived', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.callerId	String	主叫 ID（邀请方）
res.calleeIdList	Array<String>	被叫 ID 列表（被邀请方）
res.groupId	String	群通话 ID。在群组通话时返回，1v1 返回 groupId 为空字符串
res.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
res.userData	String	用户增加的扩展字段

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

- 主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）。
- 被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）。
- 异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）。

```
TUICallingEvent.addEventListener('onCallCancelled', (res) => {
  console.log('onCallCancelled', res);
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

```
TUICallingEvent.addEventListener('onCallBegin', (res) => {
  console.log('onCallBegin', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
res.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
res.callRole	Number	角色，枚举类型：主叫(callRole = 1)、被叫(callRole = 2)

onCallEnd

表示通话结束，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
TUICallingEvent.addEventListener('onCallEnd', (res) => {
  console.log('onCallEnd', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.roomID	Number	此次通话的音视频房间 ID，目前仅支持数字房间号，后续版本会支持字符串房间号
res.callMediaType	Number	通话的媒体类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)

版权所有：腾讯云计算（北京）有限责任公司

res.callRole	Number	角色，枚举类型：主叫(callRole = 1)、被叫(callRole = 2)
res.totalTime	Number	此次通话的时长，单位：秒

> **注意**
> 客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用 REST API 来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。

```
TUICallingEvent.addEventListener('onCallMediaTypeChanged', (res) => {
  console.log('onCallMediaTypeChanged', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.oldCallMediaType	Number	旧的通话类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
res.newCallMediaType	Number	新的通话类型，比如：语音通话(callMediaType = 1)、视频通话(callMediaType = 2)

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。

```
TUICallingEvent.addEventListener('onUserReject', (res) => {
  console.log('onUserReject', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。

```
TUICallingEvent.addEventListener('onUserNoResponse', (res) => {
  console.log('onUserNoResponse', JSON.stringify(res));
});
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	无响应用户的 ID

onUserLineBusy

通话忙线回调。

```
TUICallingEvent.addEventListener('onUserLineBusy', (res) => {
  console.log('onUserLineBusy', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	忙线用户的 ID

onUserJoin

有用户进入此次通话的回调。

```
TUICallingEvent.addEventListener('onUserJoin', (res) => {
  console.log('onUserJoin', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	加入当前通话的用户 ID

onUserLeave

有用户离开此次通话的回调。

```
TUICallingEvent.addEventListener('onUserLeave', (res) => {
  console.log('onUserLeave', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数

版权所有：腾讯云计算（北京）有限责任公司

res.userId	String	离开当前通话的用户 ID

onUserVideoAvailable

用户是否开启视频上行回调。

```
TUICallingEvent.addEventListener('onUserVideoAvailable', (res) => {
  console.log('onUserVideoAvailable', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	通话用户 ID
res.isVideoAvailable	boolean	用户视频是否可用

onUserAudioAvailable

用户是否开启音频上行回调。

```
TUICallingEvent.addEventListener('onUserAudioAvailable', (res) => {
  console.log('onUserAudioAvailable', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.userId	String	用户 ID
res.isAudioAvailable	boolean	用户音频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

```
TUICallingEvent.addEventListener('onUserVoiceVolumeChanged', (res) => {
  console.log('onUserVoiceVolumeChanged', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.volumeMap	Map<String,	音量表，根据每个 userId 可以获取对应用户的音量大小，音量最小值为0，

版权所有：腾讯云计算（北京）有限责任公司

	Integer>	音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。

```
TUICallingEvent.addEventListener('onUserNetworkQualityChanged', (res) => {
  console.log('onUserNetworkQualityChanged', JSON.stringify(res));
});
```

参数如下表所示：

参数	类型	含义
res	Object	回调参数
res.networkQuality List	List	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。

```
TUICallingEvent.addEventListener('onKickedOffline', (res) => {
  console.log('onKickedOffline', JSON.stringify(res));
});
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。

```
TUICallingEvent.addEventListener('onUserSigExpired', (res) => {
  console.log('onUserSigExpired', JSON.stringify(res));
});
```

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：2023-10-23 17:11:51

TUICallKit (含 UI 接口)

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用 TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
setCallingBell	设置自定义来电铃音

TUICallEngine (无 UI 接口)

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套 API 根据您的业务需求自定义封装。

API	描述
init	初始化 SDK 函数，完成包含通话服务鉴权在内初始化动作
unInit	反初始化函数，释放 TUICallEngine 使用到的相关资源。
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话

版权所有：腾讯云计算（北京）有限责任公司

inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式
setVideoEncoderParams	设置视频编码的编码参数
queryRecentCalls	查询通话记录
deleteRecordCalls	删除通话记录
setBeautyLevel	设置美颜级别

TUICallObserver

TUICallObserver 是回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调

版权所有：腾讯云计算（北京）有限责任公司

onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxx 用户是否有视频流的回调
onUserAudioAvailable	xxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被移下线
onUserSigExpired	在线时票据过期

TUICallKit

最近更新时间：2023-12-29 17:31:22

TUICallKit API 简介

TUICallKit API 是音视频通话组件的**含 UI 接口**，使用TUICallKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，更详细的接入步骤，详情请参见 快速接入（TUICallKit）。

API 概览

API	描述
login	登录
logout	登出
setSelfInfo	设置用户的昵称、头像
call	发起 1v1 通话
groupCall	发起群组通话
joinInGroupCall	主动加入当前的群组通话中
enableMuteMode	开启/关闭静音模式
enableFloatWindow	开启/关闭悬浮窗功能
setCallingBell	设置自定义来电铃音

API 详情

login

登录。

```
TUIResult result = TUICallKit.instance.login(sdkAppId: 1400000001,  // 请替换为第一步得到的SDKAppID
                                             userId: 'xxxxxx',      // 请替换为您的User ID
                                             userSig: 'xxxxxx');    // 您可以在控制台计算一个UserSig并填到该位置
// 返回值TUIResult的定义：
class TUIResult {
    String code;
    String? message;
    TUIResult({required this.code, required this.message});
}
```

参数	类型	含义
sdkAppId	int	用户 SDKAppID
userId	String	用户 ID
userSig	String	用户签名 userSig

返回值	TUIResult	包含code和message信息：code = 0 表示调用成功；code != 0 表示调用失败，失败原因见message

logout

登出。

```
TUIResult result = TUICallKit.instance.logout()

// 返回值TUIResult的定义：
class TUIResult {
    String code;
    String? message;
    TUIResult({required this.code, required this.message});
}
```

参数	类型	含义
返回值	TUIResult	包含code和message信息：code = 0 表示调用成功；code != 0 表示调用失败，失败原因见message

setSelfInfo

设置用户昵称、头像。用户昵称不能超过500字节，用户头像必须是URL格式。

```
// 函数定义为: Future<TUIResult> setSelfInfo(String nickname, String avatar)
TUIResult result = TUICallKit.instance.setSelfInfo('userName', 'url:********');

// 返回值TUIResult的定义：
class TUIResult {
    String code;
    String? message;
    TUIResult({required this.code, required this.message});
}
```

参数	类型	含义
nickName	String	目标用户的昵称，非必填
avatar	String	目标用户的头像，非必填
返回值	TUIResult	包含code和message信息：code = 0 表示调用成功；code != 0 表示调用失败，失败原因见message

call

拨打电话（1v1通话）。

```
// call的定义为：Future<void> call(String userId, TUICallMediaType callMediaType)
TUICallKit.instance.call('mike', TUICallMediaType.video);
// TUICallMediaType 定义为：
enum TUICallMediaType {
    none,
```

```
    audio,
    video
}
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userID
callMediaType	TUICallMediaType	通话的媒体类型，比如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

groupCall

发起群组通话，注意：使用群组通话前需要创建IM 群组，如果已经创建，请忽略。

```
// groupCall定义为：Future<void> groupCall(String groupId, List<String> userIdList, TUICallMediaType callMediaType)
TUICallKit.instance.groupCall('0001', ['denny', 'mike', 'tommy'], TUICallMediaType.video);
```

enableFloatWindow

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List<String>	目标用户的userId 列表
callMediaType	TUICallMediaType	通话的媒体类型，比如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

joinInGroupCall

加入群组中已有的音视频通话。

```
// joinInGroupCall定义为：Future<void> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaType callMediaType)
final roomId = TUIRoomID(intRoomId: 123);
TUICallKit.instance.joinInGroupCall(, ['denny', 'mike', 'tommy'], TUICallMediaType.video);
```

参数	类型	含义
roomId	TUIRoomID	此次通话的音视频房间 ID
groupId	String	此次群组通话的群 ID
callMediaType	TUICallMediaType	通话的媒体类型，比如：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

enableMuteMode

开启后，收到通话请求，不会播放来电铃声。

```
final enable = true;
TUICallKit.instance..enableMuteMode(enable);
```

参数	类型	含义
enable	Boolean	开启、关闭静音；true 表示开启静音

enableFloatWindow

开启/关闭悬浮窗功能，设置为false后，通话界面左上角的悬浮窗按钮会隐藏。

```
final enable = true;
TUICallKit.instance..enableFloatWindow(enable);
```

参数	类型	含义
enable	Boolean	开启、关闭悬浮窗功能；true 表示开启浮窗

setCallingBell

自定义来电铃声：将铃声文件添加至主工程的 `assets` 资源中，传入资源文件名称即可。

```
final fileName = 'ring.mp3';
TUICallKit.instance.setCallingBell(fileName);
```

TUICallEngine

最近更新时间：2023-08-21 14:27:54

TUICallEngine API 简介

TUICallEngine API 是音视频通话组件的**无 UI 接口**，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。

API 概览

API	描述
init	初始化SDK函数，完成包含通话服务鉴权在内初始化动作
unInit	反初始化函数，释放TUICallEngine使用到的相关资源。
addObserver	增加事件回调
removeObserver	移除回调接口
call	发起 1v1 通话
groupCall	发起群组通话
accept	接听通话
reject	拒绝通话
hangup	结束通话
ignore	忽略通话
inviteUser	在群组通话中，邀请其他人加入
joinInGroupCall	主动加入当前的群组通话中
switchCallMediaType	切换通话媒体类型，比如视频通话切音频通话
startRemoteView	开始订阅远端用户视频流
stopRemoteView	停止订阅远端用户视频流
openCamera	开启摄像头
closeCamera	关闭摄像头
switchCamera	切换前后摄像头
openMicrophone	打开麦克风
closeMicrophone	关闭麦克风
selectAudioPlaybackDevice	选择音频播放设备（听筒/免提）
setSelfInfo	设置用户的头像、昵称
enableMultiDeviceAbility	开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）
setVideoRenderParams	设置用户视频画面的渲染模式

版权所有：腾讯云计算（北京）有限责任公司

setVideoEncoderParams	设置视频编码的编码参数
queryRecentCalls	查询通话记录
deleteRecordCalls	删除通话记录
setBeautyLevel	设置美颜级别

API 详情

init

初始化函数，请在使用所有功能之前先调用该函数，以便完成包含通话服务鉴权在内初始化动作。

```
Future<TUIResult> init(int sdkAppID, String userId, String userSig)
```

unInit

反初始化函数，释放 `TUICallEngine` 使用到的相关资源。

```
Future<TUIResult> unInit()
```

addObserver

添加回调接口，您可以通过这个接口，监听 `TUICallObserver` 相关的事件回调。

```
Future<void> addObserver(TUICallObserver observer)
```

removeObserver

移除回调接口。

```
Future<void> removeObserver(TUICallObserver observer)
```

call

拨打电话（1v1通话）。

```
Future<TUIResult> call(String userId, TUICallMediaType mediaType, TUICallParams params)
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

groupCall

版权所有：腾讯云计算（北京）有限责任公司

发起群组通话。

> ⚠ 注意：
> 使用群组通话前需要创建 IM 群组，如果已经创建，请忽略。

```
Future<TUIResult> groupCall(String groupId, List<String> userIdList, TUICallMediaType mediaType,
TUICallParams params)
```

参数	类型	含义
groupId	String	此次群组通话的群 ID
userIdList	List<String>	目标用户的 userId 列表
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话
params	TUICallParams	通话参数扩展字段，例如：房间号、通话邀请超时时间，离线推送自定义内容等

accept

接收当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数接听来电。

```
Future<TUIResult> accept()
```

reject

拒绝当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数拒绝来电。

```
Future<TUIResult> reject()
```

ignore

忽略当前通话，当您作为被叫收到 `onCallReceived()` 的回调时，可以调用该函数忽略来电，此时主叫会收到 `onUserLineBusy` 的回调。
备注：如果您的业务中存在直播、会议等场景，在直播/会议中的情况时，也可以调用这个函数来忽略此次来电。

```
Future<TUIResult> ignore()
```

hangup

挂断当前通话，当您处于通话中，可以调用该函数结束通话。

```
Future<TUIResult> hangup()
```

inviteUser

邀请用户加入此次群组通话。
使用场景：一个群组通话中的用户主动邀请其他人时使用。

```
Future<void> iniviteUser(List<String> userIdList, TUICallParams params, TUIValueCallback callback)
```

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
userIdList	List<String>	目标用户的 userId 列表
params	TUICallParams	通话参数扩展字段，例如：离线推送自定义内容

joinInGroupCall

主动加入此次群组通话。
使用场景：群组内用户主动加入此次群组通话使用。

```
Future<TUIResult> joinInGroupCall(TUIRoomId roomId, String groupId, TUICallMediaType mediaType)
```

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID.
groupId	String	此次群组通话的群 ID.
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话.

switchCallMediaType

切换视频通话到语音通话。

```
Future<void> switchCallMediaType(TUICallMediaType mediaType)
```

参数	类型	含义
mediaType	TUICallMediaType	通话的媒体类型，比如视频通话、语音通话

startRemoteView

设置显示视频画面的 View 对象。

```
Future<void> startRemoteView(String userId, intviewId)
```

参数	类型	含义
userId	String	目标用户的 userId
intviewId	int	显示视频画面的 widget 对应的 Id

stopRemoteview

停止订阅远端用户的视频数据。

```
Future<void> stopRemoteView(String userId)
```

参数	类型	含义

userId	String	目标用户的 userId

openCamera

开启摄像头。

```
Future<TUIResult> openCamera(TUICamera camera, int? viewId)
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头
viewId	int	显示视频画面的widget对应的Id

closeCamera

关闭摄像头。

```
Future<void> closeCamera()
```

switchCamera

切换前后摄像头。

```
Future<void> switchCamera(TUICamera camera)
```

参数	类型	含义
camera	TUICamera	前置/后置 摄像头

openMicrophone

打开麦克风。

```
Future<TUIResult> openMicrophone()
```

closeMicrophone

关闭麦克风。

```
Future<void> closeMicrophone()
```

selectAudioPlaybackDevice

选择音频播放设备。
目前支持听筒、扬声器，在通话场景中，可以使用这个接口来开启/关闭免提模式。

```
Future<void> selectAudioPlaybackDevice(TUIAudioPlaybackDevice device)
```

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

device	TUIAudioPlaybackDevice	听筒/扬声器

setSelfInfo

设置用户昵称、头像。
用户昵称不能超过500字节，用户头像必须是 URL 格式。

```
Future<TUIResult> setSelfInfo(String nickname, String avatar)
```

参数	类型	含义
nickname	String	用户昵称
avatar	String	用户头像（格式为 URL）

enableMultiDeviceAbility

开启/关闭 TUICallEngine 的多设备登录模式（尊享版套餐支持）。

```
Future<TUIResult> enableMultiDeviceAbility(bool enable)
```

setVideoRenderParams

设置用户视频画面的渲染模式。

```
Future<TUIResult> setVideoRenderParams(String userId, VideoRenderParams params)
```

参数如下表所示：

参数	类型	含义
userId	String	目标用户的 userId
params	VideoRenderParams	视频画面渲染参数，例如：画面旋转角度、填充模式

setVideoEncoderParams

设置视频编码的编码参数。
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。

```
Future<TUIResult> setVideoEncoderParams(VideoEncoderParams params)
```

参数如下表所示：

参数	类型	含义
params	VideoEncoderParams	视频编码的参数，例如：分辨率、视频宽高比

queryRecentCalls

查询通话记录。

版权所有：腾讯云计算（北京）有限责任公司

```
Future<void> queryRecentCalls(TUICallRecentCallsFilter filter, TUIValueCallback callback)
```

参数如下表所示：

参数	类型	含义
filter	TUICallRecentCallsFilter	查询过滤条件

deleteRecordCalls

删除通话记录。

```
Future<void> deleteRecordCalls(List<String> callIdList, TUIValueCallback callback)
```

参数如下表所示：

参数	类型	含义
callIdList	List<String>	待删除记录的 Id 列表

setBeautyLevel

设置美颜级别。

```
Future<TUIResult> setBeautyLevel(double level)
```

参数如下表所示：

参数	类型	含义
level	double	美颜等级，使用范围0.0～9.0

TUICallObserver

最近更新时间：2023-12-29 17:31:22

TUICallObserver API 简介

TUICallObserver 是 TUICallKit 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

回调事件概览

API	描述
onError	通话过程中错误回调
onCallReceived	通话请求的回调
onCallCancelled	通话取消的回调
onCallBegin	通话接通的回调
onCallEnd	通话结束的回调
onCallMediaTypeChanged	通话媒体类型发生改变的回调
onUserReject	xxxx 用户拒绝通话的回调
onUserNoResponse	xxxx 用户不响应的回调
onUserLineBusy	xxxx 用户忙线的回调
onUserJoin	xxxx 用户加入通话的回调
onUserLeave	xxxx 用户离开通话的回调
onUserVideoAvailable	xxxx 用户是否有视频流的回调
onUserAudioAvailable	xxxx 用户是否有音频流的回调
onUserVoiceVolumeChanged	所有用户音量大小的反馈回调
onUserNetworkQualityChanged	所有用户网络质量的反馈回调
onKickedOffline	当前用户被移下线
onUserSigExpired	在线时票据过期

回调事件详情

通过 addObserver 监听Flutter插件抛出的事件。

```
TUICallEngine.instance.addObserver(TUICallObserver(
  onError: (int code, String message) {

  },onCallCancelled: (String callerId) {
```

```
    }, onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole) {

    }, onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole, double totalTime)
{

    }, onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType newCallMediaType) {

    }, onUserReject: (String userId) {

    }, onUserNoResponse: (String userId) {

    }, onUserLineBusy: (String onUserLineBusy) {

    }, onUserJoin: (String userId) {

    }, onUserLeave: (String userId) {

    }, onUserVideoAvailable: (String userId, bool isVideoAvailable) {

    }, onUserAudioAvailable: (String userId, bool isAudioAvailable) {

    }, onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {

    }, onCallReceived: (String callerId, List<String> calleeIdList, String groupId, TUICallMediaType
callMediaType) {

    }, onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {

    }, onKickedOffline: () {

    }, onUserSigExpired: () {

    }
));
```

onError

错误事件回调。

> **说明**
> 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。

```
TUICallEngine.instance.addObserver(TUICallObserver(
  onError: (int code, String message) {
    //您的回调处理逻辑
  }
));
```

参数如下表所示：

参数	类型	含义
code	int	错误码

message	String	错误信息

onCallReceived

收到一个新的来电请求回调，被叫会收到，您可以通过监听这个事件，来决定是否显示通话接听界面。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallReceived: (String callerId, List<String> calleeIdList, String groupId, TUICallMediaType
callMediaType) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
callerId	String	主叫 ID（邀请方）
calleeIdList	List<String>	被叫 ID 列表（被邀请方）
groupId	String	群组ID
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`

onCallCancelled

表示此次通话主叫取消、被叫超时、拒接等，涉及多个场景，您可以通过监听这个事件来实现类似未接来电、重置 UI 状态等显示逻辑。

- 主叫取消：主叫收到该回调（userId 为自己）；被叫收到该回调（userId 为**主叫的 ID**）。
- 被叫超时：主叫会同时收到 onUserNoResponse 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫拒接：主叫会同时收到 onUserReject 和 onCallCancelled 回调（userId 是自己的 ID）；被叫收到 onCallCancelled 回调（userId 是自己的 ID）。
- 被叫忙线：主叫会同时收到 onUserLineBusy 和 onCallCancelled 回调（userId 是自己的 ID）。
- 异常中断：被叫接收通话失败，收到该回调（userId 是自己的 ID）。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallCancelled: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	用户的 ID

onCallBegin

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来开启云端录制、内容审核等流程。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallBegin: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 ID
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`
callRole	TUICallRole	角色，枚举类型：主叫(`TUICallRole.caller`)、被叫(`TUICallRole.called`)

onCallEnd

表示通话接通，主叫和被叫都可以收到，您可以通过监听这个事件来显示通话时长、通话类型等信息，或者来停止云端的录制流程。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallEnd: (TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole, double totalTime) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
roomId	TUIRoomId	此次通话的音视频房间 Id
callMediaType	TUICallMediaType	通话的媒体类型，示例：`TUICallMediaType.video` 或 `TUICallMediaType.audio`
callRole	Number	角色，枚举类型：主叫(callRole = 1)、被叫(callRole = 2)
totalTime	double	此次通话的时长，单位 ms

> **注意**
> 客户端的事件一般都会随着杀进程等异常事件丢失掉，如果您需要通过监听通话时长来完成计费等逻辑，建议可以使用 REST API 来完成这类流程。

onCallMediaTypeChanged

表示通话的媒体类型发生变化。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onCallMediaTypeChanged: (TUICallMediaType oldCallMediaType, TUICallMediaType newCallMediaType) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
oldCallMediaType	TUICallMediaType	通话的媒体类型，示例：TUICallMediaType.video 或 TUICallMediaType.audio
newCallMediaType	TUICallMediaType	通话的媒体类型，示例：TUICallMediaType.video 或 TUICallMediaType.audio

onUserReject

通话被拒绝的回调，在1v1 通话中，只有主叫方会收到拒绝回调，在群组通话中，所有被邀请者都可以收到该回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserReject: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
res.userId	String	拒绝用户的 ID

onUserNoResponse

对方无回应的回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNoResponse: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	无响应用户的 ID

onUserLineBusy

通话忙线回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLineBusy: (String onUserLineBusy) {
      //您的回调处理逻辑
    },
));
```

参数如下表所示：

参数	类型	含义
userId	String	忙线用户的 ID

onUserJoin

有用户进入此次通话的回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserJoin: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	加入当前通话的用户 ID

onUserLeave

有用户离开此次通话的回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserLeave: (String userId) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	离开当前通话的用户 ID

onUserVideoAvailable

用户是否开启视频上行回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVideoAvailable: (String userId, bool isVideoAvailable) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	通话用户 ID
isVideoAvailable	bool	用户视频是否可用

版权所有：腾讯云计算（北京）有限责任公司

onUserAudioAvailable

用户是否开启音频上行回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserAudioAvailable: (String userId, bool isAudioAvailable) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID
isAudioAvailable	bool	用户音频是否可用

onUserVoiceVolumeChanged

用户通话音量的回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserVoiceVolumeChanged: (Map<String, int> volumeMap) {
      //您的回调处理逻辑
    }
));
```

参数如下表所示：

参数	类型	含义
volume Map	Map<String, int>	音量表，根据每个 userId 可以获取对应用户的音量大小，音量最小值为0，音量最大值为100

onUserNetworkQualityChanged

用户网络质量的回调。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserNetworkQualityChanged: (List<TUINetworkQualityInfo> networkQualityList) {
      //您的回调处理逻辑
    }
));

//TUINetworkQualityInfo的定义如下：
class TUINetworkQualityInfo {
    String userId;
    TUINetworkQuality quality;
    TUINetworkQualityInfo({required this.userId, required this.quality});
}
// TUINetworkQuality的定义如下：
enum TUINetworkQuality {
  unknown,
```

```
  excellent,
  good,
  poor,
  bad,
  vBad,
  down
}
```

参数如下表所示：

参数	类型	含义
networkQualityList	List<TUINetworkQualityInfo>	网络状态，根据每个 userId 可以获取对应用户当前的网络质量

onKickedOffline

当前用户被踢下线：此时可以 UI 提示用户，并再次重新调用初始化。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onKickedOffline: () {
      //您的回调处理逻辑
    }
));
```

onUserSigExpired

在线时票据过期：此时您需要生成新的 userSig，并再次重新调用初始化。

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onUserSigExpired: () {
      //您的回调处理逻辑
    }
));
```

类型定义

最近更新时间：2023-08-21 14:27:54

常用结构

TUIResult

调用 API 的返回值

类型	类型	描述
code	String	code 为空 "" 表示调用成功，code 不为空 "" 表示调用失败
message	String	失败原因

TUIRoomId

房间 ID

类型	类型	描述
intRoomId	int	数字房间号
strRoomId	String	字符串房间号

VideoRenderParams

视频画面的渲染参数

类型	类型	描述
fillMode	FillMode	视频画面填充模式
rotation	Rotation	视频画面旋转方向

VideoEncoderParams

视频编码参数

类型	类型	描述
resolution	Resolution	视频分辨率
resolutionMode	ResolutionMode	视频宽高比模式

TUICallParams

通话参数

类型	类型	描述
roomId	TUIRoomId	房间 ID
offlinePushInfo	TUIOfflinePushInfo	厂商离线推送配置信息

timeout	String	超时时常
userData	String	用户自定义数据

TUIOfflinePushInfo

厂商离线推送配置信息，详情配置步骤请参考：离线唤醒

类型	类型	描述
title	String	离线推送展示通知栏标题
desc	String	离线推送展示通知栏内容
ignoreIOSBadge	bool	离线推送忽略 badge 计数（仅对 iOS 生效），如果设置为 true，在 iOS 接收端，这条消息不会使 APP 的应用图标未读计数增加
iOSSound	String	离线推送声音设置（仅对 iOS 生效）。当 sound = IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。当 sound = IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀）设置给 iOSSound
androidSound	String	离线推送声音设置（仅对 Android 生效, IMSDK 6.1 及以上版本支持）。只有华为和谷歌手机支持设置声音提示，小米手机设置声音提示，请您参照：小米自定义铃声。另外，谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须调用 setAndroidFCMChannelID 设置好 channelID，才能生效
androidOPPOChannelID	String	离线推送设置 OPPO 手机 8.0 系统及以上的渠道 ID
androidVIVOClassification	int	VIVO 推送消息分类 (待废弃接口，VIVO 推送服务于 2023 年 4 月 3 日优化消息分类规则，推荐使用 setAndroidVIVOCategory 设置消息类别)。0：运营消息 1：系统消息，默认取值为 1
androidXiaoMiChannelID	String	小米手机 8.0 系统及以上的渠道 ID
androidFCMChannelID	String	FCM 通道手机 8.0 系统及以上的渠道 ID
androidHuaWeiCategory	String	华为推送消息分类，详见：华为消息分类标准
isDisablePush	bool	是否关闭推送（默认开启推送）
iOSPushType	TUICallIOSOfflinePushType	iOS 离线推送类型，默认：APNs

TUICallRecords

通话记录信息

类型	类型	描述
callId	String	通话记录 ID

inviter	String	邀请者 ID
inviteList	List<String>	被邀请用户 ID 列表
scene	TUICallScene	通话场景
mediaType	TUICallMediaType	通话媒体类型
groupId	String	群组 ID
role	TUICallRole	通话角色
result	TUICallResultType	通话结果类型
beginTime	int	开始时间
totalTime	int	总时间

TUICallRecentCallsFilter

通话记录过滤条件

类型	类型	描述
begin	double	开始时间
end	double	结束时间
resultType	TUICallResultType	通话记录类型

枚举定义

TUICallMediaType

通话类型

类型	取值	描述
none	0	未知类型
audio	1	语音通话
video	2	视频通话

TUICallRole

通话角色

类型	取值	描述
none	0	未知类型
caller	1	主叫（邀请方）
called	2	被叫（被邀请方）

TUICallStatus

通话状态

类型	取值	描述
none	0	未知类型
waiting	1	通话等待中
accept	2	通话已接听

TUICallScene

通话场景

类型	取值	描述
groupCall	0	群组通话
multiCall	1	匿名群组通话（暂不支持，敬请期待）
singleCall	2	单人通话

TUINetworkQuality

网络质量

类型	取值	描述
unknown	0	未定义
excellent	1	当前网络非常好
good	2	当前网络比较好
poor	3	当前网络一般
bad	4	当前网络较差
vBad	5	当前网络很差
down	6	当前网络不满足通话的最低要求

FillMode

视频画面填充模式

类型	取值	描述
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。
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。

Rotation

视频画面旋转方向

版权所有：腾讯云计算（北京）有限责任公司

类型	取值	描述
rotation_0	0	不旋转
rotation_90	1	顺时针旋转90度
rotation_180	2	顺时针旋转180度
rotation_270	3	顺时针旋转270度

ResolutionMode

视频宽高比模式

类型	取值	描述
landscape	0	横屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Landscape = 640 × 360
portrait	1	竖屏分辨率，例如：Resolution.Resolution_640_360 + ResolutionMode.Portrait = 360 × 640

Resolution

视频分辨率

类型	取值	描述
resolution_640_360	0	宽高比 16:9；分辨率 640x360；建议码率（VideoCall）500kbps
resolution_640_480	1	宽高比 4:3；分辨率 640x480；建议码率（VideoCall）600kbps
resolution_960_540	2	宽高比 16:9；分辨率 960x540；建议码率（VideoCall）850kbps
resolution_960_720	3	宽高比 4:3；分辨率 960x720；建议码率（VideoCall）1000kbps
resolution_1280_720	4	宽高比 16:9；分辨率 1280x720；建议码率（VideoCall）1200kbps
resolution_1920_1080	5	宽高比 16:9；分辨率 1920x1080；建议码率（VideoCall）2000kbps

TUICallIOSOfflinePushType

iOS离线推送类型

类型	取值	描述
APNs	0	普通推送
VoIP	1	VoIP 推送

TUICamera

摄像头类型

类型	取值	描述
front	0	前置摄像头

back	1	后置摄像头

TUIAudioPlaybackDevice

音频播放路由

类型	取值	描述
speakerphone	0	扬声器
earpiece	1	耳麦

TUICallResultType

通话记录类型

类型	取值	描述
unknown	0	未知
missed	1	未接
incoming	2	接入
outgoing	3	拨出

版权所有：腾讯云计算（北京）有限责任公司

错误码表（TUICallKit）

最近更新时间：2023-11-17 15:12:11

通知用户在使用音视频通话过程中出现的警告和错误。

TUICallDefine 错误码

定义	取值	描述
ERROR_PACKAGE_NOT_PURCHASED	-1001	您当前未购买音视频通话能力套餐，请前往IM控制台 开通免费体验 或 加购正式版 。
ERROR_PACKAGE_NOT_SUPPORTED	-1002	您当前购买的音视频通话能力套餐不支持该功能，建议您前往IM控制台 升级套餐类型 。
ERROR_TIM_VERSION_OUTDATED	-1003	IM SDK 版本过低，请升级 IM SDK 版本 ⩾ 2.21.2（仅 web、小程序平台支持）。
ERROR_PERMISSION_DENIED	-1101	获取权限失败，当前未授权音/ 视频权限，请查看是否开启设备权限。
ERROR_INIT_FAIL	-1201	未调用 init，TUICallEngine API 使用需在 init 之后, 初始化失败。
ERROR_PARAM_INVALID	-1202	参数错误（参数格式、类型等错误）。
ERROR_REQUEST_REFUSED	-1203	当前状态不支持调用。例如： 需要在空闲状态才可使用的 API，在呼叫中/通话中时调用。
ERROR_REQUEST_REPEATED	-1204	当前方法正在执行中，请勿重复调用。
ERROR_SCENE_NOT_SUPPORTED	-1205	当前通话场景，不支持该功能。
ERROR_SIGNALING_SEND_FAIL	-1401	信令发送失败，具体失败原因可查看具体方法回调中的 errorMsg。
FAILURE_SWITCH_TO_AUDIO	60001	视频切语音失败（仅 web 平台支持）。
FAILURE_SWITCH_TO_VIDEO	60002	语音切视频失败（仅 web 平台支持）。
NOT_MICROPHONE_LIST	60003	没有可用的麦克风设备（仅 web 平台支持）。
NOT_CAMERA_LIST	60004	没有可用的摄像头设备（仅 web 平台支持）。
NOT_SUPPORT	60005	当前页面不具备 webRTC 能力（仅 web 平台支持）。
UNKNOWN	65535	未知错误（仅 web、小程序平台支持）。

IM 错误码

音视频通话使用了腾讯云 IM SDK 作为通信的基础服务，例如通话拨打信令、通话忙线信令等核心逻辑。常见错误码如下：

错误码	描述
6014	IM SDK 未登录，请先登录，成功回调之后重试，或者已被移下线，可使用 TIMManager getLoginUser 检查当前是否在线。

版权所有：腾讯云计算（北京）有限责任公司

6017	参数无效，请检查参数是否符合要求，具体可查看错误信息进一步定义哪个字段。
6206	UserSig 过期，请重新获取有效的 UserSig 后再重新登录，获取方法请参见 生成 UserSig 。
7013	套餐包不支持该接口的使用，请升级到旗舰版套餐。
8010	信令请求 ID 无效或已经被处理过。
8021	呼叫的用户还没有登录过 TIM，导致呼叫失败。

说明：
更多 IM SDK 错误码详见：IM 错误码 。

TRTC 错误码

音视频通话使用了腾讯云 TRTC SDK 作为音频和视频的基础服务，例如开关摄像头、开关麦克风等核心逻辑。常见错误码如下（Web TRTC SDK 错误码详见：TRTC 错误码 ）：

枚举	取值	描述
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_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。

说明：
更多 TRTC SDK 错误码详见：TRTC 错误码 。Web TRTC SDK 错误码详见：TRTC 错误码 。

升级方案（TUICallkit）
微信小程序

最近更新时间：2023-12-22 14:19:21

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI 降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信。
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置**页面，并在页面的右下角找到**开通腾讯实时音视频服务**功能区，单击**免费体验**即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 **前往加购** 即可进入购买页面。



注意：
IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单三步就可以升级完成，预计花费20分钟。

下载并导入 TUICallKit 组件

1. 下载 TUICallKit 组件。

```
npm i @tencentcloud/call-uikit-wechat@2.x
```

MacOS 端

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wechat/ ./TUICallKit
```

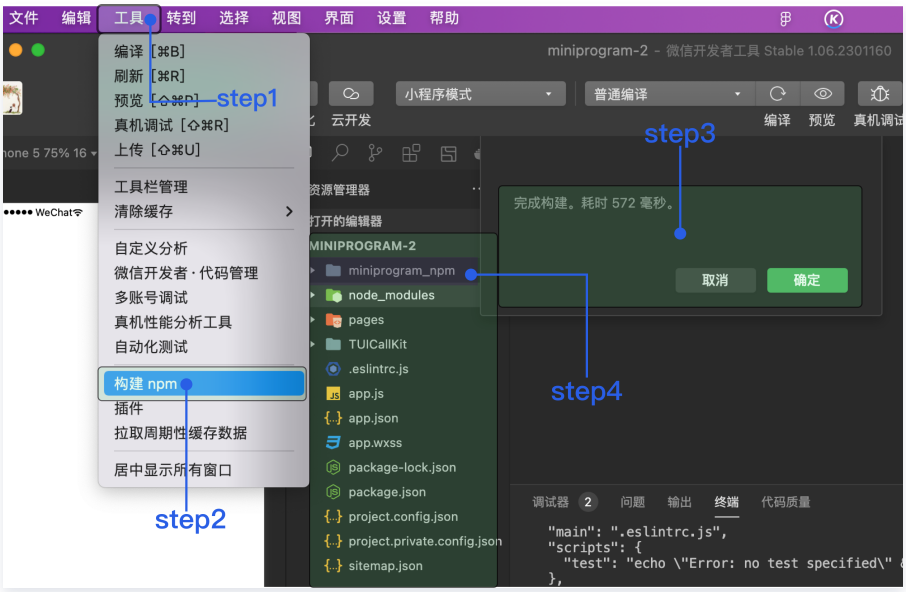
Windows 端

```
xcopy node_modules\@tencentcloud\call-uikit-wechat .\TUICallKit /i /e
```

2. 执行完以上命令后，您的目录下生成 TUICallKit 文件夹，其中包含有 TUICallKit 组件。目录如下：



3. 构建 npm，微信开发者工具**工具 > 构建 npm**，新增 miniprogram_npm 目录。目录如下：



修改 API 使用

```
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/TUICallService/index";
```

在完成上述步骤后，您的工程就无法正常编译了，需要将 TUICalling API 替换成新的TUICallKit API, 可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling 不再维护	TUICallKit	备注
初始化	TUICalling.init	TUICallService.init	/
设置用户的昵称	TUICalling.setUserNickname	TUICallService.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallService.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
发起 1v1 通话	TUICalling.call	TUICallService.call	/
发起群组通话	/	TUICallService.groupCall	/
自定义来电铃声	/	TUICallService.setCallingBell	/
设置日志等级	/	TUICallService.setLogLevel	/

主要是 setUserNickname、setUserAvatar API 整合到 setSelfInfo 实现。

Web

最近更新时间：2023-03-22 10:20:46

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TRTCCalling 的升级版本**，支持群组通话、AI 降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TRTCCalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信。
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的 **基本配置** 页面，并在页面的右下角找到 **开通腾讯实时音视频服务** 功能区，单击 **免费体验** 即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 **前往加购** 即可进入购买页面。



升级概览

TUICallKit 在设计之初就兼顾了 TRTCCalling 客户的升级诉求，仅需要简单几步就可以升级完成，预计花费 20 分钟；
您可以选择两种升级方式：

1. 升级到 TUICallKit Vue3 UI 开源组件：只需要编写几行代码就可以为您的 App 添加音视频通话功能，并且支持离线唤起能力。（推荐）
2. 升级到 TUICallEngine SDK：可直接从旧版本 TRTCCalling 迁移而来，UI 层可沿用已开发版本。

接下来将分别介绍两种方式的升级步骤。

TUICallKit Vue3 UI 组件升级步骤

TUICallKit Vue3 UI 提供带 UI 的开源组件，节省 90% 开发时间，快速上线音视频通话应用。

- 如果您想要体验并调试通话效果，请阅读 Demo 快速跑通。
- 如果您想把我们的功能直接嵌入到您的项目中，请阅读 快速接入 TUICallKit。
- 如果您想要修改 UI 界面，请阅读 TUICallKit 界面定制指引。

TUICallEngine SDK 1.x 无 UI 升级步骤

本部分将帮助您从 TRTCCalling 1.x 升级到 TUICallEngine 1.x。

1. 升级 SDK 包依赖

安装 TUICallEngine SDK 包，并将其他依赖更新到最新版本

```
npm install tuicall-engine-webrtc # 安装新版本 SDK
npm install tim-js-sdk@latest trtc-js-sdk@latest tsignaling@latest # 更新依赖到最新版本
npm uninstall trtc-calling-js # 移除旧版本 SDK
```

2. 替换 SDK 引入方式与创建实例方式

在完成上述步骤后，您的工程就无法正常编译了，接下里的步骤将手把手修改其中冲突的几个地方。
1. 首先，替换引入方式。

```
// 旧版本
import TRTCCalling from 'trtc-calling-js';

// 新版本
import { TUICallEngine } from "tuicall-engine-webrtc";
```

2. 修改创建实例方式，示例代码如下，若需详细接口请参考 createInstance() 。

```
// 旧版本
new TRTCCalling({ SDKAppID });

// 新版本
TUICallEngine.createInstance({ SDKAppID });
// TUICallEngine.destroyInstance(); // 如需要销毁实例，可调用此接口
```

3. 替换监听事件常量

监听事件常量地址由 `TRTCCalling.EVENT` 变为 `TUICallEvent` ，后者可以从包 `tuicall-engine-webrtc` 中引入。
此步骤推荐使用编辑器全局搜索一键替换。

```
// 旧版本
trtcCalling.on(TRTCCalling.EVENT.ERROR, this.handleError);
trtcCalling.on(TRTCCalling.EVENT.INVITED, this.handleNewInvitationReceived);
trtcCalling.on(TRTCCalling.EVENT.USER_ACCEPT, this.handleUserAccept);
... ( 包含其他监听 )

// 新版本
import { TUICallEvent } from "tuicall-engine-webrtc";
tuiCallEngine.on(TUICallEvent.ERROR, this.handleError); // tuiCallEngine 为新实例名称，沿用旧 trtcCalling 无妨
tuiCallEngine.on(TUICallEvent.INVITED, this.handleNewInvitationReceived);
tuiCallEngine.on(TUICallEvent.USER_ACCEPT, this.handleUserAccept);
... ( 包含其他监听 )
```

4. 替换通话类型常量

版权所有：腾讯云计算（北京）有限责任公司

若您之前未使用通话类型常量可跳过本步骤。
通话类型常量地址由 TRTCCalling.CALL_TYPE 变为 TUICallType ，后者可以从包 tuicall-engine-webrtc 中引入。
此步骤推荐使用编辑器全局搜索一键替换。

```
// 旧版本
this.TRTCCalling.CALL_TYPE.AUDIO_CALL // 语音类型
this.TRTCCalling.CALL_TYPE.VIDEO_CALL // 视频类型

// 新版本
import { TUICallType } from "tuicall-engine-webrtc";
TUICallType.AUDIO_CALL // 语音类型
TUICallType.VIDEO_CALL // 视频类型
```

5. 调整渲染远端视频的时机

在旧版本 TRTCCalling 中，推荐在收到 USER_ENTER 事件回调时，调用 startRemoteView() 渲染远端摄像头。
在新版本 TUICallEngine 中，推荐在收到 USER_VIDEO_AVAILABLE 事件回调时，调用 startRemoteView() 渲染远端摄像头。

```
// 旧版本
let handleUserEnter = async function(event) {
  await trtcCalling.startRemoteView({userID: 'user1', videoViewDomID: 'video_1'});
};
trtcCalling.on(TRTCCalling.EVENT.USER_ENTER, handleUserEnter);

// 新版本
let handleUserVideoAvailable = async function(event) {
  await tuiCallEngine.startRemoteView({userID: 'user1', videoViewDomID: 'video_1'});
};
tuiCallEngine.on(TUICallEvent.USER_VIDEO_AVAILABLE, handleUserVideoAvailable);
```

6. 其他关键接口变更

API 含义	旧版本 TRTCCalling SDK	新版本 TUICallEngine SDK	备注
切换通话类型	trtcCalling.switchToAudioCall()	tuiCallEngine.switchCallMediaType()	名称变更，通过参数选择切换类型
获取麦克风设备列表	trtcCalling.getMicrophones()	tuiCallEngine.getDeviceList()	接口合并，通过参数选择设备类型
获取摄像头设备列表	trtcCalling.getCameras()		
关闭/打开麦克风	trtcCalling.setMicMute()	tuiCallEngine.closeMicrophone() tuiCallEngine.openMicrophone()	接口拆分
设置用户昵称和头像	无	tuiCallEngine.setSelfInfo()	设置用户昵称和头像
开启/关闭 AI 降噪	无	tuiCallEngine.enableAIVoice()	4.12.1 版本以后支持，具体使用详情请参阅使用 AI 降噪

版权所有：腾讯云计算（北京）有限责任公司

7. 补充说明

- 由于 TUICallKit 需要搭配 IM 音视频通话能力套餐，您还可能遇到 `"The package you purchased does not support this ability."`，可以单击 IM 控制台，进入对应 SDKAppID 应用的 **基本配置** 页面，并在页面的右下角找到 **开通腾讯实时音视频服务** 功能区，单击 **免费体验** 即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 **前往加购** 即可进入购买页面。
- 升级完上述 API 之后，就可以正常使用 TUICallEngine SDK 了，如果遇到其他 API 接口调用问题，可参阅 SDK API 文档 。
- 欢迎加入 QQ 群：**646165204**，进行技术交流和反馈~

版权所有：腾讯云计算（北京）有限责任公司

Android

最近更新时间：2023-12-29 17:31:22

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信；
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案**功能区，选择**音视频通话**，然后单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



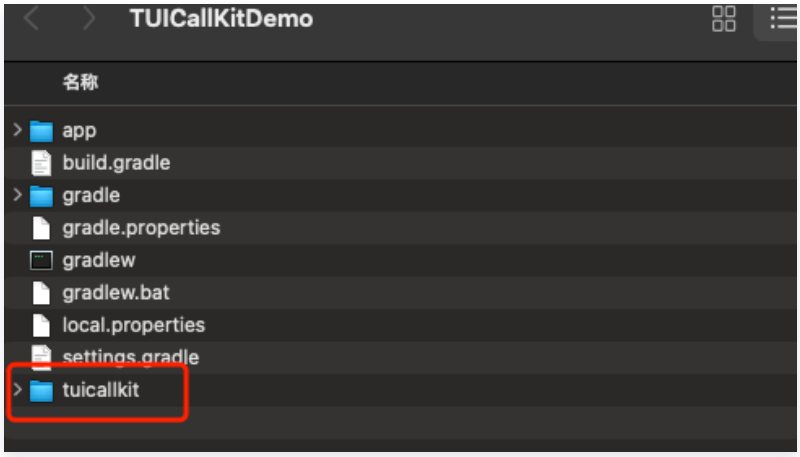
> **注意：**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单两步就可以升级完成，预计花费 20 分钟。

下载&替换 TUICallKit

1. 在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 tuicallkit 子目录到您当前工程中的 app 同级目录中，同时删除 tuicalling



目录，结果如下图：

2. 在工程根目录下找到 `setting.gradle` 文件，将 `include ':tuicalling'` 替换成 `include ':tuicallkit'`
3. 在 app 目录下找到 `build.gradle` 文件，将 `api project(':tuicalling')` 替换成 `api project(':tuicallkit')`

> **说明：**
> 如果您之前是配合 TUIChat、TUIContact 等IM 组件一起使用的话，执行完这个步骤即可正常使用 TUICallKit，TUICallKit 组件内部已经处理好兼容逻辑。

修改 API 使用

在完成上述步骤后，您的工程就无法正常编译了，需要将 TUICalling API 替换成新的 TUICallKit API, 可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling（包名：com.tencent.liteav.trtccalling）	TUICallKit（包名：com.tencent.qcloud.tuikit.tuicallkit）	备注
创建 TUICallKit 实例	TUICallingImpl.sharedInstance	TUICallKit.createInstance	替换引用、名称
设置用户的昵称	TUICalling.setUserNickname	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 `setSelfInfo` 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 `setSelfInfo` 接口
发起 1v1 通话	TUICalling.call	TUICallKit.call	详见 TUICalling.call 接口变更
发起群组通话	/	TUICallKit.groupCall	/
主动加入当前的群组通话中	/	TUICallKit.joinInGroupCall	/
设置自定义来电铃音	TUICalling.setCallingBell	TUICallKit.setCallingBell	替换引用、名称
开启/关闭静音模式	TUICalling.enableMuteMode	TUICallKit.enableMuteMode	替换引用、名称
开启/关闭悬浮窗功能	TUICalling.enableFloatWindow	TUICallKit.enableFloatWindow	替换引用、名称

设置监听事件	TUICalling.setCallingListener	TUICallEngine.addObserver	详见 TUICalling.setCallingListener 接口变更

如下是此次升级过程中，变化较大的两个API的适配方案：

TUICalling.call 接口变更

- TUICalling 代码示例

```
//原接口
void call(String[] userIDs, Type type);

//原调用方式
String[] userIDs = new String[1];
userIDs[0] = "123";
TUICallingImpl.sharedInstance(this).call(userIDs, TUICalling.Type.AUDIO);
```

- TUICallKit 代码示例

```
//新接口
public void call(String userId, TUICallDefine.MediaType callMediaType)

//新调用方式
TUICallKit.createInstance().call("mike", TUICallDefine.MediaType.Audio)
```

TUICalling.setCallingListener 接口变更

- TUICalling 代码示例

```
TUICallingImpl.sharedInstance(this).setCallingListener(new TUICalling.TUICallingListener() {
    @Override
    public boolean shouldShowOnCallView() {
        return true;
    }

    @Override
    public void onCallStart(String[] userIDs, TUICalling.Type type, TUICalling.Role role, View tuiCallingView)
{
    }

    @Override
    public void onCallEnd(String[] userIDs, TUICalling.Type type, TUICalling.Role role, long totalTime) {
    }

    @Override
    public void onCallEvent(TUICalling.Event event, TUICalling.Type type, TUICalling.Role role, String
message) {
    }
});
```

- TUICallKit 代码示例

```
  private TUICallObserver mTUICallObserver = new TUICallObserver() {
    @Override
    public void onError(int code, String msg) {
        super.onError(code, msg);
    }

    @Override
    public void onCallReceived(String callerId, List<String> calleeIdList, String groupId,
TUICallDefine.MediaType callMediaType) {
        super.onCallReceived(callerId, calleeIdList, groupId, callMediaType);
    }

    @Override
    public void onCallCancelled(String callerId) {
        super.onCallCancelled(callerId);
    }

    @Override
    public void onCallBegin(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole) {
        super.onCallBegin(roomId, callMediaType, callRole);
    }

    @Override
    public void onCallEnd(TUICommonDefine.RoomId roomId, TUICallDefine.MediaType callMediaType,
TUICallDefine.Role callRole, long totalTime) {
        super.onCallEnd(roomId, callMediaType, callRole, totalTime);
    }
    //按需加载其他事件监听
    ......
}

  TUICallEngine.createInstance(mContext).addObserver(mTUICallObserver);
```

升级完上述 API 后，即可正常使用 `TUICallKit` 组件。

iOS

最近更新时间：2023-12-29 17:31:22

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信；
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案** 功能区，选择**音视频通话**，然后单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



> **注意：**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单两步就可以升级完成，预计花费 20 分钟。

替换 TUICallKit

1. 删除您的工程 `Podfile` 文件同一级目录下的 `TUICalling` 文件夹。
2. 删除您的 `Podfile` 文件中的以下依赖。

```
# :path => "指向TUICalling.podspec的相对路径"
pod 'TUICalling', :path => "TUICalling/TUICalling.podspec", :subspecs => ["TRTC"]
```

3. 在您的 `Podfile` 文件中添加以下依赖。

```
pod 'TUICallKit'
```

4. 执行以下命令，安装组件。

版权所有：腾讯云计算（北京）有限责任公司

```
pod install
```

如果无法安装 TUICallKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

> **说明：**
> 如果您之前是配合 TUIChat、TUIContact 等IM 组件一起使用的话，执行完这个步骤即可正常使用 TUICallKit，TUICallKit 组件内部已经处理好兼容逻辑。

修改 API 使用

在完成上述步骤后，您的工程就无法正常编译了，需要将 TUICalling API 替换成新的TUICallKit API, 可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling	TUICallKit	备注
创建 TUICallKit 实例	TUICallingImpl.sharedInstance	TUICallKit.createInstance	替换引用、名称
设置用户的昵称	TUICalling.setUserNickname	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
发起 1v1 通话	TUICalling.call	TUICallKit.call	详见 TUICalling.call 接口变更
发起群组通话	/	TUICallKit. groupCall	/
主动加入当前的群组通话中	/	TUICallKit.joinInGroupCall	/
设置自定义来电铃音	TUICalling.setCallingBell	TUICallKit.setCallingBell	替换引用、名称
开启/关闭静音模式	TUICalling.enableMuteMode	TUICallKit.enableMuteMode	替换引用、名称
开启/关闭悬浮窗功能	TUICalling.enableFloatWindow	TUICallKit.enableFloatWindow	替换引用、名称
设置监听事件	TUICalling.setCallingListener	TUICallEngine.addObserver	详见 TUICalling.setCallingListener 接口变

如下是此次升级过程中，变化较大的两个API的适配方案：

TUICalling.call 接口变更

- **TUICalling 代码示例：**

```
// 原接口
[[TUICalling shareInstance] call:@[@"mike"] type:TUICallingTypeVideo];
```

- **TUICallKit 代码示例：**

```
// 新调用方式
[[TUICallKit createInstance] call:@"mike" callMediaType:TUICallMediaTypeVideo];
```

TUICalling.setCallingListener 接口变更

- **TUICalling 代码示例**

```
[[TUICalling shareInstance] setCallingListener:self];

- (BOOL)shouldShowOnCallView {
    return YES;
}
- (void)callStart:(nonnull NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)role
viewController:(UIViewController * _Nullable)viewController {

}
- (void)callEnd:(nonnull NSArray<NSString *> *)userIDs type:(TUICallingType)type role:(TUICallingRole)role
totalTime:(float)totalTime {

}
- (void)onCallEvent:(TUICallingEvent)event type:(TUICallingType)type role:(TUICallingRole)role message:
(nonnull NSString *)message {

}
```

- **TUICallKit 代码示例**

```
[[TUICallEngine createInstance] addObserver:self];

- (void)onCallBegin:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole {

}
- (void)onCallEnd:(TUIRoomId *)roomId callMediaType:(TUICallMediaType)callMediaType callRole:
(TUICallRole)callRole totalTime:(float)totalTime {

}
- (void)onUserNetworkQualityChanged:(NSArray<TUINetworkQualityInfo *> *)networkQualityList {

}
// 按需加载其他事件监听
……
```

升级完上述API后，您可以正常使用 `TUICallKit` 组件啦~

uni-app（客户端）

最近更新时间：2023-12-29 17:31:22

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信；
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案** 功能区，选择**音视频通话**，然后单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



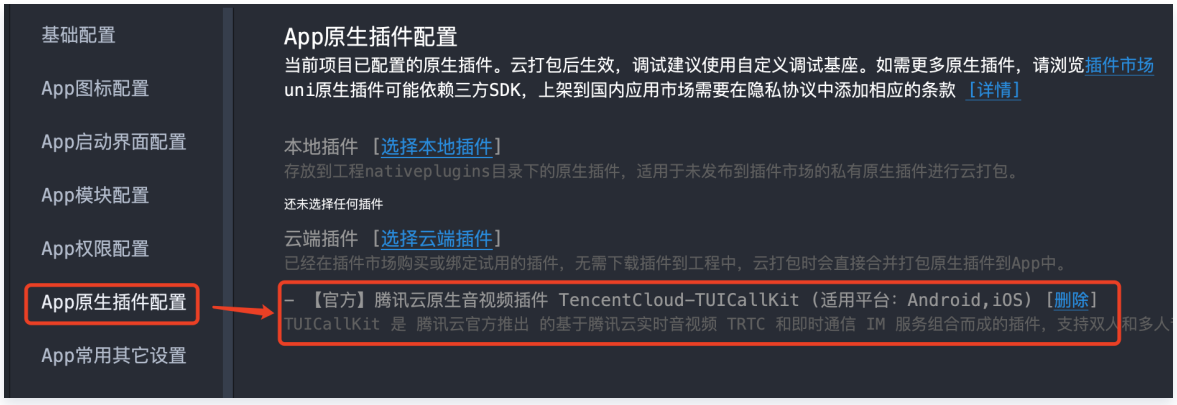
> **注意：**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单两步就可以升级完成，预计花费 20 分钟；

云端&本地替换为 TUICallKit 插件

1. 打基座时，云端插件更换为 TUICallKit 插件，具体如下：



2. 如果打基座使用本地插件，需要先下载 TUICallKit 插件 到本地，然后放到 nativeplugins 目录下，打包时选择 “本地插件”。具体如下：



修改 API 使用

在完成上述步骤后，您的工程部分接口可能使用报错，需要将 TUICalling API 替换成新的 TUICallKit API，可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling 插件不再维护	TUICallKit 插件	备注
登录	TUICalling.login	TUICallKit.login	原来 login 入参 sdkAppId，现在改为：SDKAppID。
登出	TUICalling.logout	TUICallKit.logout	/
设置用户的昵称	TUICalling.setUserNickname	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
发起 1v1 通话	TUICalling.call	TUICallKit.call	/
发起群组通话	/	TUICallKit.groupCall	/
主动加入当前的群组通话中	/	TUICallKit.joinInGroupCall	/
设置自定义来电铃音	TUICalling.setCallingBell	TUICallKit.setCallingBell	/
开启/关闭静音模式	TUICalling.enableMuteMode	TUICallKit.enableMuteMode	/
开启/关闭悬浮窗功能	TUICalling.enableFloatWindow	TUICallKit.enableFloatWindow	/

主要是 setUserNickname、setUserAvatar API 整合到 setSelfInfo 实现。

Flutter

最近更新时间：2023-12-29 17:31:22

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信；
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置页面**，并在页面的**右下角**找到**含 UI 低代码场景方案** 功能区，选择**音视频通话**，然后单击**免费体验**即可**开通 TUICallKit 的 7 天免费试用服务**。如果需要正式应用上线，可以单击 购买正式版 即可进入购买页面。



> **注意：**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单两步就可以升级完成，预计花费 20 分钟；

1、升级依赖为TUICallKit

在您的工程中完成对TUICallKit的依赖升级，对您的工程的配置文件pubspec.yaml进行设置：

```
dependencies:
  #删除旧的依赖tim_ui_kit_calling_plugin，并添加新的依赖tencent_calls_uikit:
  tencent_calls_uikit:
```

设置完成后执行 `flutter pub get` 命令。

> **说明：**
> 如果您之前是配合 TUIChat、TUIContact 等IM 组件一起使用的话，执行完这个步骤即可正常使用 TUICallKit，TUICallKit 组件内部已经处理好兼容逻辑。

2、修改 API 使用

在完成上述步骤后，您的工程就无法正常编译了，需要将 TUICalling API 替换成新的 TUICallKit API，可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling(tim_ui_kit_calling_plugin)	TUICallKit(tencent_calls_uikit)	备注
创建 TUICallKit 实例	TUICalling.sharedInstance	TUICallKit.instance	替换引用、名称
设置用户的昵称	TUICalling.setUserNickname	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallKit.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
发起 1v1 通话	TUICalling.call	TUICallKit.call	详见 TUICalling.call 接口变更
发起群组通话	/	TUICallKit.groupCall	/
主动加入当前的群组通话中	/	TUICallKit.joinInGroupCall	/
设置自定义来电铃音	TUICalling.setCallingBell	TUICallKit.setCallingBell	替换引用、名称
开启/关闭静音模式	TUICalling.enableMuteMode	TUICallKit.enableMuteMode	替换引用、名称
开启/关闭悬浮窗功能	TUICalling.enableFloatWindow	TUICallKit.enableFloatWindow	替换引用、名称
设置监听事件	TUICalling.registerListener	TUICallEngine.addObserver	详见 TUICalling.registerListener 接口变更

如下是此次升级过程中，变化较大的两个 API 的适配方案：

TUICalling.call 接口变更

- **TUICalling 代码示例**

```
//原接口
Future<void> call(String userId, CallingScenes type, [OfflinePushInfo? offlinePushInfo]);
```

- **TUICallKit 代码示例**

```
//新接口
Future<void> call(String userId, TUICallMediaType callMediaType, [TUICallParams? params]);

//新调用方式
TUIOfflinePushInfo offlinePushInfo = TUIOfflinePushInfo();
  offlinePushInfo.title = "Flutter TUICallKit";
  offlinePushInfo.desc = "This is an incoming call from Flutter TUICallkit";
   TUICallParams params = TUICallParams(offlinePushInfo: offlinePushInfo);
```

版权所有：腾讯云计算（北京）有限责任公司

```
TUICallKit.instance.call(callUserId, TUICallMediaType.audio, params);
```

TUICallKit.setCallingListener 接口变更

- **TUICalling 代码示例**

```
TUICalling.sharedInstance().registerListener(TUICallingListener(
        onInvited: (params) {

        }

        onCallingCancel: () {

        }
        ......
));
```

- **TUICallKit 代码示例**

```
TUICallEngine.instance.addObserver(TUICallObserver(
    onError:(int code, String message) {

    },
    onCallCancelled: (String callerId) {

    },
    onCallBegin:(TUIRoomId roomId, TUICallMediaType callMediaType, TUICallRole callRole) {

    },
    ......
));
```

升级完上述 API 后，即可正常使用 `TUICallKit` 组件。

uni-app（小程序）

最近更新时间：2023-12-22 14:19:21

升级说明

TUICallKit 是腾讯云推出一款新的音视频通话 UI 组件，**是 TUICalling 的升级版本**，支持群组通话、AI 降噪等更多功能特性、支持全平台间互通呼叫，功能更加稳定，欢迎您使用新的 TUICallKit 组件，在升级前您需要了解到：

- **TUICalling 和 TUICallKit 支持相互呼叫**，升级前后，请保持 SDKAppID 不变，否则会影响相互通信。
- TUICallKit 需要搭配 IM 音视频通话能力套餐，可以单击 IM 控制台，进入对应 SDKAppID 应用的**基本配置**页面，并在页面的右下角找到**开通腾讯实时音视频服务**功能区，单击**免费体验**即可开通 TUICallKit 的 7 天免费试用服务。如果需要正式应用上线，可以单击 **前往加购** 即可进入购买页面。



> **注意：**
> IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

升级步骤

> **注意：**

> @tencentcloud/call-uikit-wechat ≥ V3.0.4 采用新架构进行重构，不再使用微信原生进行开发，而是采用 Vue（支持 Vue2.6、Vue3）开发的组件。本文档使用主包接入的方式集成 TUICallKit，如果您想使用分包集成的方式请参考 uni-app（小程序）。

TUICallKit 在设计之初就兼顾了 TUICalling 客户的升级诉求，仅需要简单三步就可以升级完成，预计花费20分钟。

下载并导入 TUICallKit 组件

MacOS 端

```
npm i @tencentcloud/call-uikit-wechat
```

```
mkdir -p ./TUICallKit && cp -r node_modules/@tencentcloud/call-uikit-wechat/ ./TUICallKit
```

Windows 端

```
npm i @tencentcloud/call-uikit-wechat
```

```
xcopy node_modules\@tencentcloud\call-uikit-wechat\ .\TUICallKit /i /e
```

修改 API 使用

```
<template>
  <TUICallKit></TUICallKit>
</template>
<script setup>
// 导入 TUICallKitServer 模块，使您的应用具有全局呼叫的能力
import { TUICallKitServer } from "../../TUICallKit/TUICallService/index";
import TUICallKit from "../../TUICallKit/src/Components/TUICallKit";
</script>
```

在完成上述步骤后，您的工程就无法正常编译了，需要将 TUICalling API 替换成新的TUICallKit API, 可以参照如下 API 对比信息，搜索替换即可。

API 含义	TUICalling 不再维护	TUICallKit	备注
初始化	TUICalling.init	TUICallService.init	/
设置用户的昵称	TUICalling.setUserNickname	TUICallService.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口
设置用户的头像	TUICalling.setUserAvatar	TUICallService.setSelfInfo	设置头像、昵称的接口整合到 setSelfInfo 接口

发起 1v1 通话	TUICalling.call	TUICallService.call	/
发起群组通话	/	TUICallService.groupCall	/
自定义来电铃声	/	TUICallService.setCallingBell	/
设置日志等级	/	TUICallService.setLogLevel	/

主要是 `setUserNickname`、`setUserAvatar` API 整合到 `setSelfInfo` 实现。

发布日志（TUICallKit）
uni-app（小程序）

最近更新时间：2024-03-06 14:51:31

注意：

1. @tencentcloud/call-uikit-wechat ≥ V3.0.4 采用新架构进行重构，不再使用微信原生进行开发，而是采用 Vue（支持 Vue2.6、Vue3）开发的组件。**目前暂时无法在微信开发者工具中引入组件直接使用，需要在 uniapp 项目中打包小程序使用。**
2. 如果想使用原生小程序 @tencentcloud/call-uikit-wechat 组件，需要低于 V3.0.4。低版本的组件会针对发现的 bug 继续升级更新维护。

Version 3.2.0 @2024.02.23

新增

增加默认的离线推送参数。

修复

修复群组通话没有昵称问题。

Version 3.1.9 @2024.01.30

修复

- 修复群组通话没有展示用户信息问题。
- 修复选人组件在没有可选成员时，“确认”按钮仍然可点击的问题。
- 修复小程序在切换到后台后通话计时暂停，导致通话时长不准确的问题。

Version 3.1.8 @2024.01.19

修复

修复选人组件样式对页面的影响。

Version 3.1.7 @2024.01.12

修复

- 修复远端关闭摄像头和麦克风之后的报错问题。
- 修复群组通话中，进房用户没有流时的报错问题。
- 修复悬浮窗偏移及层级样式问题。

Version 3.1.6 @2023.12.29

优化

- 优化群组通话下的提示信息。
- 优化昵称过长时的显示问题。

修复

- 修复小程序 setSelfInfo 接口传参问题。
- 修复 destroyed 存在的问题。
- 修复不同呼叫场景下悬浮窗里的挂断问题。
- 修复主叫呼叫状态下显示远端问题。
- 修复 pc 上填充不全的样式问题。

Version 3.1.4 @2023.12.01

新增

融合到 Chat 中增加 isFromChat 上报。

修复

修复按钮 loading 下可点击的问题。

Version 3.1.3 @2023.11.17

新增

接口增加参数校验。

Version 3.1.2 @2023.11.03

新增

- 新增邀请他人 inviteUser 功能。
- 新增中途加入 joinInGroupCall 功能。
- 新增被叫来电铃声静音 enableMuteMode 功能。

修复

修复远端流麦克风状态展示不对问题。

Version 3.1.1 @2023.10.27

修复

修复部分 iOS 机型上按钮被隐藏问题。

Version 3.1.0 @2023.10.20

新增

- 增加悬浮窗功能。
- 增加 enableFloatWindow 接口，用于开启/关闭悬浮窗功能。
- 增加呼叫黑名单用户失败提示信息。
- 增加对日语的支持。

优化

视频通话下，大屏默认显示远端用户。

Version 3.0.8 @2023.10.10

新增

增加版本号、框架等信息上报。

Version 3.0.7 @2023.10.08

新增

增加桌面端视频通话下，通话时长展示。

优化

- 优化桌面端上视频流预览圆角黑边问题。
- 优化远端流用户信息显示优先级：备注 > 昵称 > userId。
- 优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 3.0.6 @2023.09.19

修复

修复融合到 TUIKit 中的消息上屏问题。

Version 3.0.5 @2023.09.15

优化

优化 TUICallKit 之间的相互引用，避免在 uniapp 打包小程序出现的栈溢出问题。

新增

- 新增无套餐的弹框提示。
- 增加全局监听功能，具体：详见 。

修复

- 修复 setCallingBell 被叫铃声被主叫铃声覆盖，导致铃声重复问题。
- 修复移动端上的样式问题。

Version 3.0.4 @2023-09-01

Breaking Change

升级底层依赖 tuicall-engine-wx 至 ^2.0.0，不再支持 tim-wx-sdk 创建 tim 实例，如需创建 tim 实例，请使用 @tencentcloud/chat 创建。

新增

支持在 uniapp 项目中以分包的方式集成。

Version 1.4.4 @2023-07-28

新增

新增 init 错误抛出处理。

Version 1.4.3 @2023-07-14

修复

调整事件监听时机，修复 SDK_READY 未抛出的问题。

Version 1.4.2 @2023-06-09

修复

- 修复全局监听卸载不彻底的问题。
- 修复全局监听本地预览取值不当的问题。

Version 1.4.1 @2023-05-25

新增

新增 call、groupCall 错误抛出处理。

Version 1.4.0 @2023-05-17

新增

新增在 tabBar 页面接入时的兼容性处理。

优化

- 优化在页面跳转情况下的初始化以及卸载逻辑。
- 优化全局监听逻辑，防止重复监听。

Version 1.3.9 @2023-05-11

新增

- 新增 call、groupCall 支持自定义数字房间号。
- 新增 call、groupCall 允许用户自定义 userData 数据。

修复

修复全局监听模式下，视频通话被叫预览黑屏导致无法接听的问题。

Version 1.3.8 @2023-04-27

> ⚠ **注意：**
> 需要依赖 tuicall-engine-wx 的版本 ⩾ v1.4.7。

新增

新增通话页面出现时的动画效果。

优化

- 优化通话界面唤起的时间。
- 优化用户异常退出的逻辑，如右滑退出以及返回退出的情况。

修复

修复用户重复点击，导致接口被重复调用的问题。以下接口增加节流处理： call，groupCall，accept，reject。

Version 1.3.7 @2023-04-14

新增

新增 设置日志级别接口 。

修复

修复铃声路径错误问题。

Version 1.3.6 @2023-03-31

新增

新增 index.js 入口文件，解决 npm 构建时的警告问题。

优化

- 优化用户没有 live-pusher 推流标签使用权限时，给予弹窗提示。
- 优化针对在微信开发者工具运行时，给予弹窗提示。

Version 1.3.5 @2023-03-24

新特性

支持自定义铃声。

修复

优化 reset 函数在生命周期中的触发时机，修复全局监听下被叫方的白屏问题。

Version 1.3.4 @2023-03-16

修复

全局监听下的视频通话，被叫方使用 ios 设备预览黑屏问题。

优化

优化常见错误提示，方便客户快速定位问题。

Version 1.3.3 @2023-03-02

优化

- 弹窗昵称显示优先级（昵称 > userID）。
- 打印显示 TUICallKit 版本号。

Version 1.3.2 @2023-02-17

修复

打包路径错误。

Version 1.3.1 @2023-02-17

修复

修复远端关闭麦克风，没有及时更新远端流导致后续调用音量变化接口报错。

Version 1.3.0 @2023-02-13

优化

优化依赖引入，组件包体积由 900KB 降低至 110KB 左右。

Version 1.2.7 @2023-02-03

修复

去除 aegis.js 文件。

Version 1.2.6 @2023-01-03

修复

未及时回收 wx.$TUICallEngine 变量引发的 init 报错。

Version 1.2.5 @2023-01-31

新特性

支持 uniapp 小程序全局监听，具体 详见。

修复

升级 tsignaling-wx，解决和客户端的互通问题。

Version 1.2.4 @2023-01-12

修复

- 打包路径错误。
- callManager 兼容 sdkAppID 与 SDKAppID 传参。
- callkit 生命周期初始化时 移除 wx.CallManager 监听报错问题。

Version 1.2.3 @2023-01-10

修复

TUICallKit 样式布局错误。

Version 1.2.1 @2023-01-09

修复

打包问题引起的 globalCall 页面引用错误。

Version 1.2.0 @2023-01-06

新特性

支持全局监听

- 主包方案详见 【TUICallKit】小程序全局监听（主包方案）。
- 分包方案详见 【TUICallKit】小程序全局监听（分包方案）。

Version 1.1.4 @2022-12-23

修复

- init 方法支持外部传入 tim。

- 修复 uni-app 打包小程序存在的栈溢出问题。

Version 1.1.3 @2022-12-14

新增

新增设备权限弹出窗口。

优化

群通话中的昵称显示优先级。

> **说明：**
> 群通话显示优先级为 **昵称 > userID**。

修复

群通话中 player 推流过程中 trtc-wx 报错问题。

Version 1.1.2 @2022-11-30

优化

优化接入流程。

Version 1.1.1 @2022-11-30

优化

兼容 chat，支持 TUICallKit 在 init 时传值。

Version 1.1.0 @2022.11.16

> **注意：**
> 升级至该版本需注意：升级指引 。

优化

- 提高多人通话稳定性
- 抛出事件的数据结构调整，具体请参见 TUICallEvent 。

Web

最近更新时间：2024-03-06 14:51:31

> **说明：**
> TUICallKit Vue3 Github 版本地址：Github TUICallKit Web 。

Version 3.2.0 @2024.02.23

新增

增加默认的离线推送参数。

修复

修复群组通话没有昵称问题。

Version 3.1.9 @2024.01.30

修复

- 修复群组通话没有展示用户信息问题。
- 修复选人组件在没有可选成员时，“确认”按钮仍然可点击的问题。
- 修复通话中关闭麦克风，再次通话不上行音频流问题（升级 trtc-cloud-js-sdk 到 v2.2.7+）。

Version 3.1.8 @2024.01.19

修复

修复选人组件样式对页面的影响。

Version 3.1.7 @2024.01.12

修复

- 增加接口的重试机制，修复播放时因找不到 dom 节点导致播放失败问题。
- 修复 PC 上设备列表选择样式问题。

Version 3.1.6 @2023.12.29

优化

- 优化群组通话下的提示信息。
- 优化昵称过长时的显示问题。

修复

- 修复语音通话请求摄像头权限问题。
- 修复 destroyed 存在的问题。
- 修复不同呼叫场景下悬浮窗里的挂断问题。
- 修复主叫呼叫状态下显示远端问题。
- 修复 pc 上填充不全的样式问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 3.1.5 @2023.12.15

新增

优化获取设备权限的时机，初始化时不再获取设备权限，使用 call 时才获取。

修复

修复 @tencentcloud/call-uikit-vue2 、@tencentcloud/call-uikit-vue2.6 组件没有声明文件问题。

Version 3.1.4 @2023.12.01

新增

融合到 Chat 中增加 isFromChat 上报。

修复

修复按钮 loading 下可点击的问题。

Version 3.1.3 @2023.11.17

新增

接口增加参数校验。

Version 3.1.2 @2023.11.03

新增

- 新增邀请他人 inviteUser 功能。
- 新增中途加入 joinInGroupCall 功能。
- 新增被叫来电铃声静音接口 enableMuteMode 功能。

修复

修复远端流麦克风状态展示不对问题。

Version 3.1.0 @2023.10.20

新增

- 增加悬浮窗功能。
- 增加 enableFloatWindow 接口，用于开启/关闭悬浮窗功能。
- 桌面端支持摄像头、麦克风切换设备。
- 增加呼叫黑名单用户失败提示信息。
- 增加对日语的支持。

优化

视频通话下，大屏默认显示远端用户。

Version 3.0.8 @2023.10.10

新增

增加版本号、框架等信息上报。

Version 3.0.7 @2023.10.08

新增

增加桌面端视频通话下，通话时长展示。

优化

- 优化桌面端上视频流预览圆角黑边问题。
- 优化远端流用户信息显示优先级：备注 > 昵称 > userId。
- 优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 3.0.6 @2023.09.19

修复

修复融合到 TUIKit 中的消息上屏问题。

Version 3.0.5 @2023.09.15

优化

优化 TUICallKit 之间的相互引用，避免在 uniapp 打包小程序出现的栈溢出问题。

新增

新增桌面端设备无权限时的提示，引导客户如何对设备进行授权。

修复

- 修复 setCallingBell 被叫铃声被主叫铃声覆盖，导致铃声重复问题。
- 修复移动端上的样式问题。

Version 3.0.4 @2023.09.01

修复

- 修复 setCallingBell 作用对象为来电铃声（被叫铃声）。
- 修复 destroyed 报错问题。
- 修复错误弹框提示没有中英文。
- 修复 1v1 通话关闭摄像头后, 无法切换大小屏。

Version 3.0.3 @2023.8.25

新增

新增 @tencentcloud/call-uikit-vue2.6 适配 Vue2.6 版本。

优化

- 优化组件默认语言，调整为系统默认的语言。
- 优化打印的日志信息。
- 优化 tuicall-engine-webrtc 抛出错误的信息提示。
- 优化组件销毁后，对所用资源的清理。

修复

- 修复 videoDisplayMode 、videoResolution 挂断后再次呼叫时，未生效问题。
- 修复通话中，statusChanged 未抛出问题。
- 修复 init 多次调用问题。
- 修复通话中，关闭摄像头无法切换大小屏问题。

Version 3.0.2 @2023.8.14

修复

- 修复 H5 端被叫组件的样式问题。
- 修复切换小窗后，再次通话出现的样式问题。

Version 3.0.1 @2023.8.8

修复

修复群通话主叫方本地预览失败的问题，修改组件层对数据层默认读取方式。

Version 3.0.0 @2023.8.4

Breaking Change

升级底层依赖 tuicall-engine-webrtc 至 ^2.0.0，不再支持 tim-js-sdk 创建 tim 实例，如需创建 tim 实例，请使用 @tencentcloud/chat 创建。

新增

新增自定义铃声功能 setCallingBell 。

Version 2.4.2 @2023.11.03

新增

- 新增邀请他人 inviteUser 功能。
- 新增中途加人 joinInGroupCall 功能。
- 新增被叫来电铃声静音接口 enableMuteMode 功能。

修复

修复远端流麦克风状态展示不对问题。

Version 2.4.0 @2023.10.20

新增

- 增加悬浮窗功能。
- 增加 enableFloatWindow 接口，用于开启/关闭悬浮窗功能。
- 桌面端支持摄像头、麦克风切换设备。
- 增加呼叫黑名单用户失败提示信息。
- 增加对日语的支持。

优化

视频通话下，大屏默认显示远端用户。

版权所有：腾讯云计算（北京）有限责任公司

Version 2.3.9 @2023.10.10

新增

增加版本号、框架等信息上报。

Version 2.3.8 @2023.10.08

新增

增加桌面端视频通话下，通话时长展示。

优化

- 优化桌面端上视频流预览圆角黑边问题。
- 优化远端流用户信息显示优先级：备注 > 昵称 > userId。
- 优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 2.3.6 @2023.09.15

优化

优化 TUICallKit 之间的相互引用，避免在 uniapp 打包小程序出现的栈溢出问题。

新增

新增桌面端设备无权限时的提示，引导客户如何对设备进行授权。

修复

- 修复 setCallingBell 被叫铃声被主叫铃声覆盖，导致铃声重复问题。
- 修复移动端上的样式问题。

Version 2.3.5 @2023.9.5

修复

修复进房前摄像头、麦克风按钮默认打开的问题。

Version 2.3.4 @2023.9.1

新增

新增视频通话接听前可以关闭或者开启摄像头功能。

修复

- 修复 1v1 通话关闭摄像头后, 无法切换大小屏问题。
- 修复从视频通话切换到语音通话 statusChanged 未抛出问题。

Version 2.3.3 @2023.8.22

修复

- 修复 videoDisplayMode 、videoResolution 挂断后再次呼叫时，未生效问题。
- 修复通话中，statusChanged 未抛出问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 2.3.2 @2023.7.26

Breaking Change

- 移除悬浮窗 TUICallKitMini 组件，合并到 TUICallKit 组件中。
- 抛出的 @kicked-out 事件调整为绑定回调 :kickedOut 的方式。
- 抛出的 @status-changed 事件调整为绑定回调 :statusChanged 的方式。

新增

- 新增通话页面出现时的动画效果。
- 新增 H5 上群组通话布局。

优化

- 优化通话中的问题提示信息、提示方式。
- 优化在 H5 页面的支持、交互。
- 优化通话界面唤起的时间。
- 优化 @tencentcloud/call-uikit-vue 包目录结构。

修复

- 修复接通后立刻挂断等边界操作下的通话问题。
- 修复 H5 在部分机型、浏览器上的样式问题。
- 修复重复点击引起的通话异常问题。

Version 2.2.1 @2023.7.7

新增

@tencentcloud/call-uikit-vue2 新增 Vue 版本的检测及提示。

修复

- 修复来电页面重复点击“接听”按钮导致接听失败。

Version 2.2.0 @2023.6.30

新增

- call 、groupCall 支持自定义数字房间号 roomID 参数。
- call 、groupCall 支持自定义扩展字段 userData 参数（用于在邀请信令中增加扩展信息）。
- 新增 setSelfInfo 接口，支持用户设置昵称和头像。

Version 2.1.0 @2023.4.14

新增

- 在 H5 模式语音通话下，呼叫时支持显示对方昵称。
- 发起通话失败时，会在呼叫页面显示“发起通话失败”。
- 接听通话失败时，会在来电页面显示“接听失败”。
- 支持监听当前用户是否被踢出（如被挤出登录），详见 TUICallKit 方法 - @kicked-out。
- 支持监听 TUICallKit 通话状态，详见 TUICallKit 方法 - @status-changed。

版权所有：腾讯云计算（北京）有限责任公司

- 支持业务侧代码控制通话的接听、取消和挂断，详见 更多特性 – 通过接口设置自动接听 。
- Vue2 版本新增 TypeScript 类型声明文件，在 TypeScript 项目中可以正常编译类型。

修复

- 修复在初始化组件时，控制台出现的更新个人资料接口的警告。
- 修复在 H5 模式下，被叫接听按钮出现背景图片错位的问题。

接口变更

`TUICallKitServer.destroy()` 增加调用限制，仅支持在未通话状态时调用。

Version 2.0.1 @2023.03.31

新增

- 优化 1v1 通话与多人通话视频的渲染逻辑，提升性能与稳定性。
- 优化 UI 展示，支持在 `TUICallKitServer.call()` 执行过程中显示对应 UI，即可以实现点击通话按钮立刻展示 `<TUICallKit/>` UI 组件。

修复

- 修复多人通话中昵称显示错误的问题。
- 修复 CSS 未限制生效范围导致污染全局样式的问题。

Version 2.0.0 @2023.03.21

新增

- 支持从 npm 引入已打包的 CallKit 文件。
- 支持 JavaScript 版本的 Vue 项目。
- 支持 Vue2 所有版本的 Vue 项目，适用于 Vue2 的 npm 包：call-uikit-vue2。

修复

修复无摄像头设备或权限导致通话发起异常的问题。

Version 1.4.2 @2023.03.03

新增

- 支持设置通话分辨率，详见 API 文档。
- 支持修改画面显示模式，详见 API 文档。
- 优化了接入步骤。
- 优化了错误抛出。

Version 1.4.1 @2023.02.13

新增

- 优化了本地摄像头的预览逻辑。
- 优化了远端视频流的渲染逻辑。

Version 1.4.0 @2023.01.06

新增

- 支持 Vue2.7+ 项目引入。
- 通话界面默认显示昵称，设置昵称请参考 TIM#updateMyProfile。

Version 1.3.3 @2022.12.27

新增

- 在 Basic Demo 中拨打电话时，新增了通话列表的空值检测。
- 在拨打电话时，Basic demo 中增加了 loading 图标。
- 优化了 Basic Demo 中设备检测的逻辑，手动跳过后不再主动弹出。
- 优化了组件图标的引用方式。
- 修改了默认的包管理工具为 npm。
- 优化了视频的渲染方式，减少了重复渲染的次数。

修复

修复了由于 vue-CLI 过时依赖导致的 Basic Demo 报错。

Version 1.3.2 @2022.12.07

新增

- 支持语言切换，接口详情可见 setLanguage。
- 优化 Basic Demo 设备检测逻辑，手动跳过后不会再主动弹出。

修复

修复引入 `defineProps` 导致的警告。

Version 1.3.1 @2022.11.29

> ⚠ **注意：**
> 本版本依赖 SDK tuicall-engine-webrtc@1.2.1 版本，请及时更新。

新增

- 优化样式细节。
- 支持未接听对方通话时，监听对方修改通话类型。
- basic demo 增加设备检测功能。

修复

修复在挂断电话时内部逻辑导致的报错。

Version 1.3.0 @2022.11.14

新增

- 支持手机 H5 使用时，自动切换为竖屏样式。
- 支持在拨打电话时预览本地摄像头。
- basic demo 增加拨打电话前的设备检测。

版权所有：腾讯云计算（北京）有限责任公司

修复

- 修复调用 TUICallKitServer.destroyed() 之后 tim 实例未完全退出的问题。
- 修复对方忙线时显示无应答的问题。
- 修复 vite 环境下 typescript 类型未成功打包的问题。

接口变更

主动调用 TUICallKitServer.call() 或 TUICallKitServer.groupCall() 时，如果遇到报错，不会再调用 beforeCalling 回调。请直接使用 try catch 捕获错误。

Version 1.2.0 @2022.11.03

新增

适配新版本 TUICallEngine SDK。

Version 1.1.0 @2022.10.21

新增

- 通话过程中，通话页面可以全屏。
- 通话过程中，可以使用 `<TUICallKitMini/>` 进行最小化。

修复

修复已知问题，提升稳定性。

Version 1.0.3 @2022.10.14

新增

basic demo 增加快捷复制 UserID、一键打开新窗口。

Version 1.0.2 @2022.09.30

新增

优化接入文档，增加演示图片和详细指引。

修复

- 修复首次进入房间时，设备状态位失效的问题。
- 修复在 webpack 工具打包时，图标偶现加载失败的问题。
- 修复已知样式问题。

Version 1.0.1 @2022.09.26

新增

在拨打电话时，隐藏对方麦克风图标。

修复

修复 basic demo SDKAppID 输入框应为数字的问题。

Version 1.0.0 @2022.09.23

版权所有：腾讯云计算（北京）有限责任公司

- TUICallKit demo 快速跑通
- TUICallKit 快速接入
- TUICallKit API
- TUICallKit 界面定制指引
- TUICallKit (Web) 常见问题

Android & iOS

最近更新时间：2024-02-21 15:15:22

新版本（1.1.0.103）涉及 TUICallEngine 接口变更：

- 如果您在开发过程中，因为 maven latest 更新机制导致的 Android 构建报错 ，可以使用如下两种方式解决：
  - 升级 TUICallKit 到最新版本。
  - 修改 tuicallkit/build.gradle 的 tuicallengine 依赖为 1.0 的固定版本：`com.tencent.liteav.tuikit:tuicallengine:1.0.0.53` 。
- 如果您在开发过程中，因为 执行 pod update 导致 iOS 构建报错，可以使用如下两种方式解决：
  - 升级 TUICallKit 到最新版本。
  - Podfile 文件中添加 `pod 'TUICallEngine', '1.0.0.53'` 。

Version 2.2.0.860 @ 2024.2.1

功能优化

Android&iOS：全新的UI视效，功能更清晰，体验更好。

缺陷修复

Android&iOS：修复在会议、直播场景下，接听来电后，麦克风和摄像头设备互相占用问题。

Version 2.1.0.810 @ 2023.12.19

功能优化

- Android：优化未登录时调用 TUICallKit 接口异常的提示。
- Android： 优化针对 Android 14平台(API 34) 的兼容性，详见：Android 14 行为变更 。
- Android&iOS：优化用户昵称的显示，按照如下顺序显示：用户备注 > 用户昵称 > 用户 Id，默认为 userId。

缺陷修复

- iOS：修复群通话头像显示重叠问题。
- iOS：修复视频通话时开启悬浮窗发消息再回到通话界面键盘无法收起问题。
- iOS：修复视频通话中，先切换摄像头、关闭摄像头，再切换回原摄像头并重新打开时，无法切换和移动摄像头的问题。
- Android：修复在阿拉伯语等从右至左布局模式(RTL 模式)下单人视频通话的小窗口无法移动的问题。

Version 2.0.0.750 @ 2023.11.03

功能优化

- Android&iOS：UIKit 支持日语。
- Android&iOS：优化通话昵称显示。
- Android&iOS：调整铃声的默认音量，由60%调整到100%。

缺陷修复

- iOS：修复Swift 版本加载图片过慢问题。
- iOS：修复主叫端发起通话后退后台，通话邀请自动取消问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.9.0.680 @ 2023.09.27

功能优化

- Android&iOS：支持阿拉伯语言。
- Android&iOS：优化套餐购买提示，支持根据链接跳转到对应的套餐购买页。
- Android&iOS：优化不同分辨率下的默认码率，高分辨率下画面更清晰，详见 Resolution 。
- Android&iOS：调整视频通话的默认码率为600kbps，美颜等级为4级。

缺陷修复

- Android&iOS：对黑名单用户发起通话，通话拒绝提示与发送单聊消息拒绝提示不一致。
- iOS：4人群视频通话界面，一名成员拒绝接听，发起人视频位置画面异常。
- iOS：修复登录成功后立即设置美颜，分辨率会被重置问题。

Version 1.8.0.620 @ 2023.08.14

功能优化

- Android&iOS：通话消息默认不计入未读数。
- Android：优化小米手机悬浮窗权限跳转页面。

缺陷修复

- iOS：修复被移下线后回调接口 onKickOffline 无效问题。
- iOS： 修复通话记录未接界面清除通话后，回到所有通话界面，列表为空问题。

Version 1.7.2.570 @ 2023.07.20

功能优化

Android：默认关闭重力感应，优化在大屏、定制设备的通话体验。

缺陷修复

- Android&iOS：A 呼叫 B（离线）后取消，A 再次呼叫 B，B 登录上线，B 的云端通话记录异常问题。
- Android： 升级 TRTC SDK 版本到11.3后，TUICallKit 的崩溃问题。

Version 1.7.0.460 @ 2023.06.25

功能优化

- Android&iOS：含 UI 集成方案，优化示例工程，完善通话设置项。
- Android：降低通话时系统保活的级别，只在状态栏显示保活提示，去掉通知和震动。

Version 1.6.1.410 @ 2023.05.22

新特性

- Android&iOS：含 UI 接口 call 、groupCall 支持自定义房间号。
- Android&iOS：支持发起通话时传入字符串型房间号，详见 CallParams 。

缺陷修复

- Android：修复通过 Arrays.asList 方式生成列表参数时，groupCall 报错问题。

版权所有：腾讯云计算（北京）有限责任公司

- Android：修复视频通话显示异常问题。
- iOS：修复与 TUIRoom 组件冲突问题。
- iOS：修复登录成功回调中，立刻发起通话崩溃问题。
- iOS：修复点击通知消息进入 APP，偶现不显示邀请页面问题。

Version 1.6.0.360 @ 2023.04.27

新特性

- Android：TUICallKit 新增 Kotlin 语言版本。
- iOS：TUICallKit 新增 Swift 语言版本。
- Android&iOS：新增通话记录（本地）的显示页面。

功能优化

- Android：优化单人视频通话头像显示。
- Android&iOS：群组通话中，默认支持邀请其他群成员加入通话。

缺陷修复

- Android：修复 Android 12及其以上 设备连接蓝牙后无声的问题。
- Android：修复被叫端偶现静音设置不生效问题。
- iOS：修复重新登录后偶现设备接收不到来电邀请的问题。
- IOS：修复 TUICallKit 中 enableCustomViewRoute 接口失效问题。
- IOS：修复 VoIP 推送页面昵称显示错误的问题。

Version 1.5.1.310 @ 2023.04.17

新特性

- Android&iOS：新增 VoIP 消息推送功能，提供更好的通话接听体验。
- Android&iOS：支持发起通话时自定义扩展字段，详见 call 方法的 TUICallDefine.CallParams 参数。

功能优化

- Android&iOS：优化华为、小米、FCM 等厂商的离线推送能力，增加厂商消息分类及 channelID 设置。

缺陷修复

- Android&iOS：修复 IM 自定义属性发起通话后被覆盖的问题。
- Android：修复回调 onCallEnd 中 totalTime 的时间单位错误的问题；

Version 1.5.0.305 @ 2023.03.09

功能优化

- Android&iOS：优化上屏消息显示。
- Android：贴耳息屏功能默认关闭。
- Android：升级 gradle 插件及版本。
- Android：优化铃声播放类，支持循环播放。

缺陷修复

- Android&iOS：修复被叫接听通话失败，没有 onCallCancel 回调问题。
- Android：修复被叫接听通话失败，主叫异常问题。
- Android：修复首次通话检查权限时，主叫端取消通话，被叫端再次拉起界面问题。
- Android：修复回调上层的网络质量 userId 为空问题。

Version 1.4.0.255 @ 2023.01.06

新特性

Android&iOS：支持自定义来电超时时间，详见 call方法 TUICallDefine.CallParams 参数。

缺陷修复

Android&iOS：修复主动加入房间（joinInGroupCall）时通话异常结束问题。
Android：修复在音视频通话接听中退后台，再回到前台时，通话状态异常问题。

Version 1.3.0.205 @ 2022.11.30

新特性：

Android&iOS：新增美颜设置接口 setBeautyLevel，支持关闭默认美颜。

功能优化：

iOS: 优化 TUICallKit 的 Framework 体积。

缺陷修复：

- Android&iOS：修复服务端解散房间或踢出用户时，通话界面不消失问题。
- Android：修复A呼叫离线用户B，然后取消；A 再次呼叫 B，B 上线后通话界面不显示问题。

Version 1.2.0.153 @ 2022.11.14

新特性：

- Android&iOS：支持视频自定义编码分辨率。
- Android&iOS：支持设置视频的渲染参数：渲染方向、填充模式。
- Android&iOS：支持第三方美颜的接入。
- Android&iOS：TUICallKit 新增重载接口 call()、groupCall()，支持自定义离线消息（详情请参见 API文档 ）。

功能优化：

- Android&iOS：优化部分 TUICallObserver 回调异常问题。
- Android&iOS：优化视频切语音功能，支持离线状态下切换。
- Android&iOS：完善 TUICallKit 的错误码及错误提示。
- iOS：规范 TUICallEngine、TUICallKit Swift API 名称。

缺陷修复：

- Android&iOS：修复重新登录账号后，仍会收到之前已拒绝的来电问题。
- Android：修复单聊中收到群聊邀请，通话异常挂断问题。
- Android：多场景退出直播间无法发起通话异常问题。
- Android：修复 A 呼叫 B，同时 B 呼叫 C，C 概率性进入 A 的房间问题。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.1.0.103 @ 2022.09.30

- Android&iOS：优化群组通话过程中，邀请群成员加入功能。
- Android&iOS：优化通话流程，避免通话接听前产生录制、审核等费用。
- Android&iOS：支持自定义音视频通话的离线推送消息。
- Android&iOS：部分 TUICallEngine 接口参数变更，详见 call()、groupCall()、inviteUser()、以及回调接口 onCallReceived()。
- Android&iOS：修复群组通话过程中，偶现的少量回调异常问题。
- Android&iOS：修复重复登录以及UserSig过期造成的状态异常问题。
- iOS：修复 TUCallKit Objective-C 和 Swift 混合编译时，`init` 接口报错的问题。
- Android：修复 Kotlin 集成悬浮窗功能时编译报错问题。

Version 1.0.0.53 @ 2022.08.15

全新发布：

- Android&iOS：支持1v1 音视频通话、群组音视频通话。
- Android&iOS：支持主流厂商的离线唤醒。
- Android&iOS：支持自定义头像、自定义昵称。
- Android&iOS：支持通话过程中开启悬浮窗。
- Android&iOS：支持设置自定义铃音。
- Android&iOS：支持多平台登录状态下的来电服务。

版权所有：腾讯云计算（北京）有限责任公司

Uni-App

最近更新时间：2023-12-20 10:37:12

Version 1.9.0 @ 2023-10-20

优化：

- Android&iOS：支持阿拉伯语言。
- Android&iOS：优化不同分辨率下的默认码率，高分辨率下画面更清晰，详见 Resolution 。
- Android&iOS：调整视频通话的默认码率为600kbps。
- Android&iOS：优化铃声播放逻辑，主叫端跟随音频设备播放，被叫端默认以扬声器播放。

修复：

- Android&iOS：修复对黑名单用户发起通话，通话拒绝提示与发送单聊消息拒绝提示不一致。
- iOS：修复4人群视频通话界面，一名成员拒绝接听，发起人视频位置画面异常。

Version 1.8.1 @ 2023-08-25

新增：

iOS：新增 enableMuteMode 接口，支持静音来电铃声。

Version 1.8.0 @ 2023-08-18

优化：

- Android & iOS：优化通话消息默认不计入未读数。
- Android：优化小米手机悬浮窗权限跳转页面。

修复：

- iOS：修复被踢下线后回调接口 onKickOffline 无效问题。
- iOS： 修复通话记录未接界面清除通话后，回到所有通话界面，列表为空问题。

Version 1.7.2 @ 2023-07-25

优化：

Android：默认关闭重力感应，优化在大屏、定制设备的通话体验。

修复：

- Android & iOS：A呼叫B(离线)后取消，A再次呼叫B，B登录上线，B的云端通话记录异常问题。
- Android： 升级TRTC SDK版本到11.3后，TUICallKit 的崩溃问题。

Version 1.7.1 @ 2023-07-06

新增：

- Android&iOS：新增云端通话记录，您可以在控制台开通服务进行体验查询。
- Android&iOS：含 UI 接口 call 、groupCall 支持自定义房间号。
- Android&iOS：支持发起通话时传入字符串型房间号，详见 。
- Android&iOS：新增 accept 方法，支持自动接听。

版权所有：腾讯云计算（北京）有限责任公司

优化：

Android：降低通话时系统保活的级别，只在状态栏显示保活提示，去掉通知和震动。

修复：

- Android：修复视频通话显示异常问题。
- iOS：修复登录成功回调中，立刻发起通话崩溃问题。
- iOS：修复点击通知消息进入 APP，偶现不显示邀请页面问题。

Version 1.6.2 @ 2023-06-15

修复：

iOS：修复会议群里被叫端接听通话失败的问题。

Version 1.6.1 @ 2023-05-16

修复：

Android：修复会议群里被叫端接听通话失败的问题。

Version 1.6.0 @ 2023.05.04

新特性：

Android&iOS：TUICallEngine 新增 setVideoEncoderParams 方法设置视频编码参数。

功能优化：

- Android：优化单人视频通话头像显示。
- Android&iOS：群组通话中，默认支持邀请其他群成员加入通话。

缺陷修复：

- Android：修复 Android 12及其以上 设备连接蓝牙后无声的问题。
- Android：修复被叫端偶现静音设置不生效问题。
- iOS：修复重新登录后偶现设备接收不到来电邀请的问题。

Version 1.5.0 @ 2023.03.15

功能优化：

- Android：贴耳息屏功能默认关闭。
- Android：优化铃声播放类，支持循环播放。

缺陷修复：

- Android&iOS：修复被叫接听通话失败，没有 onCallCancel 回调问题。
- Android：修复被叫接听通话失败，主叫异常问题。
- Android：修复首次通话检查权限时，主叫端取消通话，被叫端再次拉起界面问题。
- Android：修复回调上层的网络质量 userId 为空问题。

Version 1.4.0 @ 2023.01.10

新特性：

- Android&iOS：增加 TUICallEngine 文件，提供 hangup 方法用来挂断通话。
- Android&iOS：增加 TUICallEngine 文件，提供 setVideoRenderParams 方法用来设置用户视频画面的渲染模式。

缺陷修复：

- Android&iOS：修复主动加入房间 joinInGroupCall 时通话异常结束问题。
- Android：修复在音视频通话接听中退后台，再回到前台时，通话状态异常问题。

Version 1.3.0 @ 2022.12.22

缺陷修复：

- Android&iOS：修复服务端解散房间或踢出用户时，通话界面不消失问题。
- Android：修复A呼叫离线用户B，然后取消；A 再次呼叫 B，B 上线后通话界面不显示问题。

Version 1.2.0 @ 2022.11.26

功能优化：

- Android&iOS：优化部分 TUICallObserver 回调异常问题。
- Android&iOS：优化视频切语音功能，支持离线状态下切换。
- Android&iOS：完善 TUICallKit 的错误码及错误提示。

缺陷修复：

- Android&iOS：修复重新登录账号后，仍会收到之前已拒绝的来电问题。
- Android：修复单聊中收到群聊邀请，通话异常挂断问题。
- Android：多场景退出直播间无法发起通话异常问题。
- Android：修复 A 呼叫 B，同时 B 呼叫 C，C 概率性进入 A 的房间问题。

Version 1.1.0.104 @ 2022.11.02

缺陷修复：

iOS 被呼叫无铃音。

Version 1.1.0.103 @ 2022.10.27

新特性：

增加抛出的事件，例如：onUserLeave、onUserLineBusy 等。

功能优化：

- 插件体积由 45.4M 优化到 36M 左右。
- 优化音视频通话套餐异常提示。
- 优化通话流程，避免通话接听前产生录制、审核等费用。
- 修复群组通话过程中，偶现的少量回调异常问题。
- 修复重复登录以及 UserSig 过期造成的状态异常问题。

Version 1.0.1 @ 2022.08.16

缺陷修复：

iOS 错误提示调整。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.0.0 @ 2022.08.10

全新发布：

- 客户端&小程序：支持1v1 音视频通话、群组音视频通话。
- 客户端&小程序：支持自定义头像、自定义昵称。
- 客户端&小程序：支持设置自定义铃音。
- 客户端：支持通话过程中开启悬浮窗。
- 客户端：支持多平台登录状态下的来电服务。

版权所有：腾讯云计算（北京）有限责任公司

Flutter

最近更新时间：2023-12-08 14:53:42

Version 2.0.3 @2023.12.04

问题修复

Android&iOS：修复了点击来电通知后有时不显示来电页面的偶现问题。

Version 2.0.2 @2023.11.27

问题修复

Android：修复工程中有多个 Flutter Engine 实例时导致 Method Channel 通信异常的问题。

Version 2.0.1 @2023.11.15

问题修复

Android&iOS：修复了在使用其它 TRTC 组件时导致 TRTC Observer 不兼容的问题。
Android：修复了与 TUIRoom 结合使用时的不兼容问题。

Version 2.0.0 @2023.11.06

依赖描述

升级依赖客户端 SDK 版本：

- Android&iOS TUICore：7.6.5011。
- Android&iOS TUICallEngine：2.0.0.750。

Version 1.9.1 @2023.10.21

新特性

iOS：支持 Voip。

问题修复

iOS：修复后台接听电话时通话页面异常的问题。

Version 1.9.0 @2023.10.09

新特性

Android&iOS：添加设置铃声的接口。

功能优化

- Android&iOS: 优化购买包的提示。
- Android&iOS: 优化不同分辨率的默认比特率。

问题修复

iOS：修复了同一个 Observer 对象可以被注册两次的问题。

依赖描述

升级依赖客户端 SDK 版本：Android&iOS TUICore：7.5.4852，Android&iOS TUICallEngine：1.9.0.680。

Version 1.8.3 @2023.08.25

问题修复

- Android&iOS：修复了使用 tencent_cloud_chat_uikit 时没有通话消息显示的问题。
- Android&iOS：修复了在单人通话过程中偶尔拉起群通话页面的问题。
- Android&iOS：修复了在通话过程中偶尔拉起通话页面两次的问题。
- Android&iOS：修复了通话时间显示异常的问题。

功能优化

Android：优化了在后台接收到通话时无法拉起界面的问题。

依赖描述

升级 tencent_cloud_uikit_core 到版本 1.1.1。

Version 1.8.2 @2023.08.19

问题修复

iOS：修复了某些编译器由于使用了已弃用的 Swift 接口而无法编译的问题。

Version 1.8.1 @2023.08.18

缺陷修复

Android&iOS：修复了在进行群通话语音通话时，另一方的视频流显示的问题。

Version 1.8.0 @ 2023.08.17

新特性

- Android&iOS：基于 Dart 语言构建全新的 TUICallkit，可以更方便的定制自己的 UI 风格。
- Android&iOS：TUICallEngine 新增诸如 hangup、accept、reject 等多个业务接口。

Version 1.7.4 @ 2023.07.20

功能优化

Android：默认关闭重力感应，优化在大屏、定制设备的通话体验。

缺陷修复

Android&iOS：A 呼叫 B (离线)后取消，A 再次呼叫 B，B 登录上线，B 的云端通话记录异常问题。

Version 1.7.3 @ 2023.07.19

功能优化

Android：支持使用模拟器进行开发&调试。

依赖说明

升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional：11.3.0.13176。

版权所有：腾讯云计算（北京）有限责任公司

Version 1.7.2 @ 2023.07.09

缺陷修复

iOS：升级客户端 SDK 版本，修复因为 Non-public API usage 问题导致的 AppStore 上架失败的问题。

Version 1.7.1 @ 2023.07.03

新特性

Android&iOS：新增云端通话记录，您可以在控制台开通服务进行体验查询。

功能优化

Android：降低通话时系统保活的级别，只在状态栏显示保活提示，去掉通知和震动。

Version 1.6.3 @ 2023.06.07

缺陷修复

- iOS：修复调用 joinInGroupCall 后中途加人页面为空的问题。
- iOS：修复调用 joinInGroupCall 后用户画面遮挡的问题。

Version 1.6.2 @ 2023.05.24

缺陷修复

Android：修复调用 joinInGroupCall api 引起 Crash 问题 。

Version 1.6.1 @ 2023.05.14

缺陷修复

- Android：修复在 Vivo 特定机型上申请悬浮窗权限时偶现的崩溃问题。
- 依赖说明。
- 升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional：11.1.0.13111、iOS TXLiteAVSDK_Professional：11.1.14143。

Version 1.6.0 @ 2023.05.07

新特性

- Android&iOS：增加 hangup 挂断接口。
- Android&iOS：增加用户自定义字段、增加用户自定义通话超时时长。
- Android&iOS：群组通话增加中途加人页面。

功能优化

- Android：优化单人视频通话头像显示。
- Android&iOS：群组通话中，默认支持邀请其他群成员加入通话。

缺陷修复

- Android：修复 Android 12及其以上 设备连接蓝牙后无声的问题。
- Android：修复被叫端偶现静音设置不生效问题。
- iOS：修复重新登录后偶现设备接收不到来电邀请的问题。

版权所有：腾讯云计算（北京）有限责任公司

- iOS：修复 VoIP 推送页面昵称显示错误的问题。

依赖说明

升级依赖的客户端 SDK 版本：Android LiteAVSDK_Professional：11.1.0.13111、iOS TXLiteAVSDK_Professional：11.1.14143。

Version 1.5.4 @ 2023.04.14

新特性

- Android&iOS：增加小米、华为、VIVO 离线推送参数。
- iOS：支持 Voip 消息推送功能。
- Android&iOS：增加分辨率设置功能。

功能优化

Android：优化 onCallEnd 回调中 totaltime 参数单位为毫秒。

缺陷修复

- Android&iOS：修复 onCallReceived 回调异常问题。
- iOS：修复屏幕旋转时通话页面显示不全的问题

Version 1.5.3 @ 2023.03.17

缺陷修复

- Android：修复打包失败问题。
- Android&iOS：修复回调方法未实现时抛出异常的问题。

Version 1.5.2 @ 2023.03.13

缺陷修复

Android：修复 `TUICallDefine.OfflinePushInfo` API 变更导致的编译报错。

Version 1.5.1 @ 2023.03.13

缺陷修复

Android&iOS：修复 `tencent_calls_engine` 版本依赖错误的问题。

Version 1.5.0 @ 2023.03.13

功能优化

- Android：贴耳息屏功能默认关闭。
- Android：升级 gradle 插件及版本。
- Android：优化铃声播放类，支持循环播放。

缺陷修复

- Android&iOS：修复被叫接听通话失败，没有 onCallCancel 回调问题。
- Android：修复被叫接听通话失败，主叫异常问题。
- Android：修复首次通话检查权限时，主叫端取消通话，被叫端再次拉起界面问题。

版权所有：腾讯云计算（北京）有限责任公司

- Android：修复回调上层的网络质量 userId 为空问题。
- iOS：修复 Example 中 Observer 注册时机错误的问题。

Version 1.4.2 @ 2023.02.24

缺陷修复

- Android&iOS：修复 Example 中错误的 Observer 注册时机导致的通话异常问题。
- iOS：修复 `removeObserver` API 偶现设置无效的问题。

Version 1.4.1 @ 2023.02.20

缺陷修复

Android：修复通话结束后，`OnCallEnd`事件丢失的问题。

Version 1.4.0 @ 2023.01.16

缺陷修复：

- Android&iOS：修复主动加入房间（joinInGroupCall）时通话异常结束问题。
- Android：修复在音视频通话接听中退后台，再回到前台时，通话状态异常问题。
- Android：修复同时集成 `tencent_cloud_chat_uikit` 插件时，登录状态导致的通话发起失败问题。
- Android：修复发起群组通话中因为参数检查问题导致的通话发起失败问题。

Version 1.3.1 @ 2022.12.27

新特性：

Android&iOS：支持通话时自定义离线推送消息。

缺陷修复：

Android：修复同时集成 tencent_cloud_chat_uikit 插件时，提示“sdkappid is invalid”的问题。

Version 1.3.0 @ 2022.12.02

功能优化：

iOS：优化 TUICallKit 的 Framework 体积。

缺陷修复：

- Android&iOS：修复服务端解散房间或踢出用户时，通话界面不消失问题。
- Android：修复A呼叫离线用户B，然后取消；A 再次呼叫 B，B 上线后通话界面不显示问题。

Version 1.2.2 @ 2022.11.17

缺陷修复：

iOS：修复静态库链接导致的编译问题。

Version 1.2.0 @ 2022.11.17

新特性：

- 支持1v1 音视频通话、群组音视频通话。

版权所有：腾讯云计算（北京）有限责任公司

- 支持自定义头像、自定义昵称。
- 支持设置自定义铃音。
- 支持通话过程中开启悬浮窗。
- 支持多平台登录状态下的来电服务。

版权所有：腾讯云计算（北京）有限责任公司

微信小程序

最近更新时间：2024-01-11 15:42:52

Version 2.1.1 @2023.12.29

优化

- 优化群组通话下的提示信息。
- 优化昵称过长时的显示问题。

修复

- 修复小程序 setSelfInfo 接口传参问题。
- 修复 destroyed 存在的问题。
- 修复不同呼叫场景下悬浮窗里的挂断问题。
- 修复主叫呼叫状态下显示远端问题。

Version 2.1.0 @2023.12.01

新增

融合到 Chat 中增加 isFromChat 上报。

修复

修复按钮 loading 下可点击的问题。

Version 2.0.0 @2023-11-17

Breaking Change

升级底层依赖 tuicall-engine-wx 至 ^2.0.0，不再支持 tim-wx-sdk 创建 tim 实例，如需创建 tim 实例，请使用 @tencentcloud/chat 创建。

新增

- 增加主叫、被叫的铃声。
- 增加被叫来电铃声静音接口 enableMuteMode 。
- 增加呼叫黑名单用户失败提示信息。
- 增加对日语的支持。
- 增加 TUICallKit 框架上报。
- 新增无套餐的弹框提示。

优化

- 视频通话下，大屏默认显示远端用户。
- 优化远端流用户信息显示优先级：备注 > 昵称 > userId。
- 优化 TUICallKit 组件包体积（删除无用图片及代码）。

Version 1.4.4 @2023-07-28

新增

新增 init 错误抛出处理。

Version 1.4.3 @2023-07-14

修复

调整事件监听时机，修复 SDK_READY 未抛出的问题。

Version 1.4.2 @2023-06-09

修复

- 修复全局监听卸载不彻底的问题。
- 修复全局监听本地预览取值不当的问题。

Version 1.4.1 @2023-05-25

新增

新增 call、groupCall 错误抛出处理。

Version 1.4.0 @2023-05-17

新增

新增在 tabBar 页面接入时的兼容性处理。

优化

- 优化在页面跳转情况下的初始化以及卸载逻辑。
- 优化全局监听逻辑，防止重复监听。

Version 1.3.9 @2023-05-11

新增

- 新增 call、groupCall 支持自定义数字房间号。
- 新增 call、groupCall 允许用户自定义 userData 数据。

修复

修复全局监听模式下，视频通话被叫预览黑屏导致无法接听的问题。

Version 1.3.8 @2023-04-27

> ⚠ **注意：**
> 需要依赖 tuicall-engine-wx 的版本 ⩾ v1.4.7。

新增

新增通话页面出现时的动画效果。

优化

- 优化通话界面唤起的时间。
- 优化用户异常退出的逻辑，如右滑退出以及返回退出的情况。

修复

修复用户重复点击，导致接口被重复调用的问题。以下接口增加节流处理： call，groupCall，accept，reject 。

Version 1.3.7 @2023-04-14

新增

新增 设置日志级别接口 。

修复

修复铃声路径错误问题。

Version 1.3.6 @2023-03-31

新增

新增 index.js 入口文件，解决 npm 构建时的警告问题。

优化

- 优化用户没有 live-pusher 推流标签使用权限时，给予弹窗提示。
- 优化针对在微信开发者工具运行时，给予弹窗提示。

Version 1.3.5 @2023-03-24

新特性

支持自定义铃声。

修复

优化 reset 函数在生命周期中的触发时机，修复全局监听下被叫方的白屏问题。

Version 1.3.4 @2023-03-16

修复

全局监听下的视频通话，被叫方使用 iOS 设备预览黑屏问题。

优化

优化常见错误提示，方便客户快速定位问题。

Version 1.3.3 @2023-03-02

优化

- 弹窗昵称显示优先级（昵称 > userID）。
- 打印显示 TUICallKit 版本号。

Version 1.3.2 @2023-02-17

修复

打包路径错误。

Version 1.3.1 @2023-02-17

修复

修复远端关闭麦克风，没有及时更新远端流导致后续调用音量变化接口报错。

Version 1.3.0 @2023-02-13

优化

优化依赖引入，组件包体积由900KB降低至110KB左右。

Version 1.2.7 @2023-02-03

修复

去除 aegis.js 文件。

Version 1.2.6 @2023-01-03

修复

未及时回收 wx.$TUICallEngine 变量引发的 init 报错。

Version 1.2.5 @2023-01-31

新特性

支持 uniapp 小程序全局监听，具体 详见。

修复

升级 tsignaling-wx，解决和客户端的互通问题。

Version 1.2.4 @2023-01-12

修复

- 打包路径错误。
- callManager 兼容 sdkAppID 与 SDKAppID 传参。
- callkit 生命周期初始化时 移除 wx.CallManager 监听报错问题。

Version 1.2.3 @2023-01-10

修复

TUICallKit 样式布局错误。

Version 1.2.1 @2023-01-09

修复

打包问题引起的 globalCall 页面引用错误。

Version 1.2.0 @2023-01-06

新特性

支持全局监听

- 主包方案详见 【TUICallKit】小程序全局监听（主包方案）。

- 分包方案详见 【TUICallKit】小程序全局监听（分包方案）。

Version 1.1.4 @2022-12-23

修复

- init 方法支持外部传入 tim。
- 修复 uni-app 打包小程序存在的栈溢出问题。

Version 1.1.3 @2022-12-14

新增

新增设备权限弹出窗口。

优化

群通话中的昵称显示优先级。

> **说明：**
> 群通话显示优先级为**昵称 > userID**。

修复

群通话中 player 推流过程中 trtc-wx 报错问题。

Version 1.1.2 @2022-11-30

优化

优化接入流程。

Version 1.1.1 @2022-11-30

优化

兼容 chat，支持 TUICallKit 在 init 时传值。

Version 1.1.0 @2022.11.16

> **注意：**
> 升级至该版本需注意：升级指引。

优化

- 提高多人通话稳定性
- 抛出事件的数据结构调整，具体请参见 TUICallEvent。

常见问题（TUICallKit）
全平台

最近更新时间：2023-10-17 16:48:51

> **说明：**
> 如果并没有解决您的问题，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群： **605115878** 来进行交流讨论。

音视频通话功能出现了未开通的错误提示?

您可以选择体验版（7 天试用）或者购买正式版。

对于体验版

进入 即时通信 IM 控制台 的配置页面，并在页面的右下角找到开通腾讯实时音视频服务功能区，单击**免费体验**，选择**领取7天试用**，即可开通 TUICallKit 的 7 天免费试用服务，成为体验版。



对于正式版

如果您体验过 7 天体验期后，需要购买正式套餐。IM 音视频通话能力针对不同的业务需求提供了差异化的付费版本供您选择，您可以在 IM 购买页 了解包含功能并选购您适合的版本。

报错 inviteID 无效或邀请已处理

该 inviteID 已经被主叫方取消，或者该邀请已经被接受。由于信令发送受网络带宽等诸多因素影响，在极端情况下(accept 与 reject 接口重复调用)可能会导致 inviteID 无效的错误。

消息发送方或接收方 UserID 无效或不存在

您呼叫的用户还没有登录过 TIM，导致呼叫失败，请尝试呼叫已经登录成功过的用户。

TUICallKit 是否可以不引入 IM SDK，只使用 TRTC?

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，比如通话拨打信令、通话忙线信令等核心逻辑，如果您已经购买有其他 IM 产品，也可以参照 TUICallKit 逻辑进行适配。

如何购买音视频通话套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请点击页面右侧，进行售前套餐咨询；或者加入 QQ 群：**605115878**，进行咨询和反馈。

如何获取通话的房间号 Roomid?

版权所有：腾讯云计算（北京）有限责任公司

通话接通后，您可以通过 onCallBegin 返回 roomid 字段进行获取。

查询用户在线状态

TUICallKit 底层通过 IM 管理用户，可以通过 IM 查询用户的在线状态，方式有以下两种：

- **方式一：**通过 getUserStatus 高级接口查询

  详情可参考：查询用户状态（ Android&iOS&Windows&Mac 、web&小程序&uniapp ），您可以调用不同平台的接口去查询。

  注意：

  1、查询其他用户状态需要升级 TUICallKit 套餐包到**群组通话版本**。

  2、查询其他用户状态需要提前在 即时通信 IM 控制台 开启 “用户状态查询及状态变更通知”。不开启，调用 getUserStatus 会报错。



  getUserStatus 接口的返回结果有以下几种：

  - 未知（UNKNOWN）
  - 在线（ONLINE）：当前用户已登录上线，通常情况下，该用户可以正常收到通话请求。
  - 离线（OFFLINE）：用户未主动调用 logout 退出登录，但长连接中断的状态。通常情况下，若接入了离线推送，该用户可以收到离线推送的消息。
  - 未登录（UNLOGINED）：用户注册账号后从未登录过，或者用户主动调用 logout 退出登录，该用户收不到通话请求。

- **方式二：**通过 rest-api 进行查询

  详情可参考： 查询账号在线状态 。

Web

最近更新时间：2024-01-24 15:28:01

说明：

- 如果并没有解决您的问题，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群： **605115878** 来进行交流讨论。
- 我们团队即将在 Web 端丰富更多不同的组件，以满足您的开发需求。
  希望您能抽出几分钟时间，为您自己的需求投上一票，我们将优先开发！
  问卷地址：TUICallKit Web 问卷调查

一、基础环境问题

说明：
以下问题对于 TUICallKit 与 TUICallEngine SDK 同样适用。

TUICallEngine 和 TUICallKit 分别是什么?

TUICallKit 是含 UI 音视频通话组件，底层是用 TUICallEngine SDK，目前支持 Typescript+Vue2 / Typescript+Vue3，可直接将组件放到页面中，调用简单的接口即可直接实现音视频通话，开源地址为：TUICallKit/Web。
TUICallEngine SDK 是音视频通话组件的无 UI SDK，如果 TUICallKit 的交互并不满足您的需求，您可以使用这套接口自己封装交互。npm 地址为 tuicall-engine-webrtc。

Web 端支持哪些浏览器? 报错“获取设备权限失败”?

请先确保页面已被授权使用麦克风或摄像头，参见 设备授权说明。
对浏览器的详细支持度，请参见 浏览器兼容信息。
对于上述没有列出的环境，您可以在需要检测的浏览器打开 能力测试 测试是否完整的支持 WebRTC 的功能。

为什么本地开发测试能正常使用，但是部署到线上用 IP 访问后无法正常视频/语音通话?

- **对网站域名协议的要求**
  出于对用户安全、隐私等问题的考虑，浏览器限制网页在 HTTPS 协议下才能正常使用本文档中所对接组件的全部功能。为确保生产环境中的用户能够顺畅体验产品功能，请将您的网站部署在 `https://` 协议的域名下。更多请参见 页面访问协议说明。
- **对网络环境的要求**
  在使用 TUICallKit 时，用户可能因防火墙限制导致无法正常进行音视频通话，请参考 应对防火墙策略 将相应端口及域名添加至防火墙白名单中。

在接通过程中报："is not included in the current tim's package"?

TUICallKit（含 TUICallEngine SDK）依赖的 `tim-js-sdk` 版本需要 >= 2.21.2。
如果 `tim-js-sdk` 依赖包版本正确，则可能是 SDKAppID 未购买音视频套餐或套餐包不支持所调用的功能，请访问 音视频通话功能出现了未开通的错误提示。

TUICallKit（含 TUICallEngine）是否支持接收离线消息?

不支持接收离线消息。
支持离线消息推送，可以通过 call / groupCall 中的 offlinePushInfo 添加需要推送的消息。

TUICallEngine init 未完成，需要在 init 完成后使用此API

未调用 login 接口，所有功能需要先进行登录完成后才能使用，具体参考 TUICallEngine login 。

> ⚠ 注意：
> TUICallEngine login 是一个异步接口，避免在 login 还未完成，直接调用 TUICallEngine 的接口。

当前通话状态无法使用该 API

API 与通话状态对照表：

API	idle	calling	connected
call	✓	×	×
groupCall	✓	×	×
accept	×	✓	×
reject	×	✓	×
hangup	×	✓	✓
inviteUser	×	✓	✓
joinInGroupCall	✓	×	×

二、TUICallKit 问题（含 UI）

TUICallKit Web 支持什么框架？支持 H5 吗？

TUICallKit 适用于 `Vue2.7 + Typescript` 或者 `Vue3 + Typescript` 项目，若您采用其他语言或者技术栈，请访问 界面定制指引。
TUICallKit 支持 H5，通过页面 UA 自动修改适应移动端的布局，为了移动端的良好体验，推荐通过修改 CSS 在 H5 时将 `<TUICallKit/>` 组件放大至全屏。

TUICallKit 打包失败？

对于 Vite 项目，您需要在 `vite.config.js` 中添加 `base: "./"` 。
对于 Vue-CLI 创建的 webpack 项目，您需要在 `vue.config.js` 中添加 `publicPath: "./"` 。

TUICallKit 报错“获取设备权限失败”？

请先确保页面已被授权使用麦克风或摄像头，参见 设备授权说明。
可以尝试 官方 Demo 是否可以正常通话，然后在 设备检测页面 检查是否支持 webrtc。如果还有问题，请联系我们 QQ群：646165204。

timeout 字段设置无效原因？

该字段目前在 call / groupcall API 中会被使用到。
目前全平台 TUICallKit 的策略是被叫登录后，仅拉取 30s 内的历史消息。所以被叫登录后无法拉到主叫 30s 前的呼叫信息，从而导致被叫无法拉起邀请页面进行通话。

视频通话切换大小屏出现闪屏现象？

ios 17 设备和 WebRTC 底层兼容性问题，等待后续修复。

safari 浏览器省电模式出现暂停键？

浏览器层面针对省电模式的视频播放策略，SDK 无法处理。

Android

最近更新时间：2023-09-27 14:16:12

TUICallKit 是否可以不引入 IM SDK，只使用 TRTC?

不可以，TUIKit 全系组件都使用了腾讯云 IM SDK 作为通信的基础服务，例如通话拨打信令、通话忙线信令等核心逻辑，如果您已经购买有其他 IM 产品，也可以参照 TUICallKit 逻辑进行适配。

错误提示“The package you purchased does not support this ability”？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，请参见 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用 TUICallKit 组件。

如何购买音视频通话套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请点击页面右侧，进行售前套餐咨询；或者加入 QQ 群：**605115878**，进行咨询和反馈。

TUICallKit 崩溃，崩溃日志："No implementation found for xxxx"?

堆栈信息如下：

```
java.lang.UnsatisfiedLinkError: No implementation found for void
com.tencent.liteav.base.Log.nativeWriteLogToNative(int, java.lang.String, java.lang.String) (tried
Java_com_tencent_liteav_base_Log_nativeWriteLogToNative and
Java_com_tencent_liteav_base_Log_nativeWriteLogToNative__ILjava_lang_String_2Ljava_lang_String_2)
    at com.tencent.liteav.base.Log.nativeWriteLogToNative(Native Method)
    at com.tencent.liteav.base.Log.i(SourceFile:177)
    at com.tencent.liteav.basic.log.TXCLog.i(SourceFile:36)
    at com.tencent.liteav.trtccalling.model.impl.base.TRTCLogger.i(TRTCLogger.java:15)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.init(ServiceInitializer.java:36)
    at com.tencent.liteav.trtccalling.model.impl.ServiceInitializer.onCreate(ServiceInitializer.java:101)
    at android.content.ContentProvider.attachInfo(ContentProvider.java:2097)
    at android.content.ContentProvider.attachInfo(ContentProvider.java:2070)
    at android.app.ActivityThread.installProvider(ActivityThread.java:8168)
    at android.app.ActivityThread.installContentProviders(ActivityThread.java:7709)
    at android.app.ActivityThread.handleBindApplication(ActivityThread.java:7573)
    at android.app.ActivityThread.access$2600(ActivityThread.java:260)
    at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2435)
    at android.os.Handler.dispatchMessage(Handler.java:110)
    at android.os.Looper.loop(Looper.java:219)
    at android.app.ActivityThread.main(ActivityThread.java:8668)
    at java.lang.reflect.Method.invoke(Native Method)
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:513)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:1109)
```

> **注意：**
> **TUICallkit 不支持虚拟机，请用真机进行体验**；如在真机上遇到上述异常，是因为 TUICallKit 依赖的 TRTC 等 SDK 部分接口使用了 JNI 实现，Android Studio 在某些条件进行编译 APK 的时候可能并不会把 Native 的 .so 库打包进去，导致出现这类报错，重新 Clean 一下即可。

allowBackup 异常?

- **报错信息：** `Manifest merger failed : Attribute application@allowBackup value=(true) from AndroidManifest.xml`
- **问题原因：** TUICallKit 依赖的 IM SDK 中默认 allowBackup 的值为 false ，表示关闭应用的备份和恢复功能。
- **解决方法：** 您可以在您工程的 `AndroidManifest.xml` 文件中删除 allowBackup 属性或将该属性改为 false，表示关闭备份和恢复功能；也可以在 `AndroidManifest.xml` 文件的 application 节点中添加 `tools:replace="android:allowBackup";` 表示覆盖 IM SDK 的设置，使用您自己的设置。

TUICallKit 是否支持离线推送呢?

支持，Android 离线推送采用腾讯云的 TUIOfflinePush 组件，详情请参见 TUIOfflinePush，在 TUICallKit 组件中，我们也接入了该组件。

在通话邀请超时时间内，被邀请者如果离线再上线，能否弹出通话界面?

根据 App 的启动类型，分别有不同的情况：



如果离线再上线后没有弹出通话界面，请过滤 `onReceiveNewInvitation` 日志，检查是否拉取到历史消息，如果没有该日志的打印，请加入 QQ 群：**592465424**，联系我们协助处理。

应用在后台时，不能自动将通话界面拉取到前台

将应用从后台自动拉取到前台，需要检查 App 是否开启了“后台自启动”或“悬浮窗”权限。
值得注意的是，不同厂商、甚至同一厂商不同 Android 版本，其对于应用开放的权限以及权限名称也会存在不一致。例如，小米6只需要开启后台弹出界面权限，而红米需要同时打开后台弹出界面和显示悬浮窗权限。

> **说明：**
> 如果您在测试中手动开启了所有权限，依然无法自动拉起通话界面到前台，可能需要做厂商兼容，可以加 QQ 群: **605115878** 联系我们协助处理。

OPPO手机权限位置	小米手机权限位置





角标问题

目前仅华为支持收到离线消息或通话请求时生成角标，其他厂商暂不支持，详见：离线唤醒-角标问题 。

iOS

最近更新时间：2023-03-01 17:38:07

错误提示“The package you purchased does not support this ability”？

如遇以上错误提示，是由于您当前应用的音视频通话能力包过期或未开通，您可以参考快速接入的 步骤一：开通服务，领取或者开通音视频通话能力，进而继续使用TUICallKit 组件。

如何购买音视频通话套餐?

请参考购买链接 音视频通话 SDK 价格总览，如有其他问题，请单击页面右侧，进行售前套餐咨询；或者加入 QQ 群：605115878，进行技术交流和反馈。

如何修改 TUICallKit 源码?

使用 CocoaPods 导入组件，具体步骤如下：

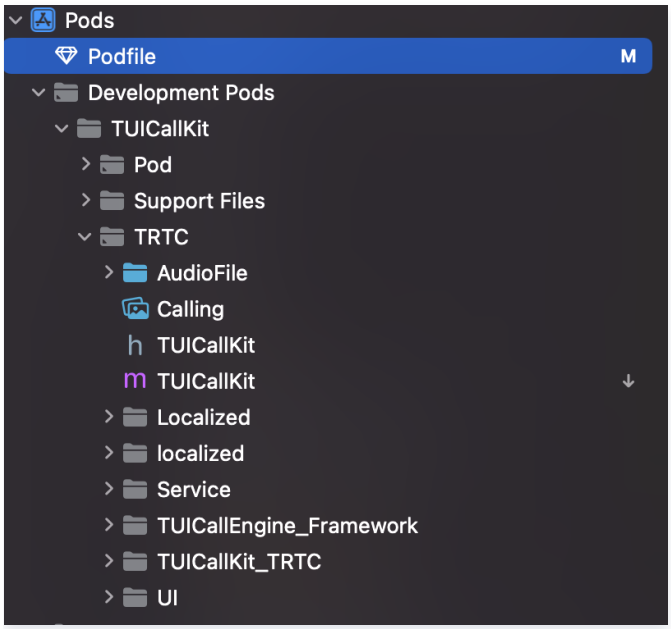
1. 在您的工程 Podfile 文件同一级目录下创建 TUICallKit 文件夹。
2. 单击进入 Github/TUICallKit ，选择克隆/下载代码，然后将 iOS 目录下的 TUICallKit 、 Resources 文件夹 和 TUICallKit.podspec 文件拷贝到您在 步骤1 创建的 TUICallKit 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖。

```
# :path => "指向TUICallKit.podspec 的相对路径"
pod 'TUICallKit', :path => "TUICallKit/TUICallKit.podspec"
```

4. 执行 pod install 命令，完成导入。

> **注意**
> TUICallKit 、 Resources 文件夹 和 TUICallKit.podspec 文件必需在同一目录下。

TUICallKit 组件集成后效果：



> **说明**

TUICallKit 组件集成后支持文件夹分层显示，方便您阅读和修改源代码。

TUICallKit 和自己集成的音视频库冲突了?

腾讯云的 音视频库 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。
1. 如果您使用了 TXLiteAVSDK_TRTC 库，不会发生符号冲突。可直接在 Podfile 文件中添加依赖，

```
pod 'TUICallKit'
```

2. 如果您使用了 TXLiteAVSDK_Professional 库，会产生符号冲突。您可在 Podfile 文件中添加依赖，

```
pod 'TUICallKit/Professional'
```

3. 如果您使用了 TXLiteAVSDK_Enterprise 库，会产生符号冲突。建议升级到 TXLiteAVSDK_Professional 后使用 TUICallKit/Professional 。

集成 TUICallKit 后运行报错 “ld: framework not found BoringSSL clang: error: linker command failed with exit code 1 sdk”

由于 TUICallKit 所依赖的音视频库暂不支持模拟器，请用真机运行或者调试。

TUICallKit 是否可以不引入 IM SDK，只使用 TRTC?

不可以。TUIKit 全系组件都使用了腾讯云 IM SDK 做为通信的基础服务，比如通话拨打信令、通话忙线信令等核心逻辑，如果您已经购买有其他 IM 产品，也可以参照 TUICallKit 逻辑进行适配。

TUICallKit 组件支持自定义铃声吗?

支持，调用 TUICalling#setCallingBell 即可。

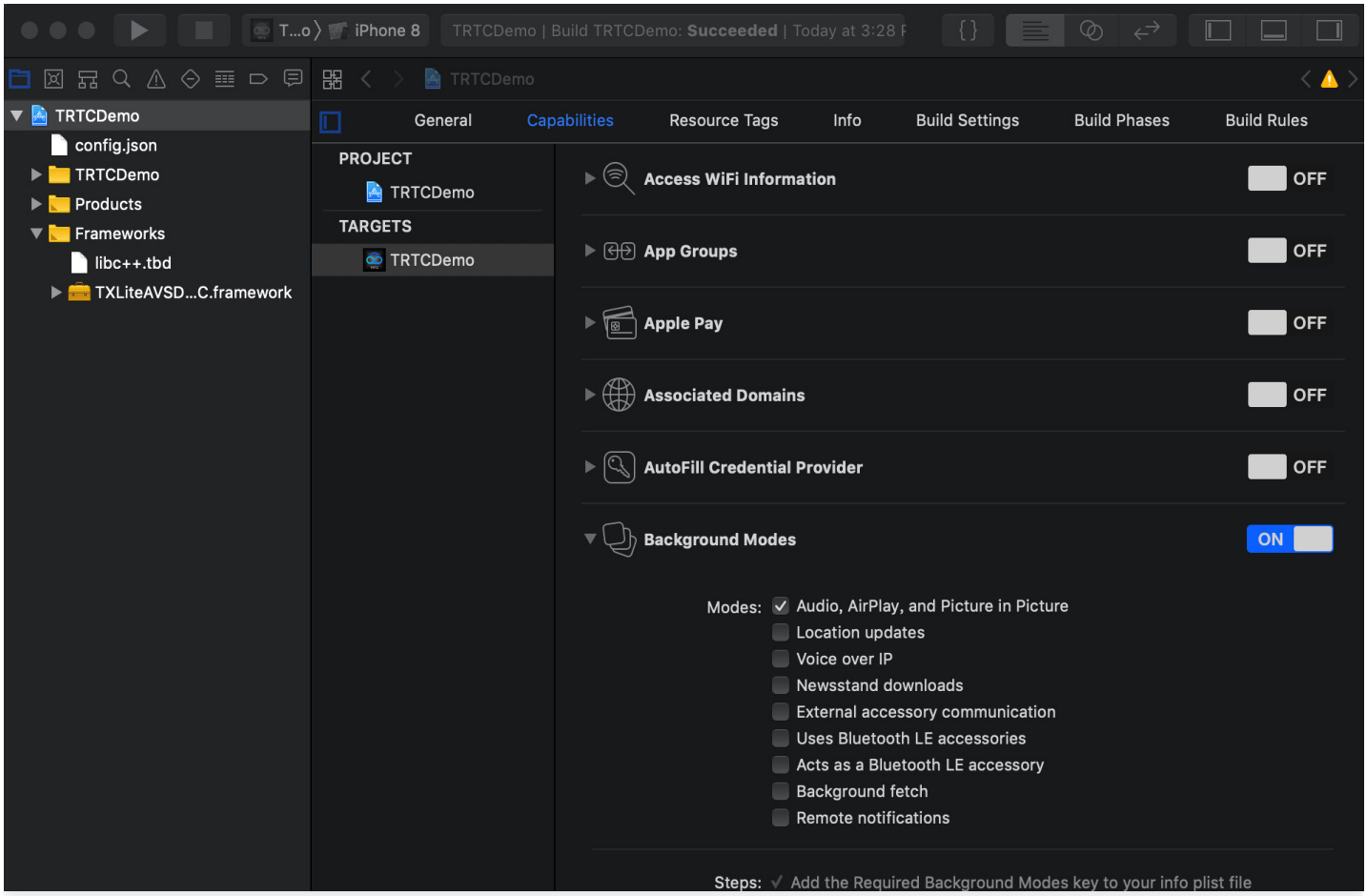
CocoaPods 如何安装?

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
```

TUICallKit 是否支持后台运行?

支持，如需要进入后台仍然运行相关功能，可选中当前工程项目，在 Capabilities 下的设置 Background Modes 打开为 ON，并勾选 Audio，AirPlay and Picture in Picture ，如下图所示：


T...o iPhone 8
TRTCDemo | Build TRTCDemo: Succeeded | Today at 3:28 F
TRTCDemo
TRTCDemo
config.json
TRTCDemo
Products
Frameworks
libc++.tbd
TXLiteAVSD...C.framework
General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
TRTCDemo
TARGETS
TRTCDemo
Access WiFi Information
OFF
App Groups
OFF
Apple Pay
OFF
Associated Domains
OFF
AutoFill Credential Provider
OFF
Background Modes
ON
Modes:
Audio, AirPlay, and Picture in Picture
Location updates
Voice over IP
Newsstand downloads
External accessory communication
Uses Bluetooth LE accessories
Acts as a Bluetooth LE accessory
Background fetch
Remote notifications
Steps:
Add the Required Background Modes key to your info plist file

uni-app（客户端）

最近更新时间：2023-07-28 14:48:24

uni-app 集成 TUICallKit 如何做离线唤醒？

建议您结合 TUIOfflinePush 插件 实现离线唤醒。

应用在后台时，不能自动将通话界面拉取到前台？

将应用从后台自动拉取到前台，需要检查 App 是否开启 “后台自动启动” 或 “悬浮窗” 权限。
值得注意的是，不同厂商、甚至同一厂商不同 Android 版本，其对于应用开放的权限以及权限名称也会存在不一致。例如，小米 6 只需开启后台弹出界面权限，而红米需要同时打开后台弹出界面和显示悬浮窗权限。

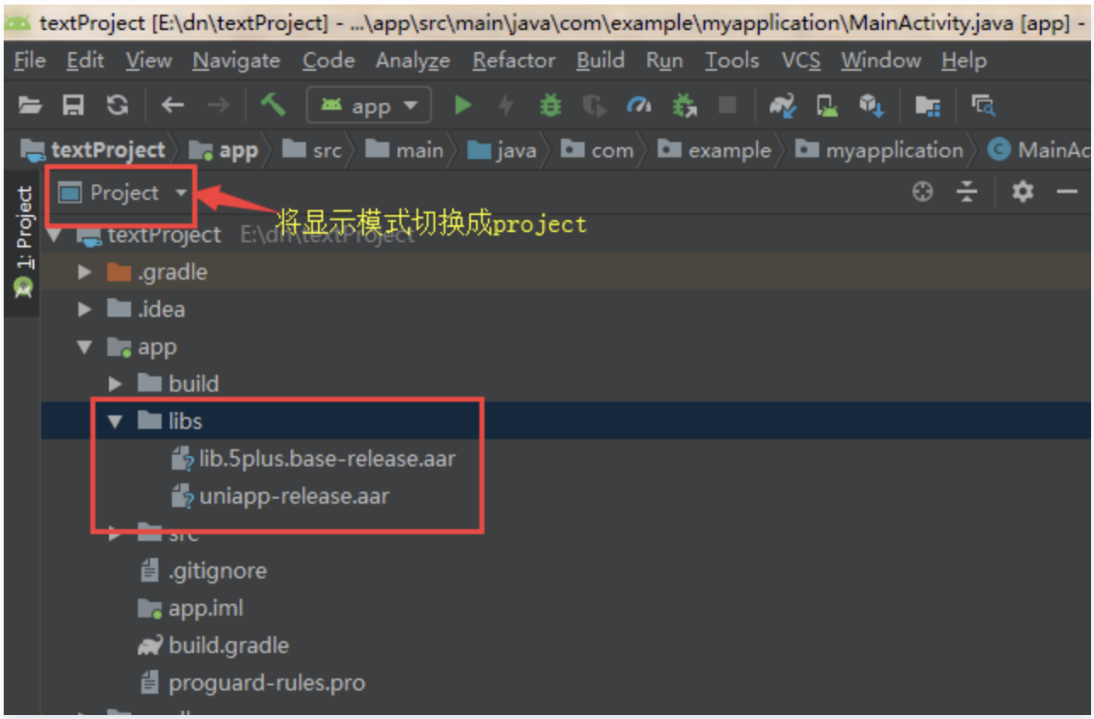
OPPO 手机的权限设置位置	小米手机的权限设置位置
	

不支持离线打包？

插件在 HBuilderX 开发工具中，现支持以下两种打包方式：

- 下载插件到本地，打包选择插件是本地的即可。
- 云端打包。

不支持直接将插件包放入 lib 文件夹，如下图所示：



uni-app（小程序）

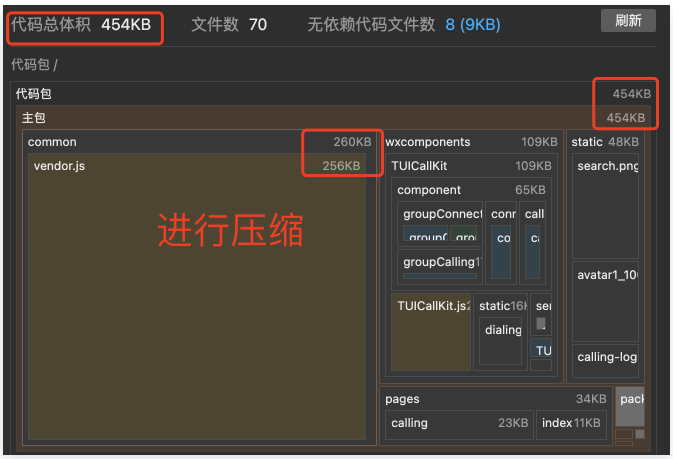
最近更新时间：2023-11-17 11:00:42

vue2 和 vue3 打包小程序包体积常见问题和方案

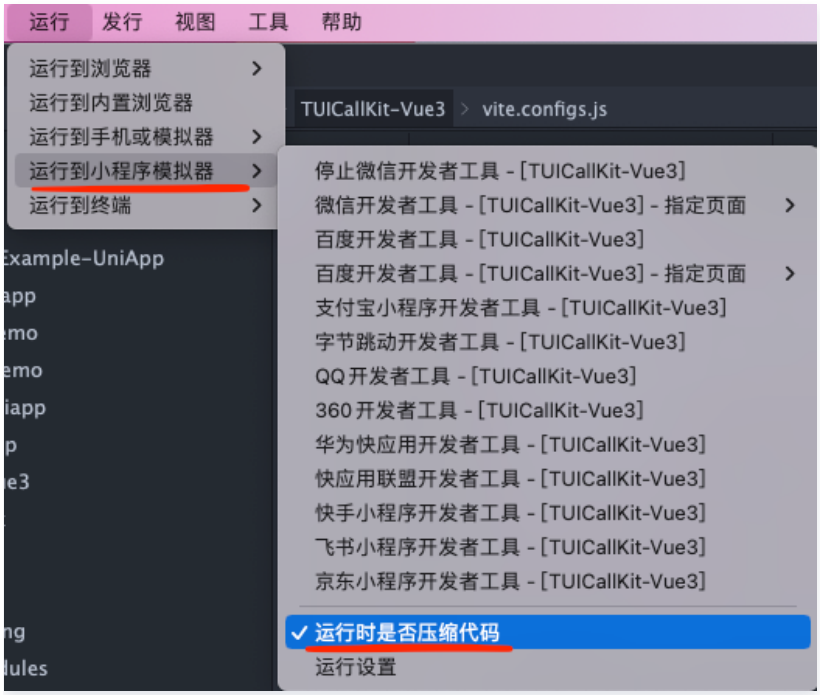
1. 是否压缩代码?

以 TUICallKit-Vue2 demo 举例，下图给出了是否勾选【运行时是否压缩代码】的效果对比（**851KB -> 454KB，降低 50%**）：





因此：HBuilder 中 **运行 > 运行到小程序模拟器 > 勾选 运行时是否压缩代码**，具体参考下图：



≤ v1.4.4

2. 依赖包如何不打入到主包的 `common/vendor.js`？

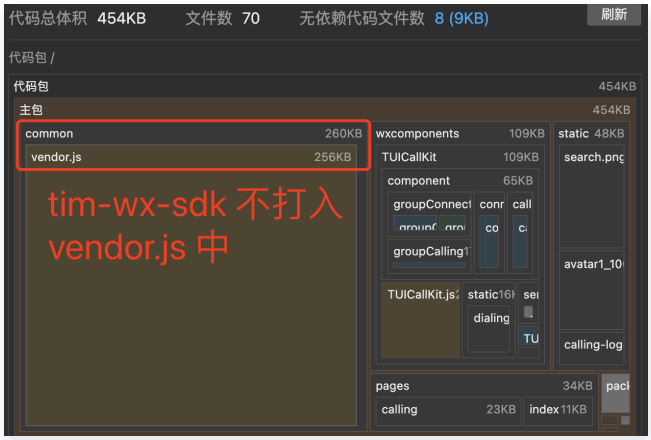
在使用 **微信开发者工具** 里 **构建 npm** 时，我们希望 HBuilder 打包小程序时，不将 `node_module` 中的依赖打入到主包的 `common/vendor.js` 中（**这会导致依赖存在多份，影响主包体积**）。

Vue2

推荐通过配置 vue.config.js 实现。

- vue.config.js 可选配置文件需放置在项目根目录下。
- vue.config.js 一般用于配置 webpack 等编译选项。HBuilder 采用 vue2 开发小程序，构建、打包时 vue.config.js 会被自动加载。

以 TUICallKit-Vue2 demo 举例，下图给出依赖包不打入主包的 common/vendor.js 的效果对比（805KB -> 256KB，降低 68%）：





具体使用如下：

1. 页面中通过 require 引入包依赖：

```
// 页面中引入方式必须使用 require，不能使用 import 的方式
const TIM = require('tim-wx-sdk');
```

2. vue.config.js 配置如下：

```
// vue.config.js 配置
module.exports = {
  configureWebpack: {
    externals: {
      'tim-wx-sdk': 'commonjs tim-wx-sdk', // 550KB 左右
    },
  }
}
```

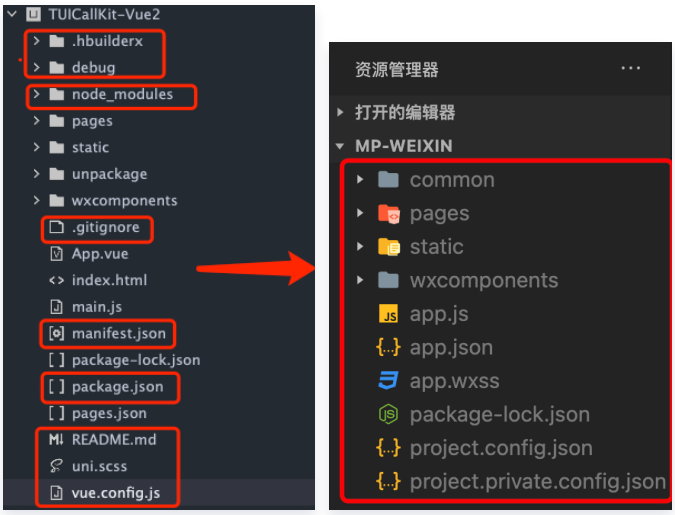
具体目录结构如下截图：



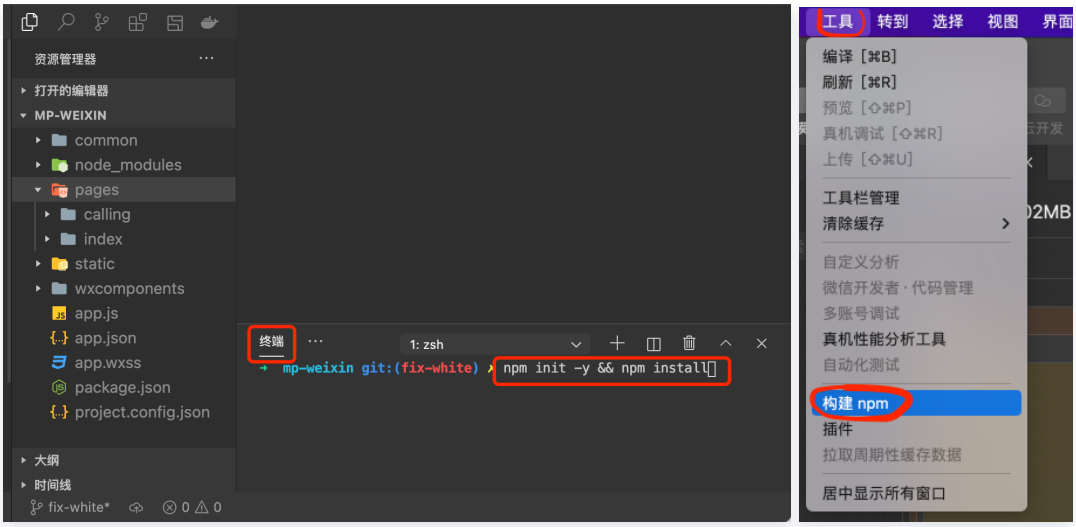
> ⚠ **注意：**
> **页面中引入依赖包的方式必须使用 require，不能使用 import 方式！**
> 原因：通过 import 引入时，HBuilder 打包小程序后，会将 import 引入方式进行编译。从而导致无法使用 node_modules 中的 tim-wx-sdk 。而 require 的方式则不会被编译。

3. 如何构建 npm?

1. 通过和 uni-app 的技术人员沟通了解到，目前 HBuilder 中不支持类似**微信开发者工具中的构建 npm**。而且他们也没有支持的计划。
2. HBuilder 中**运行 > 运行到小程序模拟器**生成的小程序项目，会清除 package.json 、 node_modules 等文件。具体如下图：



如果想在**微信开发者工具 > 工具 > 构建 npm**，就必须在**微信开发者工具的终端里**创建 package.json 、 npm install 安装依赖。然后才能**构建 npm**。具体如下：




```
npm init -y && npm install xx
```

4. 如何使用 copy-webpack-plugin 提升效率?

上面 3. 如何构建 npm 中，需要在 微信开发者工具 的终端里创建 package.json ，然后安装依赖才行。通过 copy-webpack-plugin 可减少该步骤，并提升效率。

以 TUICallKit-Vue2 demo 举例，下图给出使用 copy-webpack-plugin 的前后对比：



具体使用如下：

1. 安装 copy-webpack-plugin：

> ⚠ **注意：**
> **最新版本 copy-webpack-plugin 插件暂不兼容，推荐 v5.0.0（锁死版本）**

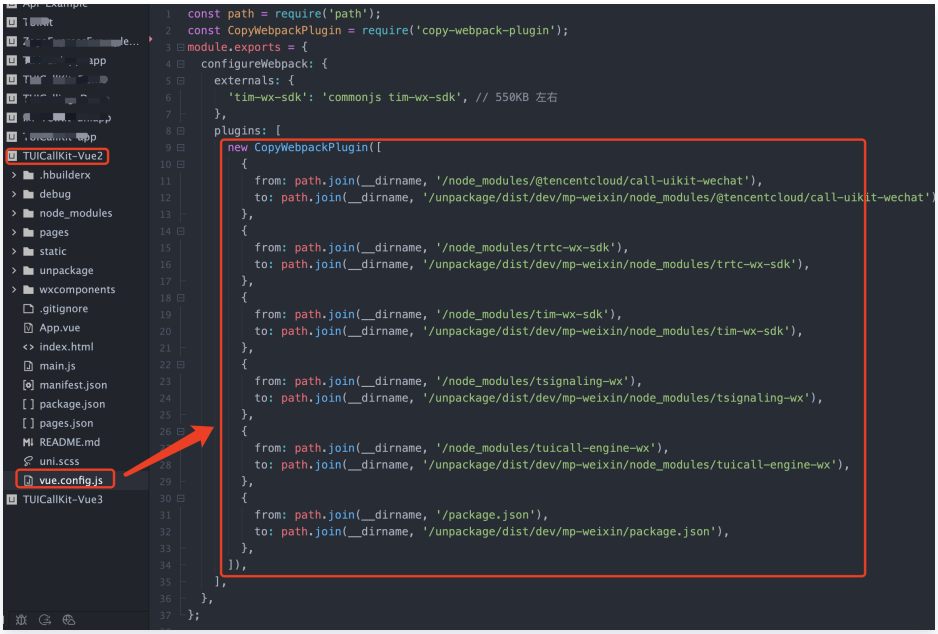
```
npm install -D copy-webpack-plugin@5.0.0
```

2. 在 vue.config.js 中使用 copy-webpack-plugin 插件，具体如下：

```
// vue.config.js 配置
const path = require('path');
const CopyWebpackPlugin = require('copy-webpack-plugin');
module.exports = {
```

```
  configureWebpack: {
    plugins: [
      new CopyWebpackPlugin([
        {
          from: path.join(__dirname, '/node_modules/@tencentcloud/call-uikit-wechat'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-
weixin/node_modules/@tencentcloud/call-uikit-wechat')
        },
        {
          from: path.join(__dirname, '/node_modules/trtc-wx-sdk'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-weixin/node_modules/trtc-wx-sdk')
        },
        {
          from: path.join(__dirname, '/node_modules/tim-wx-sdk'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-weixin/node_modules/tim-wx-sdk')
        },
        {
          from: path.join(__dirname, '/node_modules/tsignaling-wx'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-weixin/node_modules/tsignaling-
wx')
        },
        {
          from: path.join(__dirname, '/node_modules/tuicall-engine-wx'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-weixin/node_modules/tuicall-
engine-wx')
        },
        {
          from: path.join(__dirname, '/package.json'),
          to: path.join(__dirname, '/unpackage/dist/dev/mp-weixin/package.json')
        }
      ])
    ]
  }
}
```

具体目录结构如下截图：



Vue3

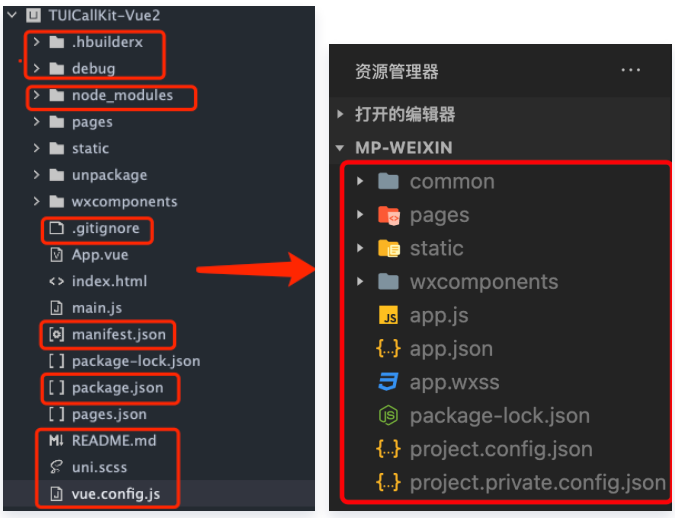
不用配置 vite.config.js ，只需通过 require 引入即可。具体如下：

```
// 页面中引入方式必须使用 require，不使用 import
const TIM = require('tim-wx-sdk');
```

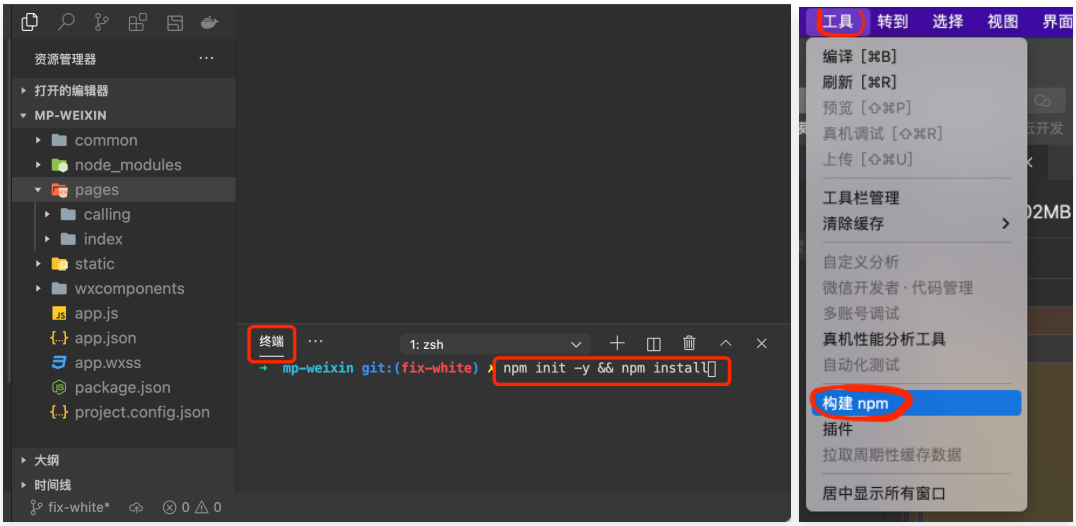
具体可参考：TUICallKit-Vue3 demo 。

如何构建 npm?

1. 通过和 uni-app 的技术人员沟通了解到，目前 HBuilder 中不支持类似**微信开发者工具**中的**构建 npm**。而且他们也没有支持的计划。
2. HBuilder 中**运行 > 运行到小程序模拟器**生成的小程序项目，会清除 package.json 、 node_modules 等文件。具体如下图：



如果想在**微信开发者工具 > 工具 > 构建 npm**，就必须在**微信开发者工具的终端里**创建 package.json 、 npm install 安装依赖。然后才能**构建 npm**。具体如下：



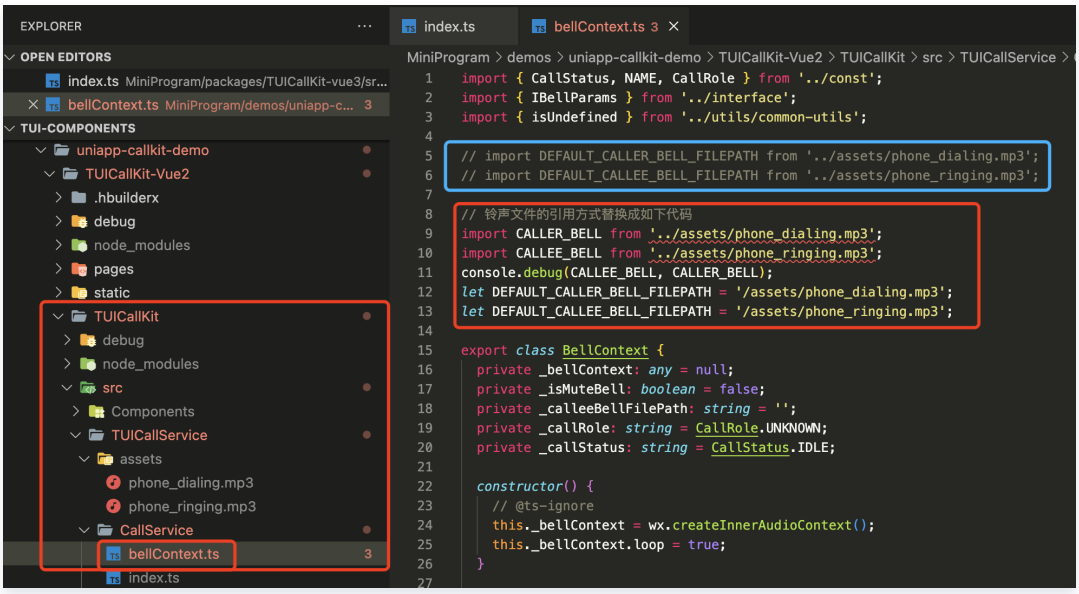
```
npm init -y && npm install tuicall-engine-wx@1.5.6
```

Vue2 uni-app 打包微信小程序项目中，铃声没有声音?

这个是因为在 vue2 中，uni-app 打包，将铃声文件编码为 base64 编码，这个编码在播放铃声的 api 那里是不支持的。
您可采取如下步骤解决该问题：

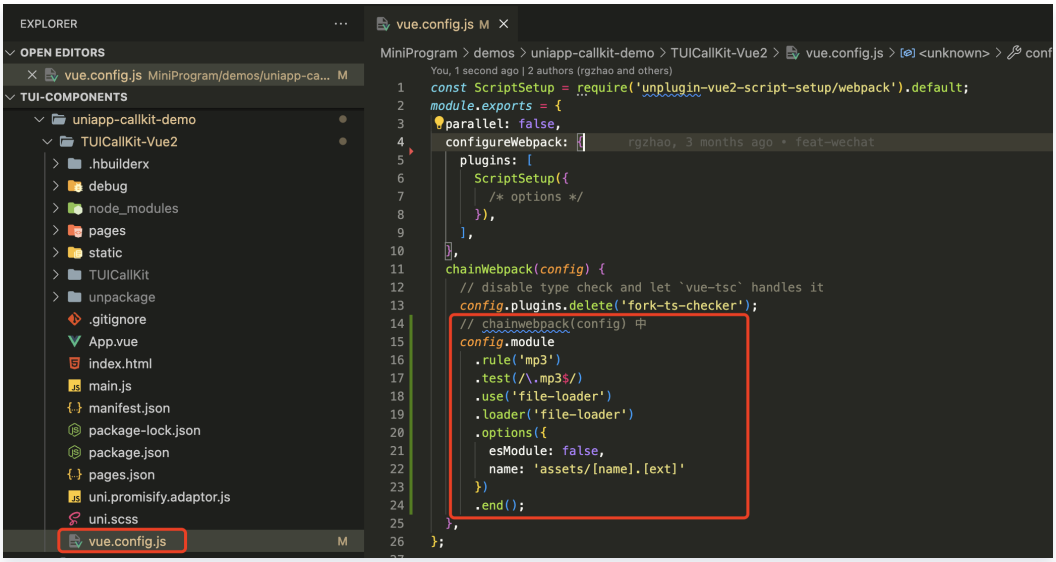
- 第一步，在项目的 TUICallKit 源码中，修改铃声的引入方式。

```
// 铃声文件的引用方式替换成如下代码
import CALLER_BELL from '../assets/phone_dialing.mp3';
import CALLEE_BELL from '../assets/phone_ringing.mp3';
console.debug(CALLEE_BELL, CALLER_BELL);
let DEFAULT_CALLER_BELL_FILEPATH = '/assets/phone_dialing.mp3';
let DEFAULT_CALLEE_BELL_FILEPATH = '/assets/phone_ringing.mp3';
```



- 第二步，修改项目中的 vue.config.js 文件。

```
// chainwebpack(config) 中
config.module
    .rule('mp3')
    .test(/\.mp3$/)
    .use('file-loader')
    .loader('file-loader')
    .options({
     esModule: false,
     name: 'assets/[name].[ext]'
    })
    .end();
```


EXPLORER
OPEN EDITORS
vue.config.js MiniProgram/demos/uniapp-ca... M
TUI-COMPONENTS
uniapp-callkit-demo
TUICallKit-Vue2
.hbuilderx
debug
node_modules
pages
static
TUICallKit
unpackage
.gitignore
App.vue
index.html
main.js
manifest.json
package-lock.json
package.json
pages.json
uni.promisify.adaptor.js
uni.scss
vue.config.js M
vue.config.js M
MiniProgram > demos > uniapp-callkit-demo > TUICallKit-Vue2 > vue.config.js > <unknown> > conf
You, 1 second ago | 2 authors (rgzhao and others)
const ScriptSetup = require('unplugin-vue2-script-setup/webpack').default;
module.exports = {
parallel: false,
configureWebpack: {
rgzhao, 3 months ago • feat-wechat
plugins: [
ScriptSetup({
/* options */
}),
],
},
chainWebpack(config) {
// disable type check and let `vue-tsc` handles it
config.plugins.delete('fork-ts-checker');
// chainwebpack(config) 中
config.module
.rule('mp3')
.test(/\.mp3$/)
.use('file-loader')
.loader('file-loader')
.options({
esModule: false,
name: 'assets/[name].[ext]'
})
.end();
},
};

微信小程序

最近更新时间：2023-06-29 16:36:51

> **说明：**
> 如果并没有解决您的问题，也欢迎您加入我们的 TUICallKit 技术交流 QQ 群： **605115878**，来进行交流讨论。

是否支持 uniapp、Taro 等开发环境?

Taro 暂不支持，uniapp 请参见 快速集成方案（uni-app）。

如何全局监听，在所有页面都可以唤起通话界面?

> **注意：**
> **TUICallKit** 版本须大于或等于 1.2.1。

仅需四步即可完成小程序全局监听：

1. 在 app.json 注册页面全局监听页面。
2. 在 app.js 中导入全局监听模块（callManager）。
3. 在 app.js onLaunch 生命周期中实例化 callManager。
4. 触发全局监听。

具体实现方法，请参见：

- **主包方案：** 【TUICallKit】小程序全局监听（主包方案）
- **分包方案：** 【TUICallKit】小程序全局监听（分包方案）

> **说明**
> - 分包适用于业务逻辑复杂，按需载入的小程序。
> - 如果您不需要使用分包接入方案 可以直接使用主包接入。

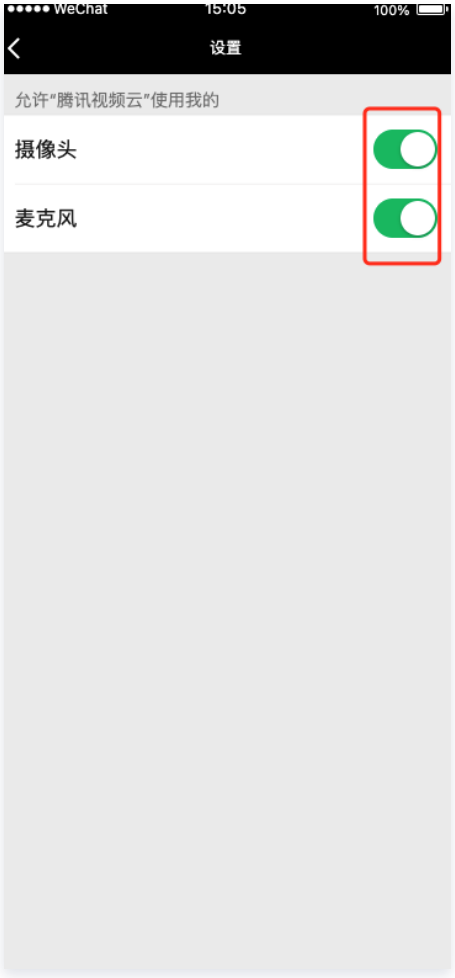
当摄像头/麦克风权限被拒绝，如何处理?

首次进入小程序需要默认是**没有设备权限**的，需要手动开启权限。

1. 打开小程序的设置。



2. 开启摄像头/麦克风权限。



报错 "authorize:fail:appid privacy api banned"，如何处理?

由于 TRTC 涉及到麦克风与摄像头的权限问题，需要用户在小程序控制台中更新隐私保护政策，具体步骤如下：

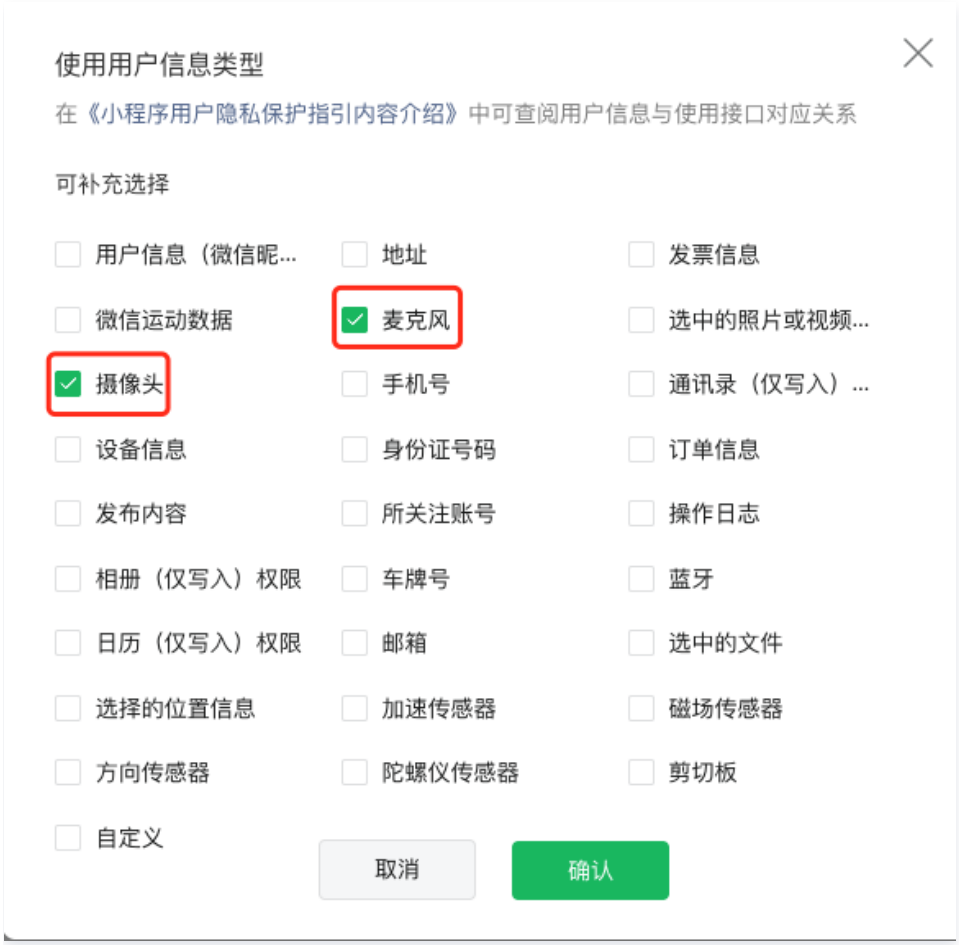
1. 登录腾讯平台的小程序后台，点击设置栏目。



2. 选择用户隐私保护指引设置功能，点击更新。



3. 在弹出的表格中，勾选小程序会涉及到的类型。



Flutter

最近更新时间：2023-08-18 16:20:31

1、同时集成 tencent_calls_uikit 和 tencent_trtc_cloud，或同时集成 tencent_calls_uikit 和 live_flutter_plugin，出现符号冲突报错，怎么解决?

问题详情：当引入 flutter "tencent_calls_uikit" 进我们现有的 project 后，Android 构建 APK 时出现如下报错：

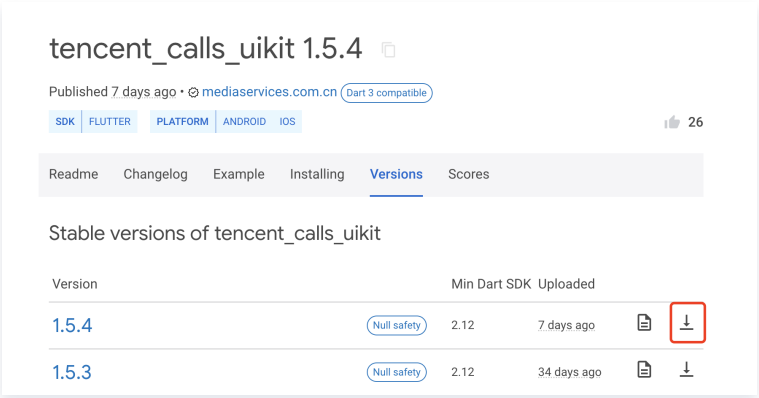
```
Duplicate class com.tencent.liteav.LiveSettingJni found in modules jetified-LiteAVSDK_Professional-10.7.0.13053-runtime (com.tencent.liteav:LiteAVSDK_Professional:10.7.0.13053) and jetified-LiteAVSDK_TRTC-10.3.0.11225-runtime (com.tencent.liteav:LiteAVSDK_TRTC:10.3.0.11225)
```

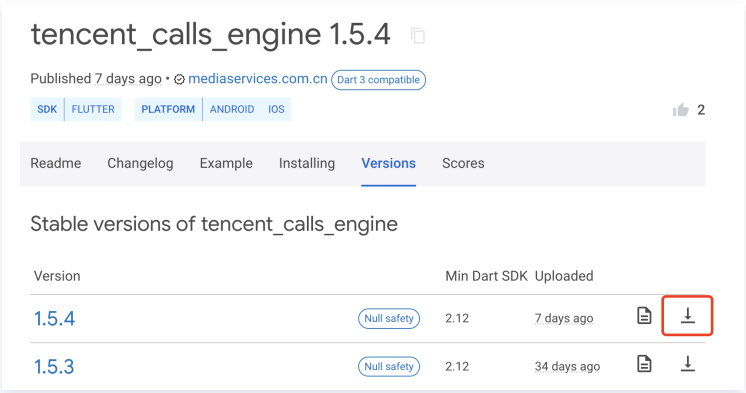
iOS 在执行 `pod install` 出现如下报错：

```
[!] The 'Pods-Runner' target has frameworks with conflicting names: txsoundtouch.xcframework and txffmpeg.xcframework.
```

这个问题原因是 tencent_calls_uikit 和 tencent_trtc_cloud 分别依赖于 TRTC Android SDK 的专业版和精简版（这个问题我们会在后续版本逐步统一），目前可以通过本地集成的方式来修复这个问题，具体步骤如下：

1. **分别进入 tencent_calls_uikit 和 tencent_calls_engine 的版本管理页面，单击下载按钮 tencent_calls_uikit 依赖于 tencent_calls_engine，等待下载完成后，解压文件。**





2. **修改 tencent_calls_engine 插件中 TRTC SDK 依赖为专业版本。**
   - **Android：进入 `tencent_calls_engine` 解压后的目录，修改 `android/build.gradle` 文件中的依赖为报错信息中呈现的：`com.tencent.liteav:LiteAVSDK_Professional:10.7.0.13053` 。**

```
}

dependencies{
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    api 'com.google.code.gson:gson:2.9.1'
    api "com.tencent.liteav.tuikit:tuicore:1.1.16"
    api 'com.tencent.imsdk:imsdk-plus:6.8.3374'
    api "com.tencent.liteav:LiteAVSDK_TRTC:10.3.0.11225"
    api 'com.tencent.liteav.tuikit:tuicallengine:1.3.0.205'
}
```

- iOS：首先进入 tencent_calls_engine 解压后的目录，修改 ios/tencent_calls_engine.podspec 文件中的依赖为：s.dependency 'TUICallEngine/Professional', '1.5.1.310'；

```
A new Flutter plugin project.
                    DESC
  s.homepage         = 'http://example.com'
  s.license          = { :file => '../LICENSE' }
  s.author           = { 'Your Company' => 'email@example.com' }
  s.source           = { :path => '.' }
  s.source_files = 'Classes/**/*'
  s.dependency 'Flutter'
  s.dependency 'TUICallEngine/Professional', '1.5.1.310'
  s.platform = :ios, '9.0'
  s.static_framework = true
  # Flutter.framework does not contain a i386 slice.
  s.pod_target_xcconfig = { 'DEFINES_MODULE' => 'YES', 'EXCLUDED_ARCHS[sdk=iphonesimulator*]' => 'i386' }
  s.swift_version = '5.0'
```

再进入 tencent_calls_uikit 解压后的目录，修改 ios/tencent_calls_uikit.podspec 文件中的依赖为：s.dependency 'TUICallKit/Professional', '1.5.1.310'。

```
A new Flutter plugin project.
                    DESC
  s.homepage         = 'http://example.com'
  s.license          = { :file => '../LICENSE' }
  s.author           = { 'Your Company' => 'email@example.com' }
  s.source           = { :path => '.' }
  s.source_files = 'Classes/**/*'
  s.dependency 'Flutter'
  s.dependency 'TUICallKit/Professional' '1.5.1.310'
  s.platform = :ios, '9.0'
  s.static_framework = true
  # Flutter.framework does not contain a i386 slice.
  s.pod_target_xcconfig = { 'DEFINES_MODULE' => 'YES', 'EXCLUDED_ARCHS[sdk=iphonesimulator*]' => 'i386' }
  s.swift_version = '5.0'
end
```

3. 修改 tencent_calls_uikit 工程中的 pubspec.yaml，将 tencent_calls_engine 的依赖调整为本地依赖，示例如下：

```
tencent_calls_engine:
  path: tencent_calls_engine/  # 这里的目录对应您刚才下载的插件目录
```

4. 修改您工程中的 pubspec.yaml，将 tencent_calls_uikit 的依赖修改为本地依赖，示例如下：

```
tencent_calls_uikit:
  path: tencent_calls_uikit/  # 这里的目录对应您刚才下载的插件目录
```

版权所有：腾讯云计算（北京）有限责任公司

2、Flutter Android未添加混淆设置，怎么设置?

如果您需要编译运行在 Android 平台，由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 proguard-rules.pro 文件中添加如下代码：

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

3、从1.8.0以下的版本升级到1.8.0及以上的版本，导致编译报错或者拉不起页面问题如何修复?

如果是从1.8.0以下升级到1.8.0及以上的版本，需要您检查下面几个步骤是否正常：

1. 将 navigatorObservers 添加到 MateriaApp。 目的是在收到呼叫邀请时导航到 TUICallKit 页面。示例代码如下：

```
import 'package:tencent_calls_uikit/tuicall_kit.dart';

MaterialApp (
    navigatorObservers：[TUICallKit.navigatorObserver]，
    ...
)
```

2. tencent_calls_engine 插件中的导入文件统一替换为新的。

```
import 'package:tencent_calls_engine/tuicall_engine.dart';
import 'package:tencent_calls_engine/tuicall_observer.dart';
import 'package:tencent_calls_engine/tuicall_define.dart';
```

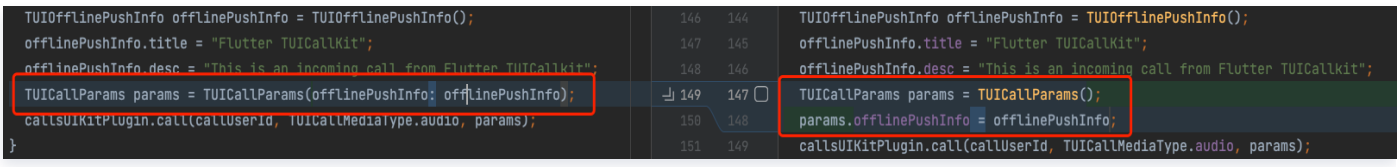
上图内容替换为下图所示：

```
import 'package:tencent_calls_engine/tencent_calls_engine.dart';
```

3. 登录 API 调整更规范，不需要再指定参数。



4. 离线推送参数构造优化。



多人音视频
组件介绍 (TUIRoomKit)

最近更新时间：2024-01-01 13:30:41

组件介绍

多人音视频房间 SDK（TUIRoomKit）是一款定位**企业会议、网络研讨会、在线教育**等多人音视频会话场景的含 UI 低代码组件，通过集成该组件，仅需三步一天内就可以为您的 App 添加多人音视频会话功能，快速上线业务。产品功能展示如下图：



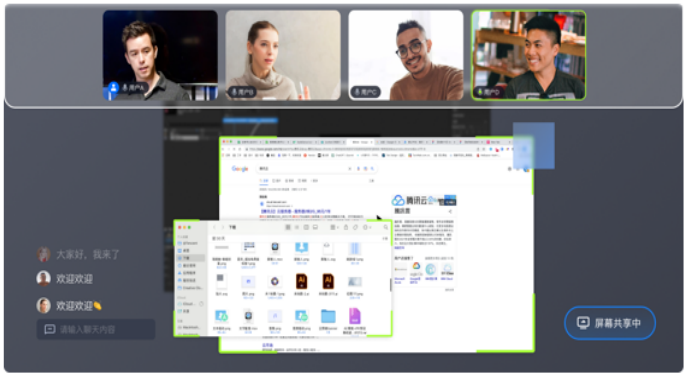
功能优势

- 快速集成：提供带 UI 的开源组件，仅需修改几行代码，即可快速上线多人音视频会话功能，节省 90% 开发时间。
- 多平台全球互通：支持 iOS、Android、Web、Electron、Windows、uni-app（小程序）等平台的 TUIRoomKit 组件入会，相互邀请入会等功能，互联互通。
- 多人音视频会话场景深度定制：提供房间管理、成员管理、会议悬浮窗等专属功能，满足在线会议、网络研讨会等多种场景需求。
- 含 UI 灵活定制：不仅提供标准会议 UI 的开源组件，可根据业务场景灵活修改以匹配需求。

版权所有：腾讯云计算（北京）有限责任公司

应用场景

多人进行音视频会话互动，覆盖企业会议、网络研讨会、在线教育、在线招聘、语聊房等场景。



音视频会议



在线教育



企业培训



Webinar

开发环境

多人音视频房间 SDK（TUIRoomKit）具体平台支持及开发环境要求如下表所示：

平台	开发环境要求
iOS	支持 iOS 12.0 及以上版本的 iPhone 或 iPad 真机 项目已配置有效的开发者签名
Android	最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本 Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本） Android 4.4 及以上的手机设备
Windows	Visual Studio 2015及以上集成开发环境 QT5.9.1及以上版本的 Qt 开发库 VS 下的 QT 开发插件 Qt Visual Studio Tools 2.2.0及以上 最低支持系统：Windows 8.1 请确保您的集成开发环境能够正常开发

版权所有：腾讯云计算（北京）有限责任公司

Web	推荐使用桌面端 Chrome 56+，详细开发环境要求请参见 支持的平台
Electron	支持 Windows 7 及以上版本、Mac OS 10.10 及以上版本 支持 Electron 4.0.0 以上版本，推荐使用最新版 Electron SDK
uni-app（小程序）	微信 App iOS 最低版本要求：7.0.9 微信 App Android 最低版本要求：7.0.8 小程序基础库最低版本要求：2.13.0

功能及计费说明

多人音视频房间 SDK（TUIRoomKit）基于即时通信 IM 和实时音视频 TRTC 底层技术封装实现，从而实现多人音视频互动场景下的各项服务。因此多人音视频 SDK 费用包含购买即时通信 IM 的费用及开通实时音视频 TRTC 包月套餐包费用两部分，其中购买 TRTC 包月套餐包会赠送等价格实时音视频 TRTC 时长补充包，赠送时长包按月赠送，仅在当月生效。

多人音视频房间 SDK 预付费套餐包包括版本：**体验版、基础互动版、进阶互动版、超大房间互动版**，各版本说明及价格如下图及表所示：

对比项		体验版	基础互动版	进阶互动版	超大房间互动版
价格	折扣价	0元/7天	2999元/月 立即选购	6499元/月 立即选购	7999元/月 立即选购
	刊例价	~~0元/7天~~	~~3374元/月~~	~~7749元/月~~	~~9249元/月~~
套餐包组成	实时音视频 TRTC 套餐包	体验版 0元/7天（免费体验）	尊享版	旗舰版	旗舰版
	即时通信 IM 套餐包	搭配即时通信 IM 任意套餐包版本体验	专业版	专业版	旗舰版
赠送资源（超出后付费）	通话时长	-	380,000分钟/月	1,400,000分钟/月	1,400,000分钟/月
	日活跃用户数（DAU）	根据 IM 版本而定	10,000 DAU	10,000 DAU	10,000 DAU
	峰值群组数	根据 IM 版本而定	100,000个/月	100,000个/月	100,000个/月
基本功能	参会人数规模	20	50	200	300 （如需扩容请 工单联系）
	同时开启视频人数	20	50	50	50
	视频清晰度设置	720P/1080P/2K/4K	720P/1080P	720P/1080P/2K/4K	720P/1080P/2K/4K
视图	标准会议 UI	✓	✓	✓	✓
	会议悬浮窗	✓	✓	✓	✓
	自定义 UI	✓	✓	✓	✓
用户状态	登录/登出	✓	✓	✓	✓

	设置个人信息	✓	✓	✓	✓
	获得个人信息	✓	✓	✓	✓
房间管理	创建/销毁/加入/退出房间	✓	✓	✓	✓
	获取房间信息详情	✓	✓	✓	✓
	房间信息自定义	✓	✓	✓	✓
	房间内文本消息	✓	✓	✓	✓
	多人视频通话	✓	✓	✓	✓
	多人语音通话	✓	✓	✓	✓
	摄像头打开/关闭	✓	✓	✓	✓
	麦克风打开/关闭	✓	✓	✓	✓
	麦克风音量调整	✓	✓	✓	✓
	监听回调	✓	✓	✓	✓
成员管理	获取成员列表	✓	✓	✓	✓
	切换成员角色	✓	✓	✓	✓
	单人禁画/禁麦/禁言	✓	✓	✓	✓
	全体禁画/禁麦/禁言	✓	✓	✓	✓
	踢人出房（会）	✓	✓	✓	✓
账号管理	多端登录会议	仅配合IM旗舰版使用	–	–	✓
	同平台多端登录会议	仅配合IM旗舰版使用	–	–	✓
麦位管理	无序/有序麦位	✓	–	✓	✓
	请求发言	✓	–	✓	✓
	自定义麦位数量	✓	–	✓	✓
	麦位邀请/锁定/踢下	✓	–	✓	✓
互动功能	屏幕分享	✓	✓	✓	✓
	AI 降噪	✓	✓	✓	✓
	音频默认混流	✓	✓	✓	✓
即时通信相关功能		可配合 IM 不同版本试用	详见 IM 专业版功能	详见 IM 专业版功能	详见 IM 旗舰版功能
超出赠送资源部分后付费		通话时长后付费价格详见 音视频时长计费说明，DAU 与峰值群组数后付费价格详见 IM 计费说明			

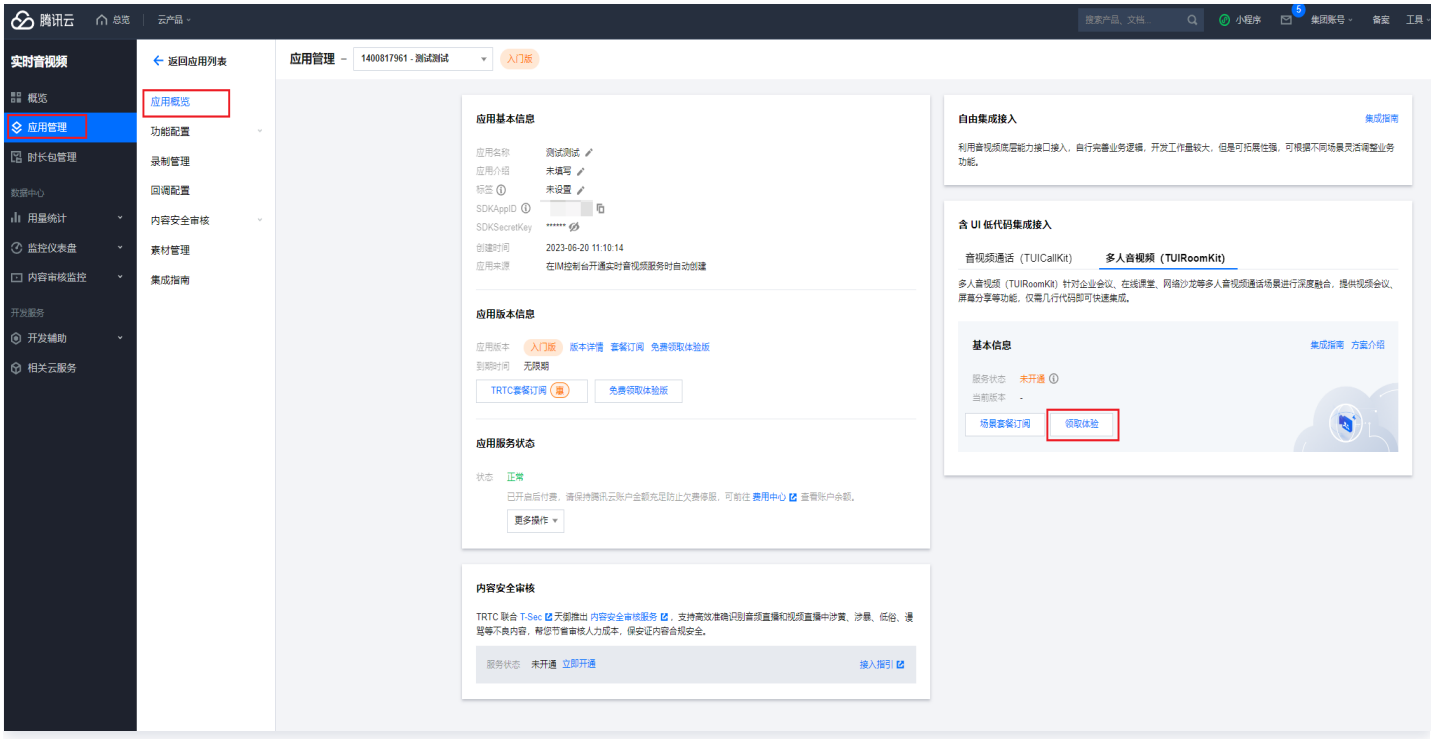
支持平台	iOS、Android、Web、Windows、Electron、uni-app（小程序）

开通 TUIRoomKit 体验版

为了您能更好地体验多人音视频会话功能，我们免费为每个 SDKAppID 提供了TUIRoomKit 7天体验版（体验版不额外赠送通话时长），每个 SDKAppID 可免费体验2次，每次有效期均为7天；同时一个账号下所有 SDKAppID 的体验总次数为10次。

您可以在 TRTC 控制台中开通 TUIRoomKit 体验版，具体操作步骤如下：

1. 访问 TRTC 控制台应用管理，点击需要使用 TUIRoomKit 应用（SDKAppID）后的**详情**，进入应用概览页面。
2. 找到页面右下角的**含UI低代码集成接入 > 多人音视频（TUIRoomKit）区域**，单击卡片内的 **领取体验** 即可开通 TUIRoomKit 的 7 天免费试用服务。



3. 体验完成后，单击 场景套餐订阅 购买正式版 TUIRoomKit 。

> **说明：**
> 开通 TUIRoomKit 的体验版将默认开通 TRTC 包月套餐的体验版，二者体验版开通次数和规则相同。

TUIRoomKit 下载

Android TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github 快速接入 API 参考 常见问题



iOS TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github　快速接入　API 参考　常见问题



Web TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github　快速接入　API 参考　常见问题



Windows TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github　快速接入



Electron TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github　快速接入　API 参考



uni-app（小程序）TUIRoomKit

经典会议 UI，支持音视频沟通、屏幕分享、聊天弹幕、禁麦禁画、音乐伴奏等多个特性，功能全面，接入简单。

Github　快速接入　API 参考

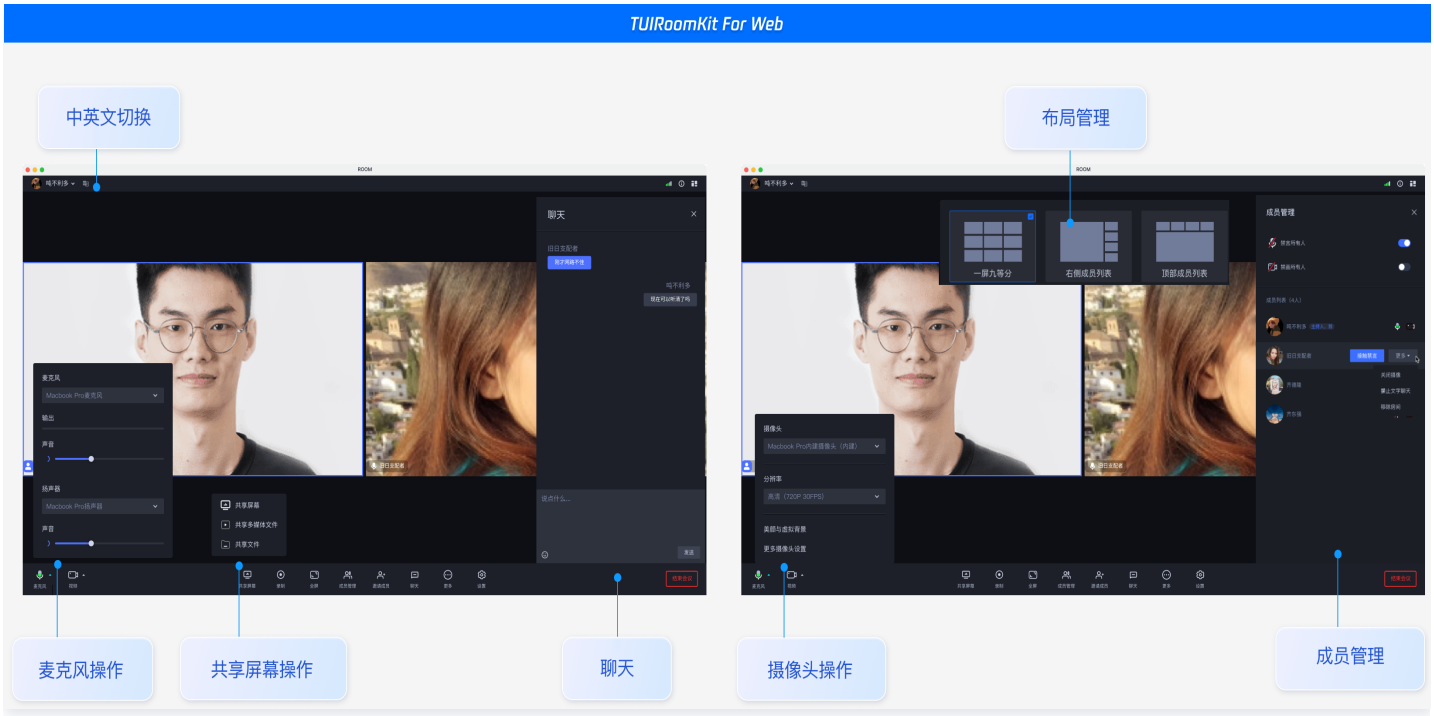
开源建设

说明：
如果您需要访问之前的TUIRoom组件，您可以在 这里 找到升级前的 TUIRoom 开源项目。

快速接入 (TUIRoomKit)
Web & H5 (Vue2/Vue3)

最近更新时间：2024-01-11 15:42:51

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理，音视频控制，屏幕共享，成员管理，麦位管理，即时聊天，自定义布局切换等丰富的功能交互，同时支持中英文切换，一键换肤等能力。



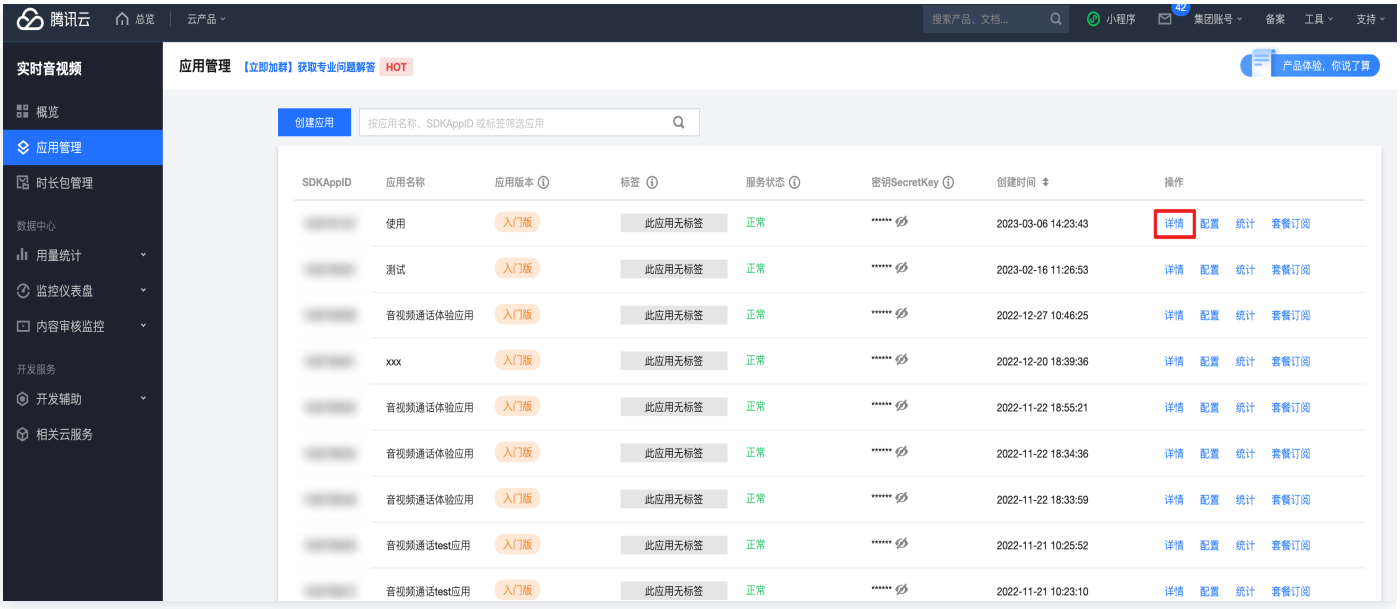


本文介绍 TUIRoomKit (Web & H5) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。
您可以单击 TUIRoomKit 在线体验链接 体验 TUIRoomKit 更多功能。
您可以单击 Github/Gitee 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Web 示例工程。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），点击详情按钮，进入应用概览界面。



2. 在应用概览页面找到含 UI 低代码集成接入 卡片，选择多人音视频（TUIRoomKit），点击领取体验按钮，领取7天体验版

TUIRoomKit 进行接入测试。

> **注意：**
> - 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
> - 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
> - 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请点击**场景套餐订阅**按钮或前往 购买页 购买；

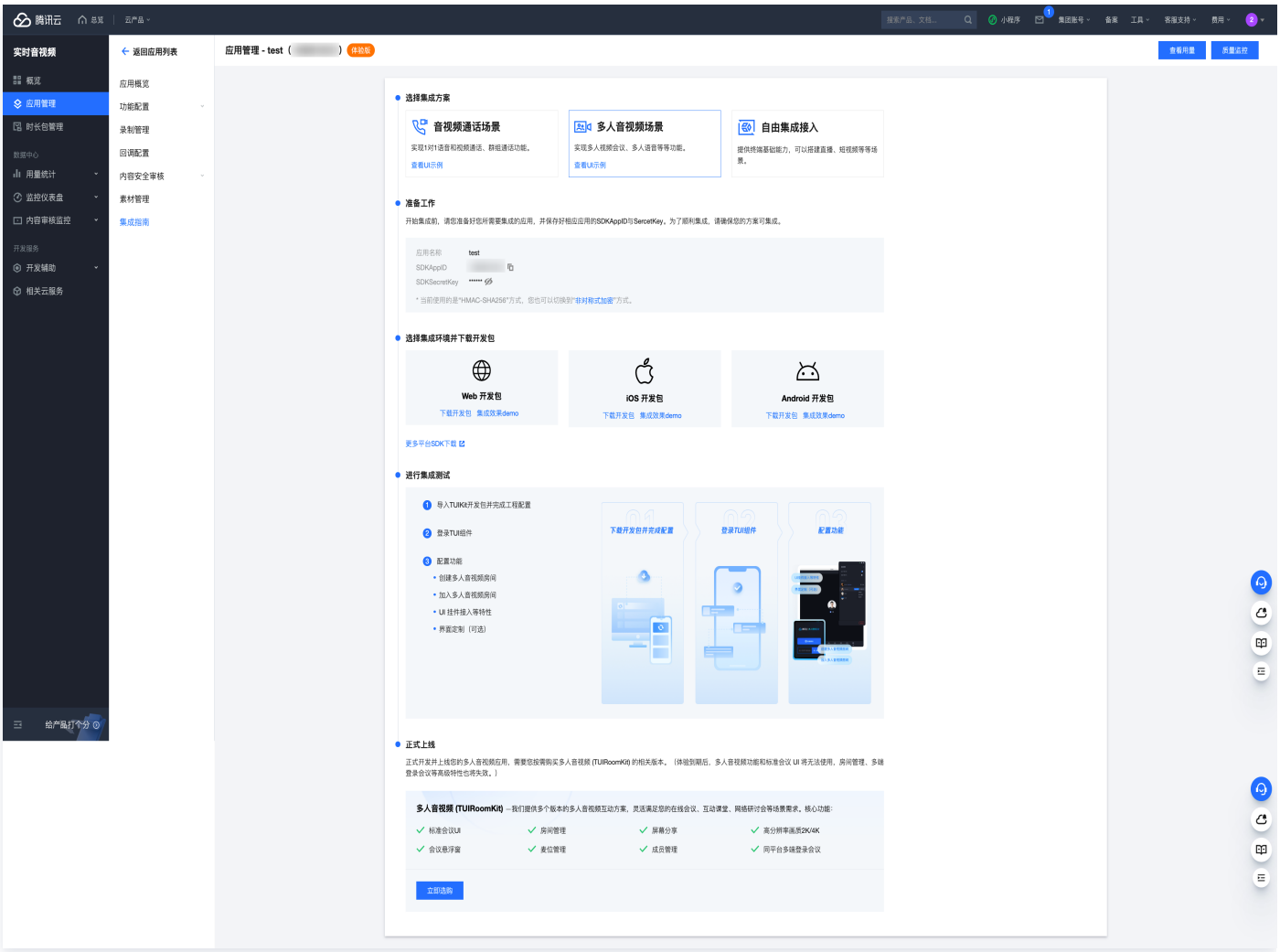




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



4. 单击**集成指南**，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：准备 Vue 工程代码

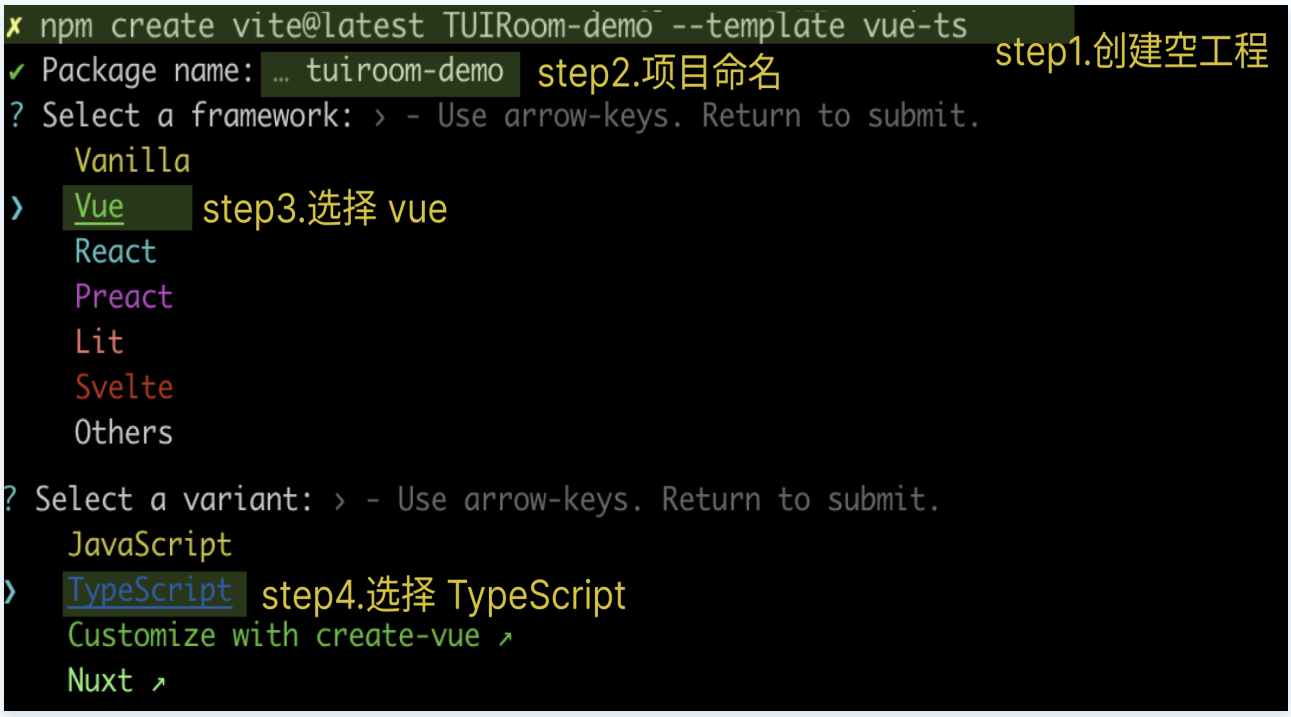
打开您已有 Vue 工程，可跳过该步骤。
如果无 Vue 项目，可选择以下方式生成模版工程。

生成 Vue3 + Vite + TS 模版工程

注意：
Vite 需要 Node.js 版本 14.18+，16+。然而，有些模板需要依赖更高的 Node 版本才能正常运行，当您的包管理器发出警告时，请注意升级您的 Node 版本。

```
# npm 6.x
npm create vite@latest TUIRoom-demo --template vue-ts

# npm 7+, extra double-dash is needed:
npm create vite@latest TUIRoom-demo -- --template vue-ts

# yarn
yarn create vite TUIRoom-demo --template vue-ts

# pnpm
pnpm create vite TUIRoom-demo --template vue-ts
```

说明：
以 npm6.x 的创建方式为例，生成模板工程操作流程图如下：（其中标记高亮区域为需要输入或选择）



成功生成 Vue3 + Vite + TS 模板工程后，执行以下脚本：

```
cd TUIRoom-demo
npm install
npm run dev
```

生成 Vue3 + Webpack + TS 模版工程

```
// 安装 vue-cli，注意 Vue CLI 4.x 要求 Node.js 为 v10 以上版本
npm install -g @vue/cli
// 创建 Vue3 + Webpack + TS 模版工程
vue create tuiroomkit-demo
```

> ⚠ **注意：**
> 执行生成模板工程脚本的过程中，生成模版的方式选择 Manually select features，其余配置选项参考图片。

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, TS, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected
polyfills, transpiling JSX)? Yes
? Pick a linter / formatter config: Standard
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated
config files
? Save this as a preset for future projects? No
```

成功生成 Vue3 + Webpack + TS 模板工程后，执行以下脚本：

```
cd tuiroomkit-demo
npm run serve
```

生成 Vue2 + Webpack + JS 模版工程

```
// 安装 vue-cli，注意 Vue CLI 4.x 要求 Node.js 为 v10 以上版本
npm install -g @vue/cli
// 创建 Vue2 + Webpack + Js 模版工程
vue create tuiroomkit-demo
```

> ⚠ **注意：**
> 执行生成模版工程脚本的过程中，生成模版的方式选择 `Default ([Vue 2] babel, eslint)` 。

```
? Please pick a preset:
  Default ([Vue 3] babel, eslint)
❯ Default ([Vue 2] babel, eslint)
  Manually select features
```

成功生成 Vue2 + Webpack + JS 模版工程后，执行以下脚本：

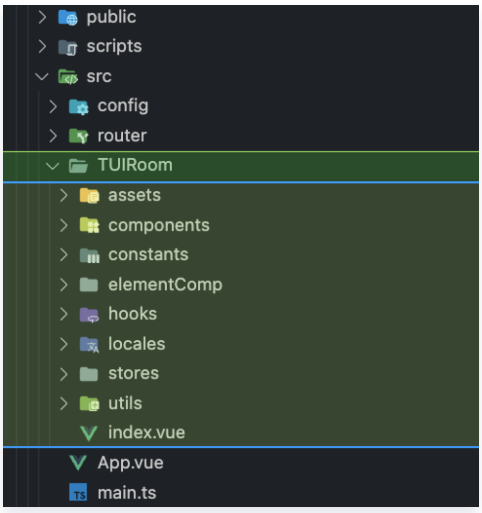
```
cd tuiroomkit-demo
npm run serve
```

步骤三：下载并引用 TUIRoom 组件

1. **下载 TUIRoom 组件代码**
   单击 Github，克隆或下载 TUIRoomKit 仓库代码。
2. **引用 TUIRoom 组件**

Vue3 项目引入 TUIRoom 组件

- 复制 `TUIRoomKit/Web/vue3/src/TUIRoom` 文件夹到已有项目 src/ 目录下，引用成功后目录结果如图所示。



- 在页面中引用 TUIRoom 组件。例如：在 `App.vue` 组件中引入 TUIRoom 组件。
  - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
  - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

> ⚠ **注意：**
> 在页面中复制以下代码之后，需要修改 TUIRoom 接口的参数为实际数据。

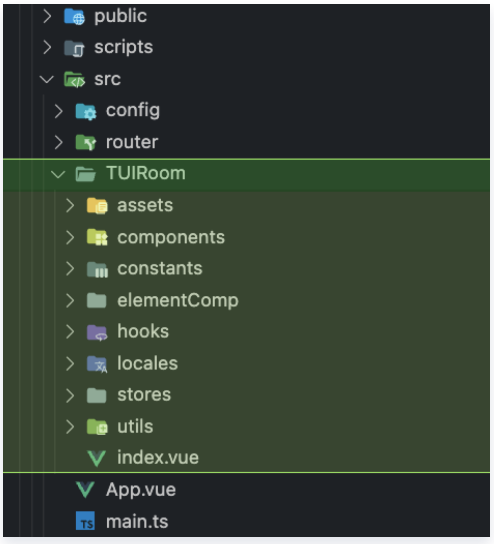
```
<template>
  <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
```

```
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
const TUIRoomRef = ref();

onMounted(async () => {
  // 初始化 TUIRoom 组件
  // 主持人在创建房间前需要先初始化 TUIRoom 组件
  // 普通成员在进入房间前需要先初始化 TUIRoom 组件
  await TUIRoomRef.value.init({
    // 获取 sdkAppId 请您参考 步骤一
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig, 注意
userSig 与 userId 为一一对应关系
    userSig: '',
    // 用户在您业务中使用的昵称
    userName: '',
    // 用户在您业务中使用的头像链接
    avatarUrl: '',
    // 用户在您业务中需要的皮肤主题颜色及是否支持切换皮肤主题
    theme: {
      defaultTheme:'black',
      isSupportSwitchTheme: true
     }
  })
  // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
  await handleCreateRoom();
})

// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认
为'FreeToSpeak'，注意目前仅支持自由发言模式
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
  const roomMode = 'FreeToSpeak';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.createRoom({ roomId, roomName: roomId, roomMode, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.createRoom error: ' + error.message);
  }
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
```

```
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.enterRoom({ roomId, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.enterRoom error: ' + error.message);
  }
}
</script>

<style lang="scss">
html, body {
    width: 100%;
    height: 100%;
    margin: 0;
}

#app {
  width: 100%;
  height: 100%;
}
</style>
```

Vue2 项目引入 TUIRoom 组件

- 复制 `TUIRoomKit/Web/vue2/src/TUIRoom` 文件夹到已有项目 src/ 目录下，引用成功后目录结果如图所示：



- 在页面中引用 TUIRoom 组件。例如：在 `App.vue` 组件中引入 TUIRoom 组件。
  - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
  - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

> ⚠ **注意：**

> 在页面中复制以下代码之后，需要修改 TUIRoom 接口的参数为实际数据。

```
<template>
  <div id="app">
    <room-container ref="TUIRoomRef"></room-container>
  </div>
</template>

<script>
import RoomContainer from '@/TUIRoom/index.vue';
export default {
  name: 'App',
  components: { RoomContainer },
  data() {
    return {};
  },
  async mounted() {
    // 初始化 TUIRoom 组件
    // 主持人在创建房间前需要先初始化 TUIRoom 组件
    // 普通成员在进入房间前需要先初始化 TUIRoom 组件
    await this.$refs.TUIRoomRef.init({
      // 获取 sdkAppId 请您参考 步骤一
      sdkAppId: 0,
      // 用户在您业务中的唯一标示 Id
      userId: '',
      // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig, 注意
userSig 与 userId 为一一对应关系
      userSig: '',
      // 用户在您业务中使用的昵称
      userName: '',
      // 用户在您业务中使用的头像链接
      avatarUrl: '',
    });
    // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
    await this.handleCreateRoom();
  },
  methods: {
    // 主持人创建房间，该方法只在创建房间时调用
    async handleCreateRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 string
      // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认
为'FreeToSpeak'，注意目前仅支持自由发言模式
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
      const roomId = '123456';
      const roomMode = 'FreeToSpeak';
      const roomParam = {
          isOpenCamera: true,
          isOpenMicrophone: true,
      }
      try {
        await this.$refs.TUIRoomRef.createRoom({ roomId, roomName: roomId, roomMode, roomParam });
      } catch (error) {
        alert('TUIRoomKit.createRoom error: ' + error.message);
      }
```

```
    },
    // 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
    async handleEnterRoom() {
      // roomId 为用户进入的房间号，要求 roomId 类型为 string
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id）
      const roomId = '123456';
      const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
      }
      try {
        await this.$refs.TUIRoomRef.enterRoom({ roomId, roomParam });
      } catch (error) {
        alert('TUIRoomKit.enterRoom error: ' + error.message);
      }
    }
  },
};

</script>

<style lang="scss">
html, body {
  width: 100%;
  height: 100%;
  margin: 0;
}

#app {
  width: 100%;
  height: 100%;
}
</style>
```

步骤四：配置开发环境

TUIRoom 组件引入之后，为了确保项目可以正常运行，需要进行以下配置：

配置 Vue3 + Vite + TS 开发环境

1. **安装依赖**
   - 安装开发环境依赖：

     ```
     npm install sass typescript -S -D
     ```

   - 安装生产环境依赖：

     ```
     npm install mitt pinia @tencentcloud/tuiroom-engine-js vue-i18n -S
     ```

2. **注册 Pinia**

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 src/main.ts 文件。

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册 pinia
app.use(createPinia());
app.mount('#app');
```

3. **配置中英文切换**

TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
// src/main.ts
// 引入 locales/index.ts 文件，请确认引入路径是否正确
import VueI18n from './TUIRoom/locales/index';

app.use(VueI18n);
```

4. 配置 esLint 校验

如果您不希望 TUIRoomKit 组件的 esLint 规则和您本地的规则出现冲突，您可以在 .eslintignore 中添加忽略 TUIRoom 文件夹。

```
src/TUIRoom
```

配置 Vue3 + Webpack + TS 开发环境

1. **安装依赖**
   - 安装开发环境依赖：

```
npm install sass sass-loader typescript -S -D
```

   - 安装生产环境依赖：

```
npm install mitt pinia @tencentcloud/tuiroom-engine-js vue-i18n rtc-detect -S
```

2. **注册 Pinia**

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 src/main.ts 文件。

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册 pinia
app.use(createPinia());
app.mount('#app');
```

3. **配置中英文切换**

TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
// src/main.ts
// 引入 locales/index.ts 文件，请确认引入路径是否正确
import VueI18n from './TUIRoom/locales/index';

app.use(VueI18n);
```

4.**配置 esLint 校验**

如果您不希望 TUIRoomKit 组件的 esLint 规则和您本地的规则出现冲突，您可以在 `.eslintignore` 中配置忽略 TUIRoom 相关文件。

```
src/TUIRoom
src/App.vue
src/main.ts
```

配置 Vue2 + Webpack + TS 开发环境

> **说明：**
> TUIRoomKit 组件要求 vue2 项目工程安装 vue2.7 版本并支持 typescript 环境。如您当前项目为 vue2.6 + js 环境，按照以下步骤执行，可升级为 vue2.7 + ts 环境。
> vue2.6 升级到 vue2.7 是平滑升级的，对已有代码无影响。配置 ts 环境之后，对已有 js 代码无影响。请放心升级。

1. **配置 typescript, 支持 TUIRoom 组件加载**

```
vue add typescript
```

配置 TS 开发环境的选项可参考图片：

```
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected polyfills, transpiling JSX)? Yes
? Convert all .js files to .ts? No
? Allow .js files to be compiled? Yes
? Skip type checking of all declaration files (recommended for apps)? No
```

2. **安装依赖**

- 安装开发环境依赖：

```
npm install sass sass-loader -S -D
```

- 安装生产环境依赖：

```
npm install vue@^2.7 pinia @tencentcloud/tuiroom-engine-js vue-i18n@^8 vue-i18n-bridge rtc-detect mitt -S
```

3. **注册 Pinia**

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 `src/main.ts` 或者

`src/main.js` 文件。

```
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
}).$mount('#app');
```

4. **配置中英文切换**

   TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
import i18n from './TUIRoom/locales/';

Vue.use(i18n);
```

5. 配置 esLint 校验

   如果您不希望 TUIRoomKit 组件的 esLint 规则和您本地的规则出现冲突，您可以在 `.eslintignore` 中配置忽略 TUIRoom 相关文件。

```
src/TUIRoom
```

步骤五：开发环境运行

在控制台执行开发环境运行脚本，使用浏览器打开包含 TUIRoomKit 的页面，即可在页面中使用 TUIRoomKit 组件。

Vue3 + Vite + TS 项目

1. 执行开发环境命令。

```
npm run dev
```

2. 在浏览器中打开页面 `http://localhost:3000/` 。

> ⚠ **注意：**
> 因 TUIRoom 按需引入 element-plus 组件，会导致开发环境路由页面第一次加载时反应较慢，等待 element-plus 按需加载完成即可正常使用。element-plus 按需加载不会影响打包之后的页面加载。

Vue3 + Webpack + TS 项目

1. 执行开发环境命令

```
npm run serve
```

2. 在浏览器中打开页面 `http://localhost:8080/`

> ⚠ **注意：**
> 运行过程中若 src/TUIRoom 目录中有 eslint 报错，可在 .eslintignore 文件中添加 /src/TUIRoom/ 路径屏蔽 eslint 检查。

3. 体验 TUIRoom 组件功能

Vue2 + Webapck + TS 项目

1. 执行开发环境命令

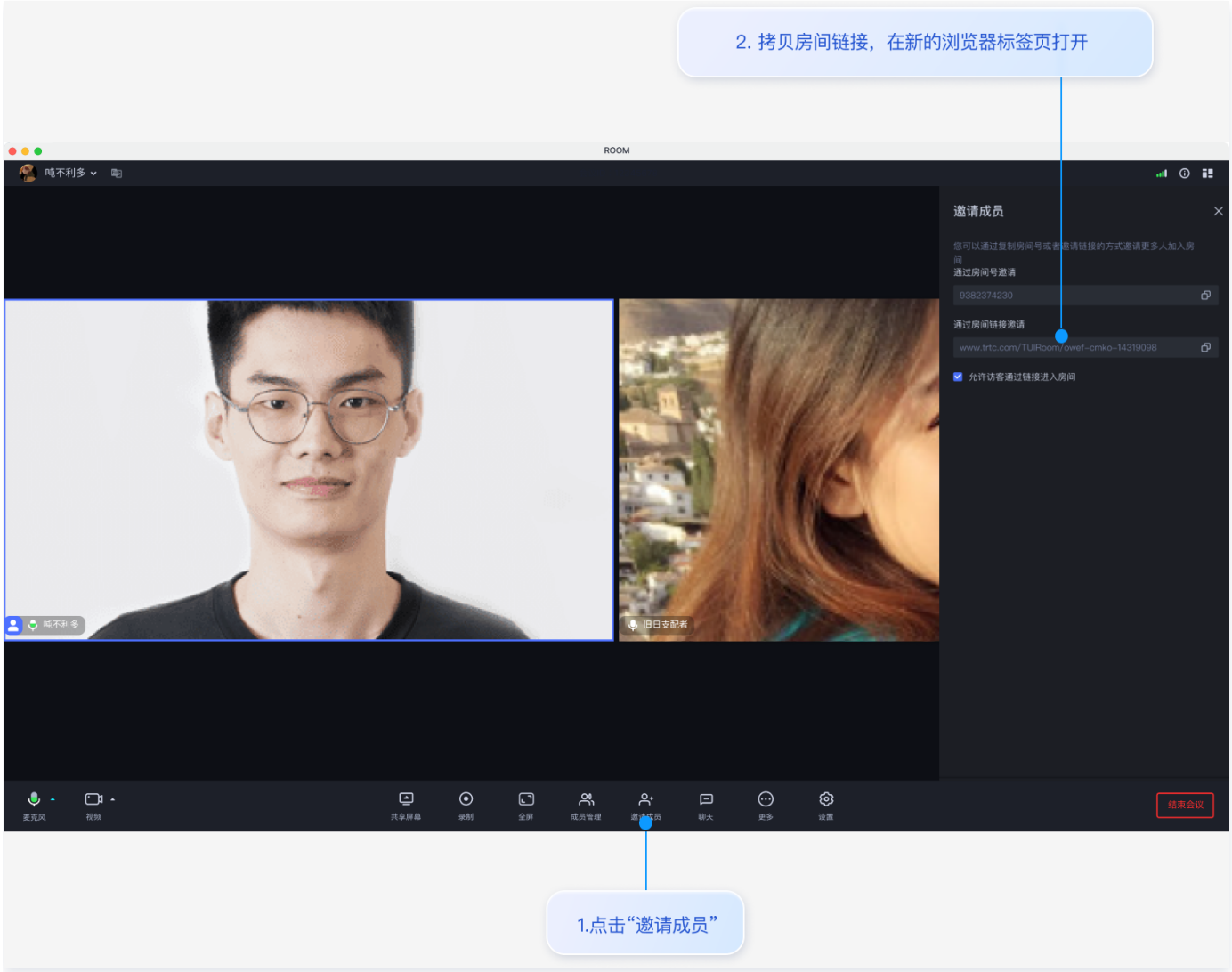
```
npm run serve
```

2. 在浏览器中打开页面 `http://localhost:8080/`

> ⚠ **注意：**
> 运行过程中若 src/TUIRoom 目录中有 eslint 报错，可在 .eslintignore 文件中添加 /src/TUIRoom/ 路径屏蔽 eslint 检查。

步骤六：开始您的第一场会议

1.项目启动后，点击下方**邀请成员**。
2.拷贝房间链接，在新的浏览器标签页打开链接进入房间。
Web 端 “开始您的第一场会议” 具体步骤示例：



H5 端 “开始您的第一场会议” 具体步骤示例：



步骤七：生产环境部署

1. 打包 dist 文件

```
npm run build
```

> **说明：**
> 实际打包命令请查看 package.json 文件

2. 部署 dist 文件到服务器上

> **生产环境要求使用 HTTPS 域名：**



页面访问协议说明

浏览器厂商出于对用户安全、隐私等问题的考虑，限制网页在 https 协议下才能正常使用 TRTC Web SDK（WebRTC）的全部功能。为确保生产环境用户顺畅接入和体验 TRTC Web SDK 的全部功能，请使用 https 协议访问音视频应用页面。

注：本地开发可以通过 http://localhost 或者 file:// 协议进行访问。

URL域名及协议支持情况请参考如下表格：

应用场景	协议	接收（拉流）	发送（推流）	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	

常见问题

常见问题

其他文档

- TUIRoomKit
- TUIRoom demo 快速跑通
- 界面定制 (TUIRoomKit)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和问题反馈。

Android

最近更新时间：2024-01-11 15:42:51

本文将介绍如何用最短的时间完成 TUIRoomKit 组件的接入，跟随本文档，您将在一个小时的时间内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的音视频会议功能。

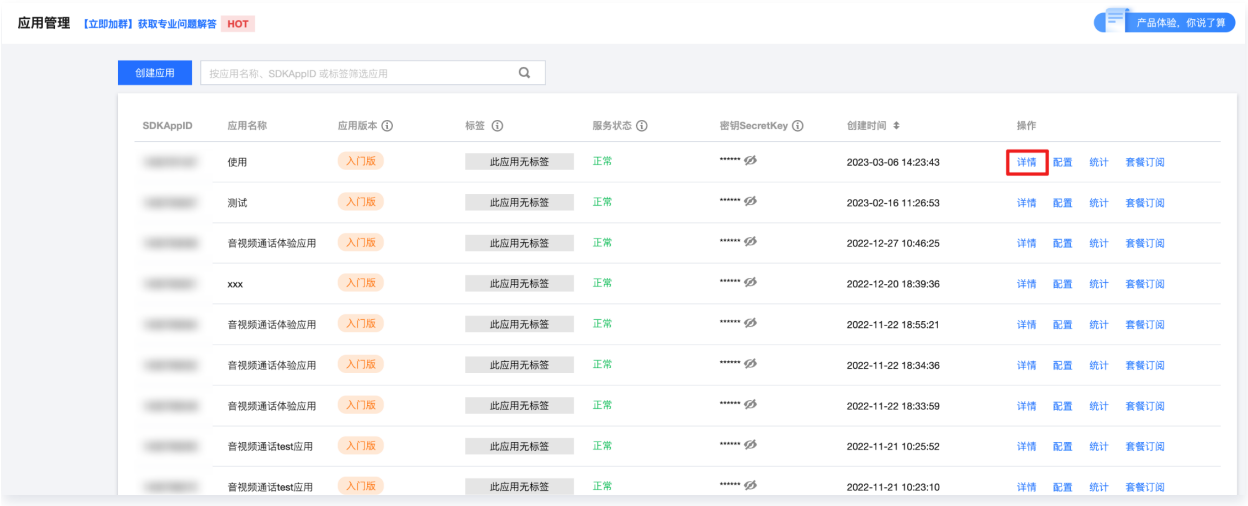
环境准备

- 最低兼容 Android 4.4（SDK API Level 19），建议使用 Android 5.0（SDK API Level 21）及以上版本。
- Android Studio 3.5 及以上的版本（Gradle 3.5.4 及以上的版本）。
- Android 4.4 及以上的手机设备。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），点击**详情**，进入应用概览界面。



2. 在应用概览页面找到**含 UI 低代码集成接入**卡片，选择**多人音视频（TUIRoomKit）**，点击**领取体验**，领取7天体验版 TUIRoomKit 进行接入测试。

> **注意**
>
> - 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
> - 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
> - 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请点击**场景套餐订阅**按钮或前往 购买页 购买；

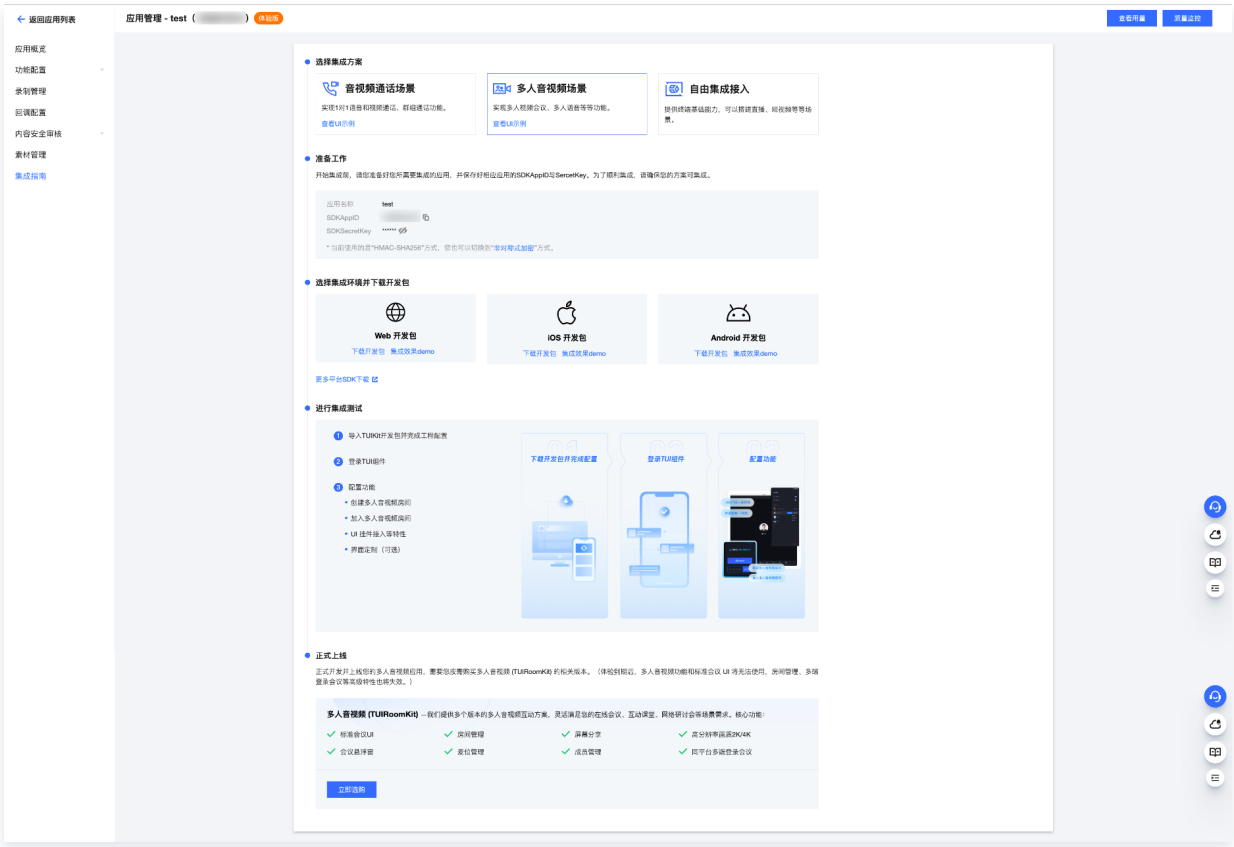




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



4. 单击**集成指南**，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：集成 TUIRoomKit 组件

1. 在 Github 中克隆/下载代码，然后拷贝 Android 目录下的 debug 、 timcommon 、 tuichat 、 tuiroomkit 子目录到您当前工程中的 app 同级目录中。
2. 工程根目录下找到 setting.gradle 文件，并在其中增加如下代码，它的作用是将 tuiroomkit 作为本地模块导入到您当前的项目中。

```
include ':debug'
```

```
include ':timcommon'
include ':tuichat'
include ':tuiroomkit'
```

3. 在 app 目录下找到 build.gradle 文件，并在其中增加如下代码，它的作用是声明当前 app 对新加入的 tuiroomkit 组件的依赖。

```
api project(':tuiroomkit')
```

说明

由于我们在 SDK 内部使用了Java 的反射特性，需要将 SDK 中的部分类加入不混淆名单，因此需要您在 proguard-rules.pro 文件中添加如下代码：

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

注意

TUIRoomKit 会在内部帮助您动态申请相机、麦克风、读取存储权限等，如果因为您的业务问题需要删减，可以请修改 tuiroomkit/src/main/AndroidManifest.xml 。

步骤三：登录 TUIRoomKit 组件

在您的项目中添加如下代码，它的作用是通过调用 TUILogin 中的相关接口完成组件的登录。这个步骤异常关键，因为只有在登录后才能正常使用 TUIRoomKit 的各项功能，故请您耐心检查相关参数是否配置正确：

```
TUILogin.login(context,
        1400000001,  // 请替换为步骤一取到的 SDKAppID
        "998",       // 请替换为您的 UserID
        "xxxxxxxxxx", // 您可以在控制台中计算一个 UserSig 并填在这个位置
        callback);   // 登录是否成功的回调
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 步骤一 中的最后一步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 步骤一 的第3步中获取的 SecretKey 对 SDKAppID 、 UserID 等信息进行加密，就可以得到 UserSig ，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig 。
- 更多信息请参见 如何计算及使用 UserSig 。

注意

- **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
  - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
  - UserSig 被错配成了加密密钥（SecretKey），UserSig 是用 SecretKey 把 SDKAppID、UserID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 UserSig。
  - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。

- Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 app 在每次使用 TUIRoomKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤四：使用 TUIRoomKit

设置用户信息

通过调用 `TUIRoomKit` 的 `setSelfInfo` 即可设置用户的用户名、头像。

```
public abstract void setSelfInfo(String userName, String avatarURL, TUIRoomDefine.ActionCallback callback);
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像 URL

创建房间

通过调用 TUIRoomKit 的 createRoom 方法就可以创建房间。

```
TUIRoomDefine.RoomInfo roomInfo = new TUIRoomDefine.RoomInfo();
roomInfo.roomId = "123456";   // 请设置为您需要的房间号
TUIRoomKit.createInstance().createRoom(roomInfo, new TUIRoomDefine.ActionCallback() {
    @Override
    public void onSuccess() {
        Log.d(TAG, "create room success");
    }

    @Override
    public void onError(TUICommonDefine.Error error, String s) {
        Log.d(TAG, "create room failed");
    }
});
```

加入房间

通过调用 TUIRoomKit 的 enterRoom 方法并指定房间类型与房间号，就可以进入指定房间。

```
TUIRoomKit.createInstance().enterRoom("123456", true, true, true, new
TUIRoomDefine.GetRoomInfoCallback() {
    @Override
    public void onSuccess(TUIRoomDefine.RoomInfo roomInfo) {
        Log.d(TAG, "enter room success");
    }

    @Override
    public void onError(TUICommonDefine.Error error, String s) {
        Log.d(TAG, "enter room failed");
    }
```

```
});
```

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和问题反馈。

iOS

最近更新时间：2024-03-06 14:51:31

本文将介绍如何用最短的时间完成 TUIRoomKit 组件的接入，跟随本文档，您将在一个小时的时间内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的音视频会议功能。

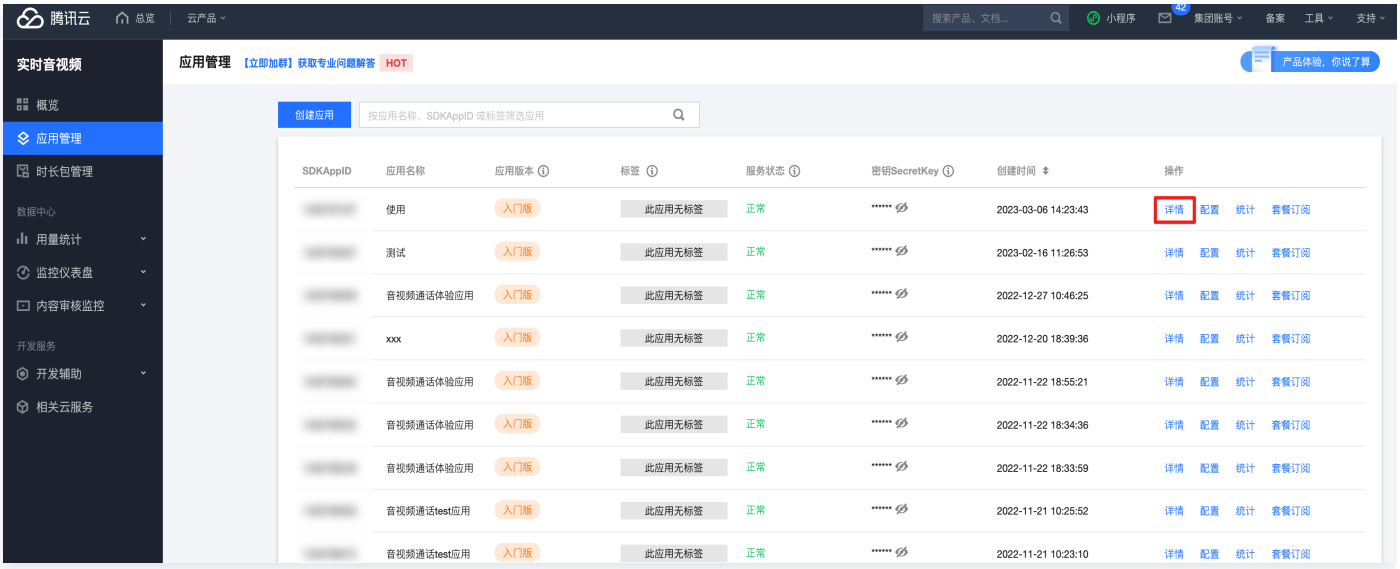
环境准备

iOS 13.0及更高。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），点击详情按钮，进入应用概览界面。



2. 在应用概览页面找到 **含 UI 低代码集成接入** 卡片，选择 **多人音视频（TUIRoomKit）**，点击领取体验按钮，领取7天体验版 TUIRoomKit 进行接入测试。

> **注意：**
> - 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
> - 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
> - 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请点击**场景套餐订阅**按钮或前往 购买页 购买；

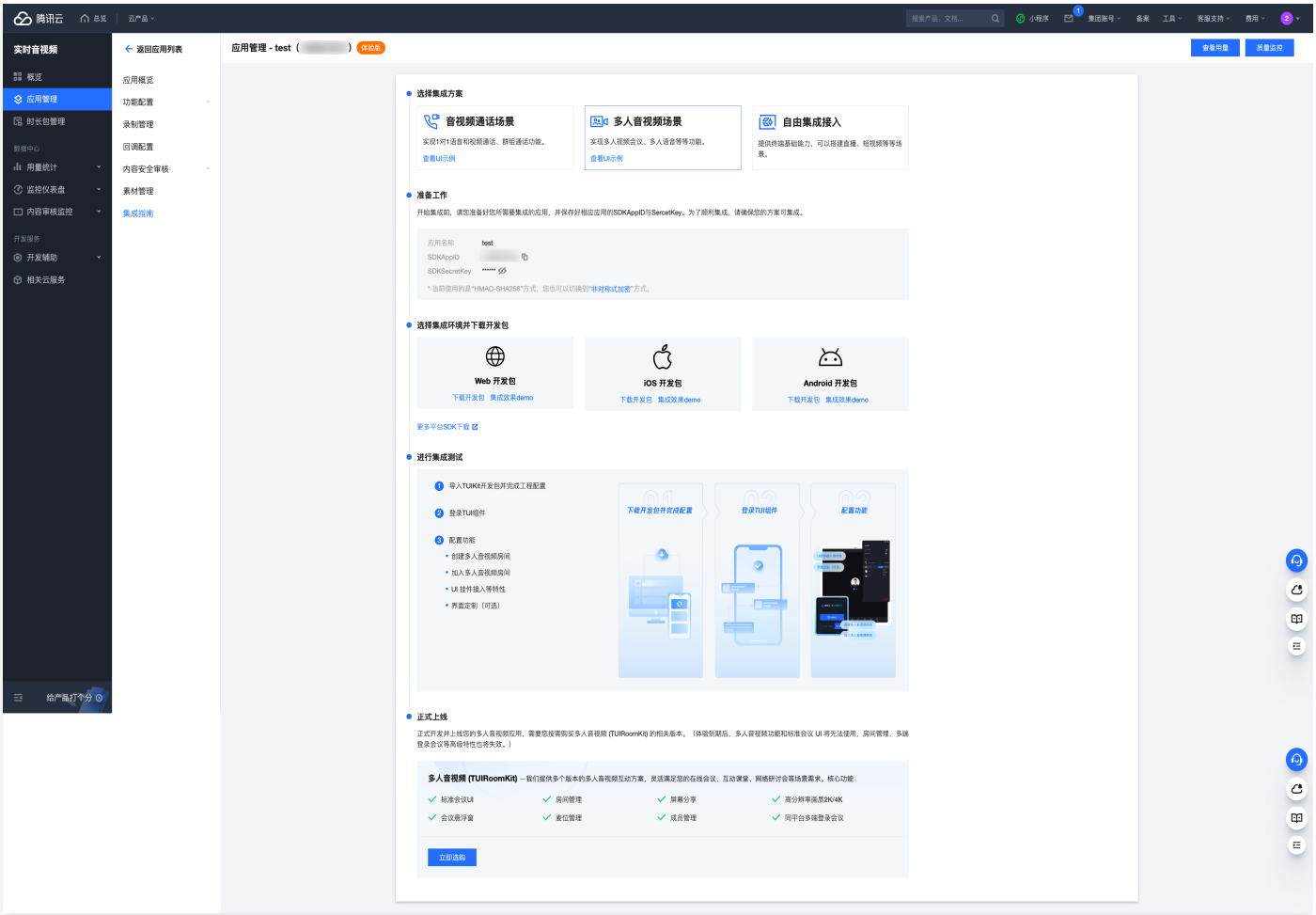




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



4. 单击**集成指南**，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：集成 TUIRoomKit 组件

使用 CocoaPods 导入组件，具体步骤如下：

1. 在您的 `Podfile` 文件中添加以下依赖。

```
pod 'TUIRoomKit'
```

2.执行以下命令，安装组件。

```
pod install
```

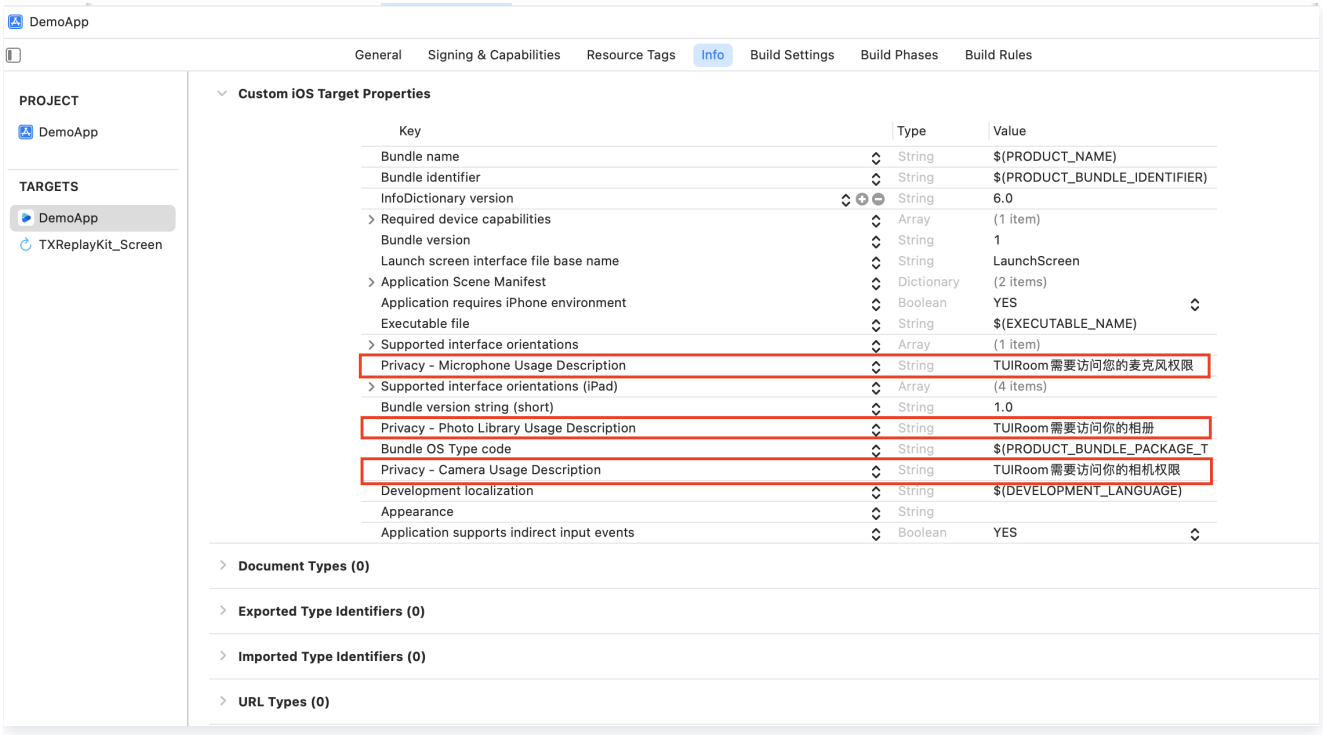
如果无法安装 TUIRoomKit 指定版本，执行以下命令更新本地的 CocoaPods 仓库列表。

```
pod repo update
```

步骤三：完成工程配置

使用音视频功能，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。

```
<key>NSCameraUsageDescription</key>
<string>TUIRoom需要访问您的相机权限</string>
<key>NSMicrophoneUsageDescription</key>
<string>TUIRoom需要访问您的麦克风权限</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>TUIRoom需要访问您的相册</string>
```



步骤四：登录TUI 组件

在您的项目中添加如下代码，它的作用是通过调用TUICore 中的相关接口完成TUI组件的初始化。这个步骤非常关键，因为只有在登录成功后才能正常使用 TUIRoomKit 的各项功能，故请您耐心检查相关参数是否配置正确：

```
import TUICore
// 组件登录
```

```
TUILogin.login(1400000001,                    // 请替换为步骤一取到的 SDKAppID
          userID: "xxxxxxxxxx",               // 请替换为您的 UserID
          userSig: "998") {                   // 您可以在控制台中计算一个 UserSig 并填在这个位置
    print("login success")
} fail: { (code, message) in
    print("login failed, code: \(code), error: \(message ?? "nil")")
}
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 步骤一 中的应用概览页面中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 步骤一 获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig。
- 更多信息请参见 如何计算及使用 UserSig 。

> ⚠ **注意：**
>
> - **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
>   - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
>   - UserSig 被错配成了加密密钥（SecretKey），UserSig 是用 SecretKey 把 SDKAppID、UserID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 UserSig。
>   - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。
> - Github 中的示例代码使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 app 在每次使用 TUIRoomKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤五：使用 TUIRoomKit

设置用户信息（可选）

通过调用 TUIRoomKit 的 setSelfInfo 即可设置用户的用户名、头像。

```
import TUIRoomKit

TUIRoomKit.createInstance().setSelfInfo(userName: "xx", avatarURL: "xx") {
    print("setSelfInfo success")
} onError: { code, message in
    print("setSelfInfo error, code:\(code), message:\(message)")
}
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像 URL

创建房间

通过调用 TUIRoomKit 的 createRoom 函数就可以创建房间。

```
import TUIRoomEngine
import TUIRoomKit

let roomInfo = TUIRoomInfo()
roomInfo.roomId = "998"                    // 请设置为您需要的房间号
TUIRoomKit.createInstance().createRoom(roomInfo: roomInfo) {
    print("createRoom success")
} onError: { code, message in
    print("createRoom error, code:\(code), message:\(message)")
}
```

> **说明：**
> 只调用创建房间接口不会自动进入房间，还需要调用 进入房间 接口。

进入房间

通过调用 TUIRoomKit 的 enterRoom 函数并指定房间号，就可以进入指定房间。

```
import TUIRoomKit

TUIRoomKit.createInstance().enterRoom(roomId: "998",        // 请设置为您要进入的房间号
                                      enableAudio: true,        // 请设置进房是否开启音频
                                      enableVideo: true,        // 请设置进房是否开启视频
                                isSoundOnSpeaker: true) {       // 请设置进房是否开启扬声器
    print("enterRoom success")
} onError: { code, message in
    print("enterRoom error, code:\(code), message:\(message)")
}
```

步骤六：更多特性

UI 挂件接入

使用 CocoaPods 导入聊天挂件，具体步骤如下：

1. 在您的 `Podfile` 文件中添加以下依赖。

```
pod 'TUIChat'        # [可选] 聊天挂件
```

2.执行以下命令，安装组件。

```
pod install
```

常见问题

如果您选择同时集成 RoomKit 与 CallKit， 可以参考以下集成方式，后续我们会尽快优化多组件之间的集成方式。

```
# CallKit
pod 'TUICallKit', '~> 1.9.0.680' # 同时集成RoomKit与CallKit，推荐TUICallKit 1.9.0.680及以上版本

# RoomKit
pod 'TUIRoomKit', '~> 1.5.1'  # 同时集成RoomKit与CallKit，推荐TUIRoomKit 1.5.1 及以上版本


# Kingfisher(swift 网络图片缓存库) 版本号设置参考：https://github.com/onevcat/Kingfisher
# iOS 10+ Xcode 12 推荐选择 pod 'Kingfisher', '~> 6.3.1'
# iOS 11+ Xcode 13 推荐选择 pod 'Kingfisher', 'version6-xcode13'
# iOS 12+ Xcode 13 推荐选择 pod 'Kingfisher', '~> 7.0'
pod 'Kingfisher', '~> 6.3.1'
```

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和问题反馈。

uni-app（小程序）

最近更新时间：2024-01-26 16:35:21

TUIRoomKit 是腾讯云推出的多人音视频 UI 组件。组件提供房间管理、音视频控制、屏幕共享、成员管理、麦位管理、即时聊天、自定义布局切换等丰富的功能交互。同时支持中英文切换，一键换肤等能力。



本文介绍 TUIRoomKit (Uniapp) 的接入指引，助力您快速上线企业会议、在线教育、医疗问诊、在线巡视、远程定损等业务场景。
您可以单击 Github/Gitee 下载 TUIRoomKit 代码，并参考代码仓库 README.md 文档跑通 TUIRoomKit Uniapp 示例工程。

小程序 Demo

- 如果您想要直接跑通一个新工程，请直接阅读 uni-app demo 快速跑通 。
- 如果您想要**亲自集成** TUIRoomKit 组件，搭建一个多人音视频房间小程序，请跟随本文档。

开发环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.13.0。

> **警告：**
> - 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，请使用**企业小程序账号**申请相关权限进行开发。
> - 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在**真机**上进行运行体验。

步骤一：开发前准备

1. 开通小程序权限

由于 TUIRoomKit 所使用的小程序标签有更苛刻的权限要求，因此集成 TUIRoomKit 的第一步就是要开通小程序的类目和标签使用权限。**否则无法使用**，这包括如下步骤：

- 小程序推拉流标签不支持个人小程序，只支持企业类小程序。需要在 注册 时填写主体类型为企业，如下图所示：



- **小程序推拉流标签使用权限暂时只开放给有限 类目 。**
- **符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：**



2. 在小程序控制台配置域名

在 微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名中设置 request 合法域名 和 socket 合法域名，如下图所示：

- request 合法域名：

```
https://official.opensso.tencent-cloud.com
https://yun.tim.qq.com
https://cloud.tencent.com
https://webim.tim.qq.com
https://query.tencent-cloud.com
https://web.sdk.qcloud.com
```

- socket 合法域名：

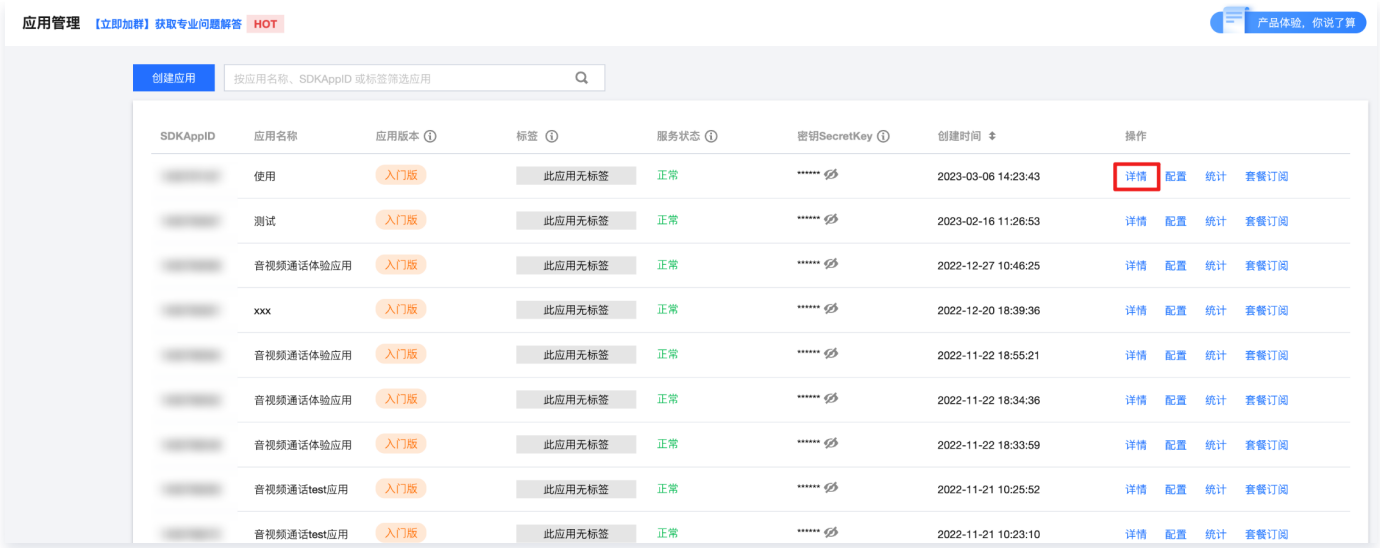
```
wss://wss.im.qcloud.com
wss://wss.tim.qq.com
```



3. 开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），单击**详情**，进入应用概览界面。



2. 在应用概览页面找到**含 UI 低代码集成接入 卡片，选择多人音视频（TUIRoomKit）**，单击**领取体验**，领取7天体验版 TUIRoomKit 进行接入测试。

> **注意：**
> - 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
> - 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
> - 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请单击**场景套餐订阅**或前往 购买页 购买；

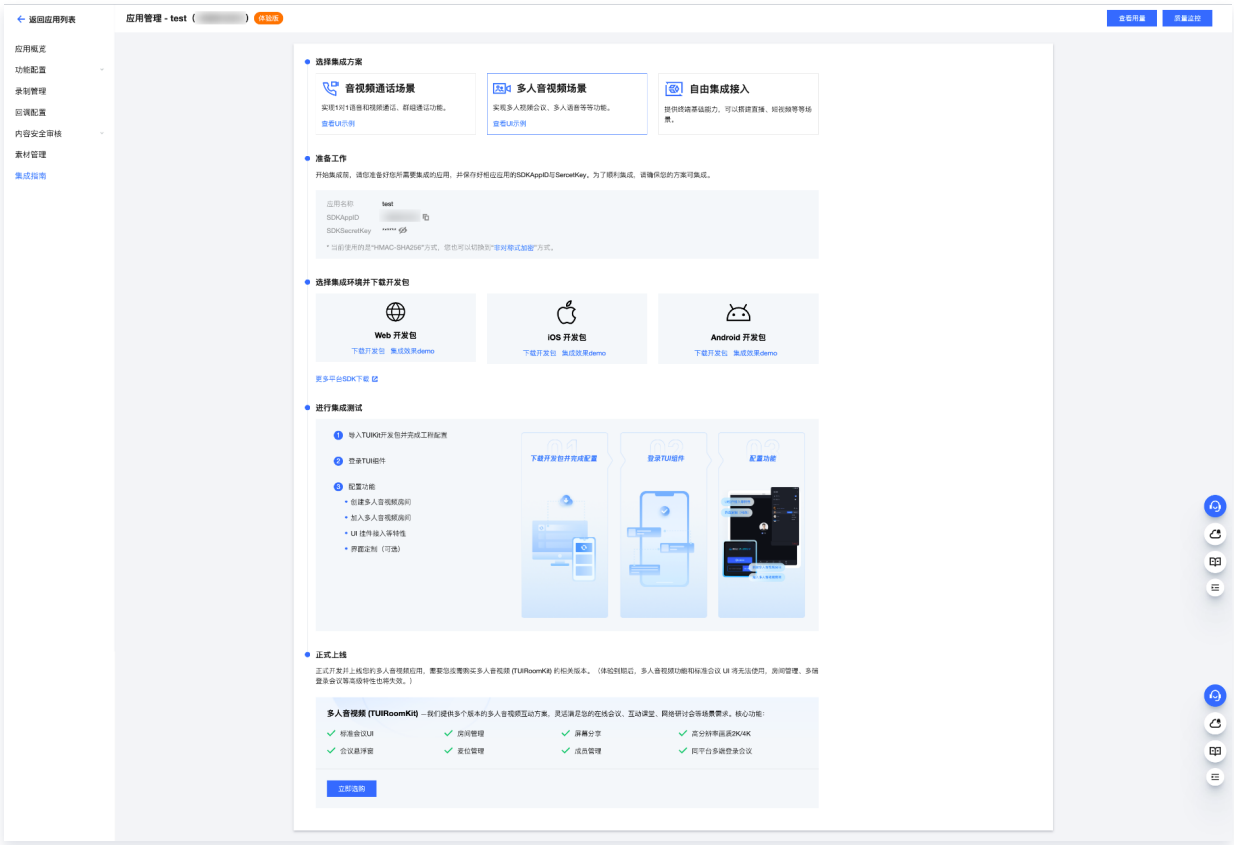




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



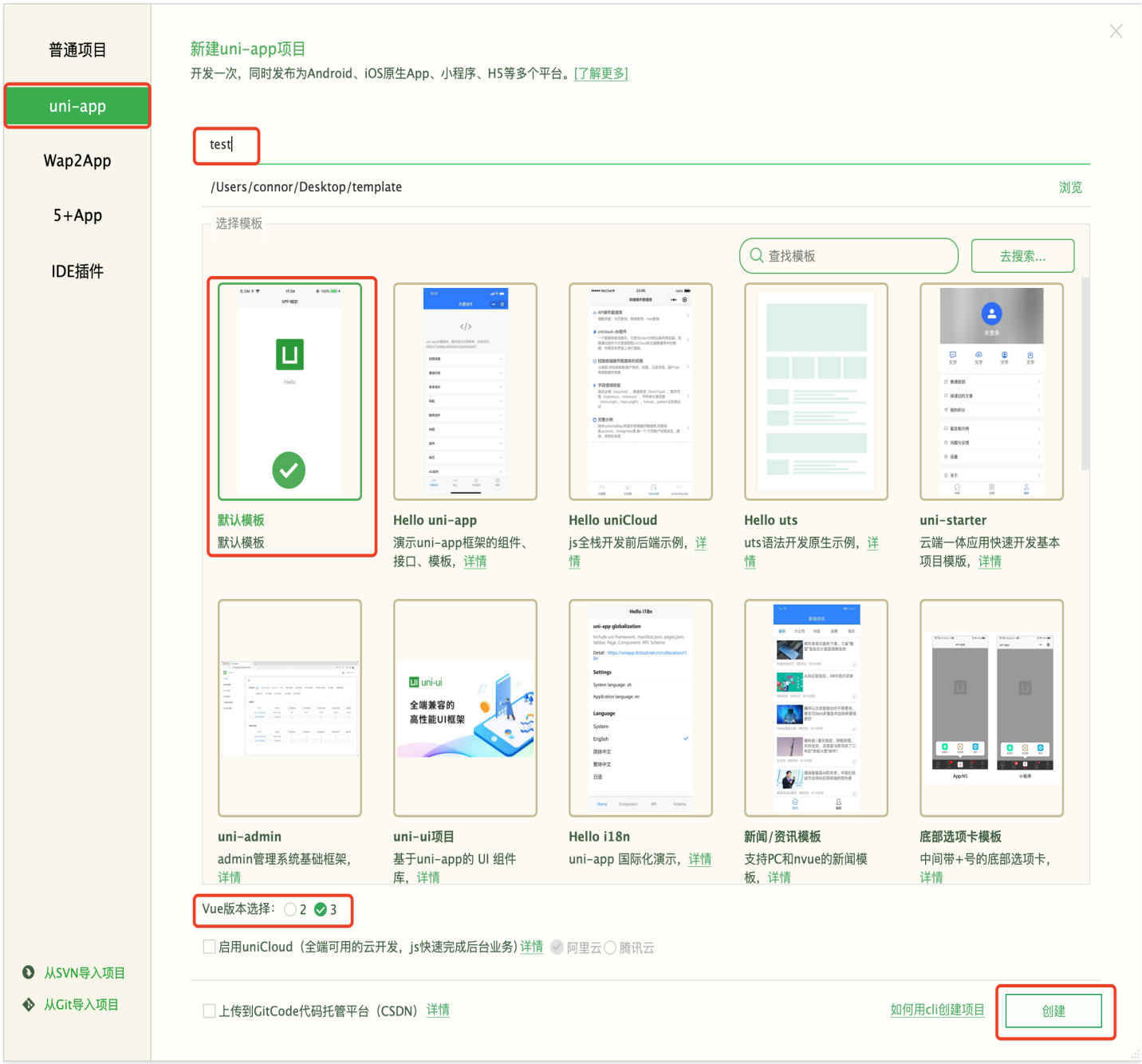
4. 单击**集成指南**，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：准备 Uniapp 工程代码

> **注意：**
> **目前只支持 ts + vite + vue3 工程**
> 如果您已有自己的 uni-app 工程，可跳过该步骤。如果没有，可选择以下方式生成模板工程。

在 HBuilder 中创建小程序项目。



步骤三：下载并导入 TUIRoom 组件

> **注意：**
> 因为小程序对主包大小的限制，这里 TUIRoom 默认使用分包方案，通过脚本自动配置。

3.1 npm 依赖安装

此处需在 uni-app 项目根目录下运行

```
# 在终端输入npm init -y，创建package.json文件。
npm init -y
# 安装项目依赖
```

```
npm i @tencentcloud/tui-roomkit-uniapp
```

3.2 执行项目自动化脚本

```
# 此脚本会自动拷贝相关依赖到您的项目。并配置 vite.config.ts、tsconfig.json、mainfest.json、pages.json 文件
# 同时创建一个空白页面 roomkitTest.vue 用于展示 roomKit 项目
node ./node_modules/@tencentcloud/tui-roomkit-uniapp/configureProject.js
```

> **警告：**
> 此处如果没有特殊需求建议您选择 **自动配置**，如执行脚本的过程中您未选择自动配置，请结合 项目配置说明 手动进行修改。

3.3 手动修改项目配置

1. 修改 `./src/main.js（main.ts）` ，引入 `VueI18n` 以及 `pinia` 。（**注意此处为增量添加，根据自己项目选择添加位置**）

> **注意：**
> 如果项目使用 HbuilderX 创建，main.js 文件会有条件编译相关代码。需要您检查是否将配置正确填写在了 **#ifdef VUE3** 条件中。

HbuilderX

```
import App from './App'
import { createPinia } from "pinia";
import VueI18n from "./locales/index";

// #ifndef VUE3
import Vue from 'vue'
import './uni.promisify.adaptor'
Vue.config.productionTip = false
App.mpType = 'app'
const app = new Vue({
  ...App
})
app.$mount()
app.use(createPinia());
app.use(VueI18n);
// #endif

// #ifdef VUE3
import { createSSRApp } from 'vue'
export function createApp() {
  const app = createSSRApp(App)
  app.use(createPinia());
  app.use(VueI18n);
  return {
    app
  }
}
// #endif
```

CLI

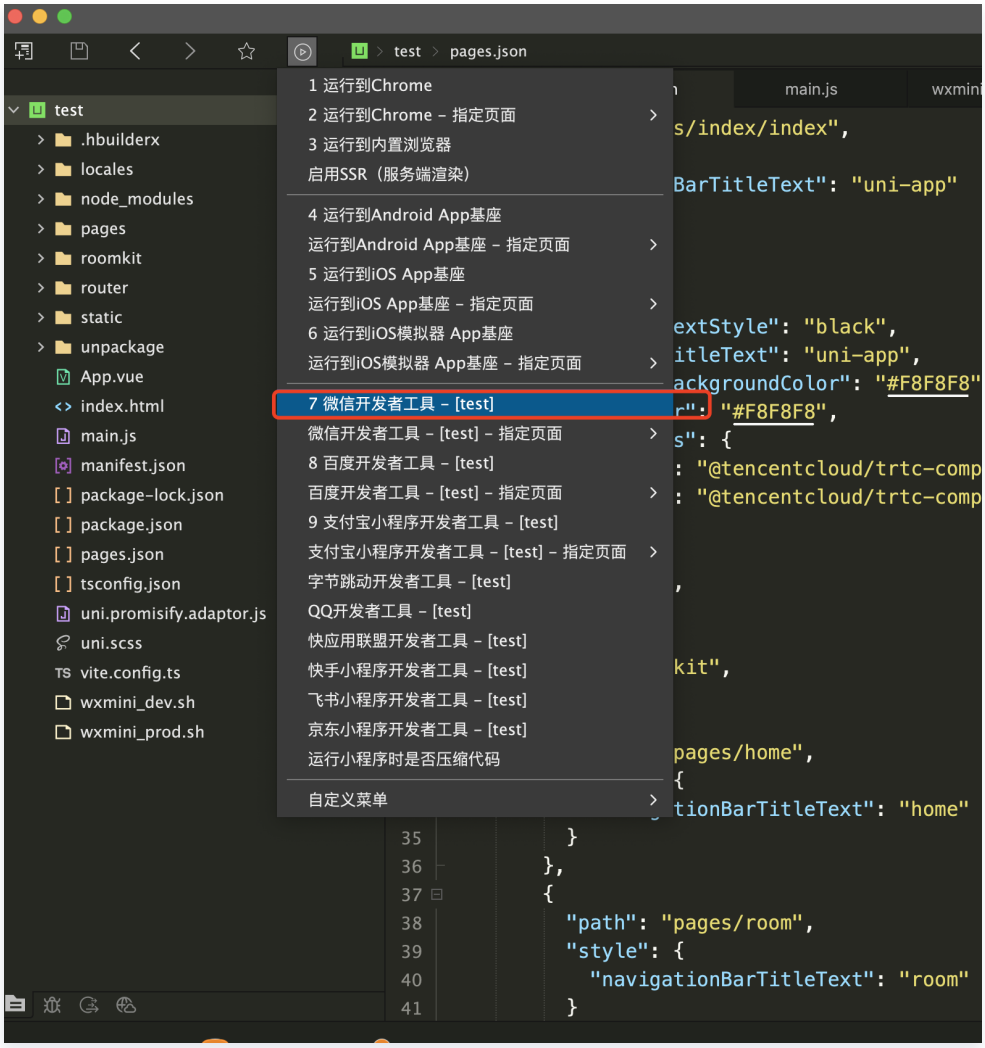
```
import { createPinia } from 'pinia';
// locales 为 TUIRoom/locales，因为主包无法使用分包资源，此处拷贝了一份 locales 在 src 路径下
import VueI18n from './locales/index';

app.use(createPinia());
app.use(VueI18n);
```

2. 找到并打开 ./roomkit/config/basic-info-config.js 文件，并修改 basic-info-config.js 文件中的相关参数：

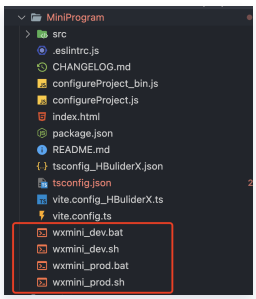
```
/**
 * Tencent Cloud SDKAppId, which should be replaced with user's SDKAppId.
 * Enter Tencent Cloud TRTC [Console] (https://console.cloud.tencent.com/trtc ) to create an application,
 * and you will see the SDKAppId.
 * It is a unique identifier used by Tencent Cloud to identify users.
 *
 * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
 * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId,
 * 它是腾讯云用于区分客户的唯一标识。
 */

export const SDKAPPID = 0;      You, 13 minutes ago • Uncommitted changes

/**
 * Encryption key for calculating signature, which can be obtained in the following steps:
 *
 * Step1. Enter Tencent Cloud TRTC [Console](https://console.cloud.tencent.com/rav ),
 * and create an application if you don't have one.
 * Step2. Click your application to find "Quick Start".
 * Step3. Click "View Secret Key" to see the encryption key for calculating UserSig,
 * and copy it to the following variable.
 *
 * Notes: this method is only applicable for debugging Demo. Before official launch,
 * please migrate the UserSig calculation code and key to your backend server to avoid
 * unauthorized traffic use caused by the leakage of encryption key.
 * Document: https://intl.cloud.tencent.com/document/product/647/35166#Server
 *
 * 计算签名用的加密密钥，获取步骤如下：
 *
 * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一个，
 * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
 * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
 *
 * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
 * 文档：https://cloud.tencent.com/document/product/647/17275#Server
 */
export const SECRETKEY = '';
```

3.4 运行到微信开发者工具

1. 运行到微信开发者工具，等待编译完成并打开微信开发者工具。



- **打开微信开发者工具后，在项目根目录（非编译后 dist 文件目录）install 相关依赖。**
  若项目中脚本不存在，可手动下载脚本并将其拷贝在您的项目根目录： wxmini_uniapp_script 。



Windows

```
# npm run dev:mp-weixin 编译完成后执行该指令
# 此处使用的路径为默认路径，如果您对此路径进行了修改，需要手动对脚本路径进行调整
./wxmini_dev.bat
```

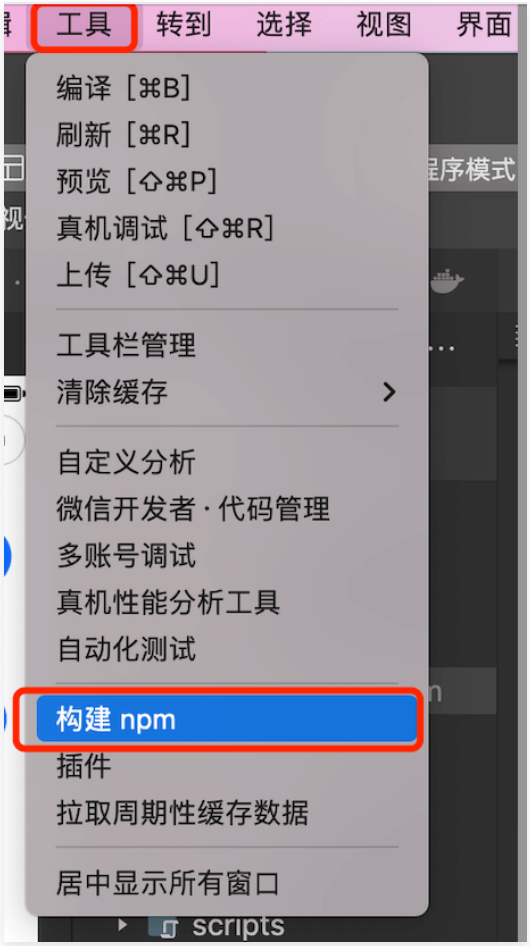
Mac

```
# npm run dev:mp-weixin 编译完成后执行该指令
```

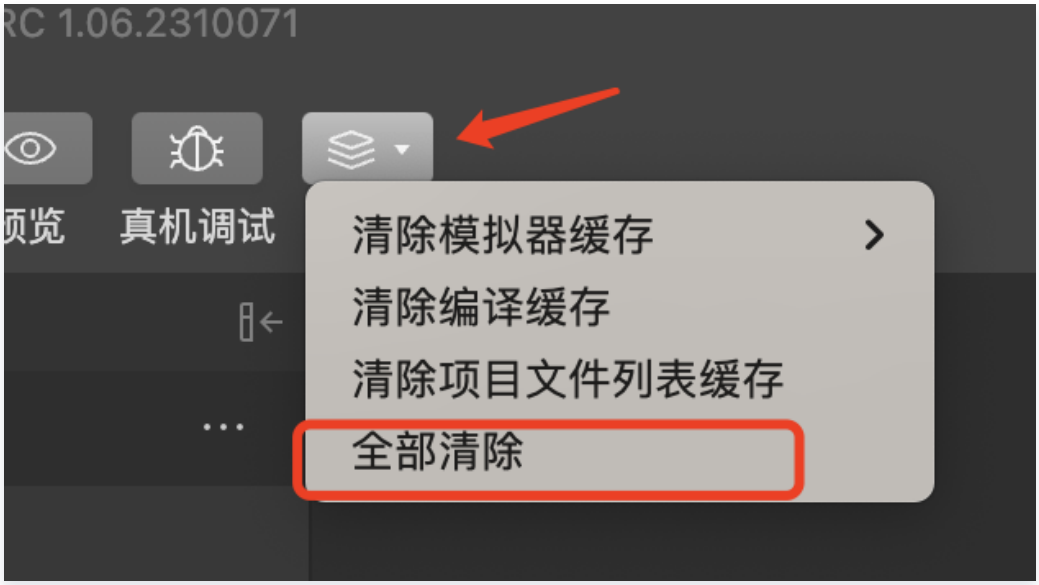
```
# 此处使用的路径为默认路径，如果您对此路径进行了修改，需要手动对脚本路径进行调整
bash wxmini_dev.sh
```

> **说明：**
> 因为小程序对主包大小有限制，这里默认采用了分包方案，部分依赖需要进入编译后的目录执行 npm install 进行下载，此处将操作指令集成在 wxmini_dev.sh/wxmini_dev.bat 脚本中，因此需要在编译完成后执行 bash wxmini_dev.sh/wxmini_dev.bat 安装依赖，您也可以根据指令内容手动安装。

- 构建 npm，微信开发者工具单击**工具 > 构建 npm**。具体如下图：



- 点击预览，扫描二维码，体验程序。
- 单击**清缓存 > 全部清除**，避免开发者工具的缓存造成渲染异常。

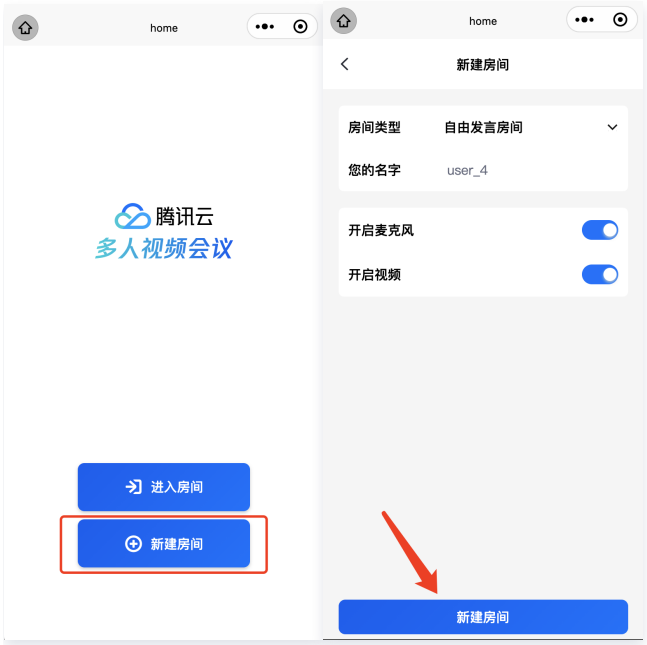


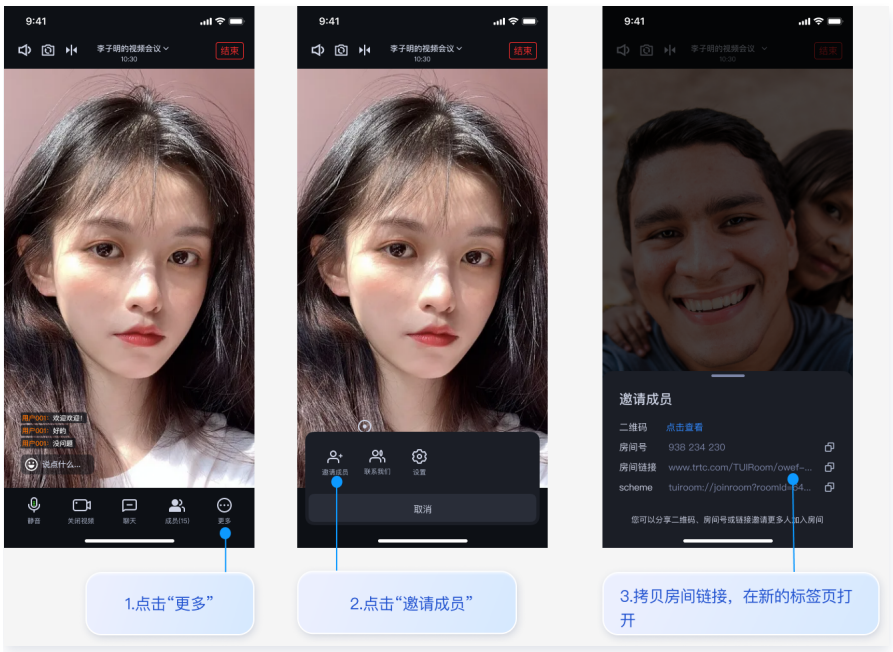
- 编译小程序。



步骤四：开始您的第一场会议

单击**新建房间**，开始您的第一场会议。





项目配置说明

> **警告：**
> 以下修改均为增量修改，需要根据自己项目配置进行添加。

1. 修改 `./src/manifest.json` 小程序相关配置，开启分包并设置 node_modules 目录，参考文档 自定义 node-modules、miniprogram-npm 位置构建 、uniapp 分包 。

```
/* 小程序特有相关 */
"mp-weixin": {
  "setting": {
    "urlCheck": false,
    "packNpmManually": true,
    "packNpmRelationList": [
```

```
      {
        "packageJsonPath": "./package.json",
        "miniprogramNpmDistDir": "./"
      },
      {
        "packageJsonPath": "./roomkit/package.json",
        "miniprogramNpmDistDir": "./roomkit"
      }
    ]
  },
  "usingComponents": true,
  "optimization": {
    "subPackages": true
  }
},
```

2. 修改 ./src/pages.json ，添加分包页面以及全局组件引入。

```
{
  "pages": [
    {
      "path": "pages/roomkitTest",
      "style": {
        "navigationBarTitleText": ""
      }
    },
  ],
  "subpackages": [
    {
      "root": "roomkit",
      "pages": [
        {
          "path": "pages/home",
          "style": {
            "navigationBarTitleText": "home"
          }
        },
        {
          "path": "pages/room",
          "style": {
            "navigationBarTitleText": "room"
          }
        }
      ]
    }
  ],
  "globalStyle": {
    "usingComponents": {
      "trtc-pusher": "@tencentcloud/trtc-component-wx/trtc-pusher",
      "trtc-player": "@tencentcloud/trtc-component-wx/trtc-player"
    }
  },
}
```

3. 修改 ./vite.config.ts ，将 @tencentcloud/tuiroom-engine-wx 设置为外部依赖，用于分包减少主包体积。

HBuliderX 创建

```
import { defineConfig } from 'vite';
import uni from '@dcloudio/vite-plugin-uni';


import path from 'path';


function resolve(dir) {
  return path.join(__dirname, dir);
}


// https://vitejs.dev/config/
export default defineConfig(({ mode }) => {
  const isProd = mode === 'production';


  const commonConfig = {
    optimizeDeps: {
      include: ['@tencentcloud/tuiroom-engine-wx'],
    },
    plugins: [
      uni(),
    ],
    resolve: {
      alias: {
        '@': resolve('./'),
        '@TUIRoom': resolve('/roomkit/TUIRoom'),
      },
    },
    build: {
      rollupOptions: {
        external: ['@tencentcloud/tuiroom-engine-wx'],
      },
    },
  };
  const devConfig = {
    ...commonConfig,
  };


  const prodConfig = {
    ...commonConfig,
  };


  return isProd ? prodConfig : devConfig;
});
```

CLI 创建

```
import { defineConfig } from 'vite';
import uni from '@dcloudio/vite-plugin-uni';


import path from 'path';


function resolve(dir) {
  return path.join(__dirname, dir);
}


// https://vitejs.dev/config/
export default defineConfig(({ mode }) => {
  const isProd = mode === 'production';


  const commonConfig = {
    optimizeDeps: {
      include: ['@tencentcloud/tuiroom-engine-wx'],
    },
    plugins: [
      uni(),
    ],
    resolve: {
      alias: {
        '@': resolve('src'),
        '@TUIRoom': resolve('src/roomkit/TUIRoom'),
      },
    },
    build: {
      rollupOptions: {
        external: ['@tencentcloud/tuiroom-engine-wx'],
      },
    },
  };
  const devConfig = {
    ...commonConfig,
  };


  const prodConfig = {
    ...commonConfig,
  };


  return isProd ? prodConfig : devConfig;
});
```

4. 修改 ./tsconfig.json ，项目使用 ts + vue3 开发，如不存在该文件，会产生编译错误。

HBuliderX 创建

```
// tsconfig.json
{
  "compilerOptions": {
    // 与 Vue 的浏览器支持保持一致
    "target": "es5",
    // 这可以对 `this` 上的数据 property 进行更严格的推断
    "strict": true,
    // 如果使用 webpack 2+ 或 rollup，可以利用 tree-shake:
    "module": "es2015",
    "moduleResolution": "node"
  }
}
```

CLI 创建

```
// tsconfig.json
{
  "extends": "@vue/tsconfig/tsconfig.json",
  "compilerOptions": {
    "sourceMap": true,
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"]
    },
    "lib": ["esnext", "dom"],
    "types": ["@dcloudio/types"]
  },
  "include": ["src/**/*.ts", "src/**/*.d.ts", "src/**/*.tsx", "src/**/*.vue"]
}
```

其他文档

- TUIRoomKit
- TUIRoom demo 快速跑通

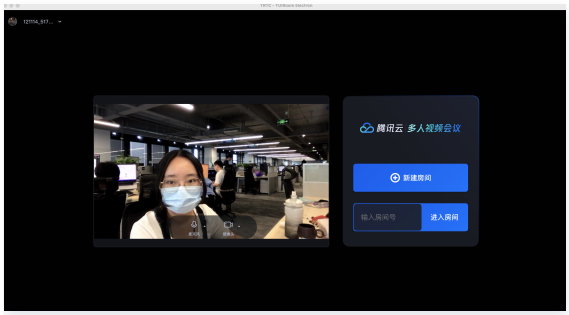
交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和问题反馈。

Electron

最近更新时间：2024-01-24 10:23:11

本文将介绍如何用最短的时间完成 TUIRoomKit 组件的接入，跟随本文档，您将在一小时内完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的多人音视频房间功能。包括成员管理、屏幕分享、聊天、自由发言申请发言模式等功能，如下图所示：





您可以单击在线体验链接： Mac OS版 及 Windows版 下载体验 TUIRoomKit Electron 更多功能。
您也可以单击 Github 下载 TUIRoomKit 代码，并参考 TUIRoomKit Electron 示例工程快速跑通 文档跑通 TUIRoomKit Electron 示例工程。
如需在现有业务中集成 Electron 端 TUIRoomKit 组件，请参考本文档。

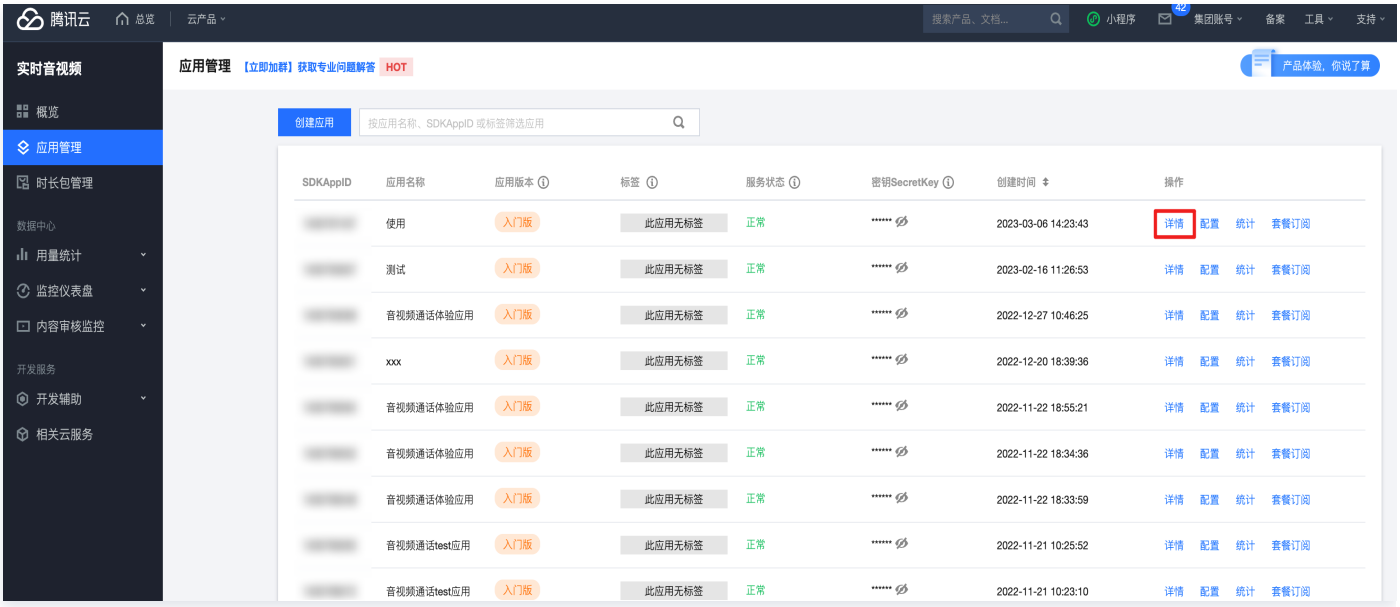
支持的平台

- Windows（PC）
- Mac

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

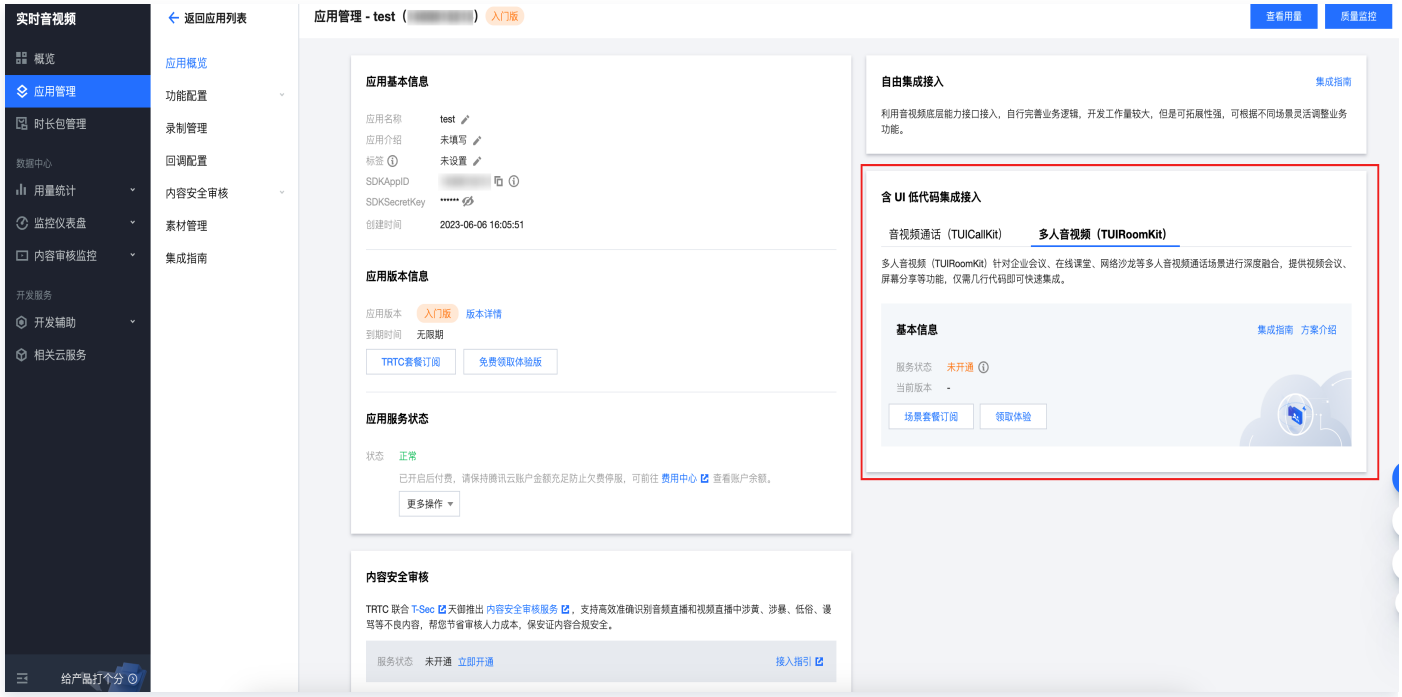
1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），点击详情按钮，进入应用概览界面。



2. 在应用概览页面找到 **含 UI 低代码集成接入** 卡片，选择 **多人音视频（TUIRoomKit）**，点击领取体验按钮，领取7天体验版 TUIRoomKit 进行接入测试。

> **注意：**

- 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
- 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
- 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请点击**场景套餐订阅**按钮或前往 购买页 购买；

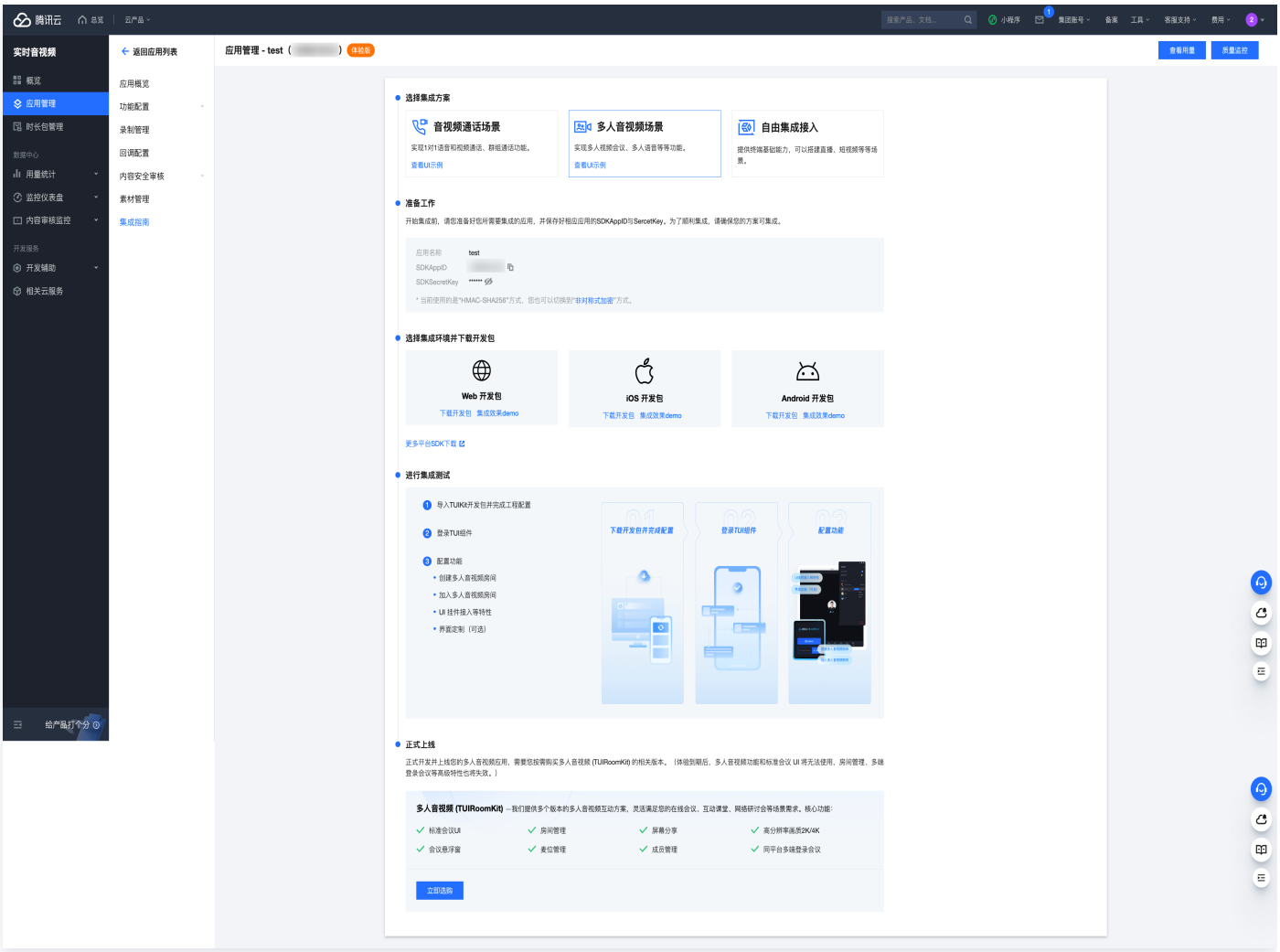




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



4. 单击**集成指南**，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：准备 Vue 工程代码

Vue3 + Vite + TS

1. 打开业务侧已有 Electron + Vue3 + TS 项目，如果无 Electron + Vue3 + TS 项目,可通过此模板 Github 生成 Electron + Vue3 + TS 的模板工程, nodejs 版本大于 14.17.0。

> ⚠ 注意：
> - 本文档介绍的集成步骤基于 electron-vite-vue 模板工程1.0.0版本。
> - electron-vite-vue 模板工程最新版本目录结构有调整，如需使用最新版本，可参照本文档自行调整目录和配置。

2. 克隆生成模板工程后，执行以下脚本：

```
git clone https://github.com/electron-vite/electron-vite-vue.git
cd electron-vite-vue
git checkout v1.0.0
git switch -c TUIRoomKit-quick-start
npm install
npm run dev
```

Vue2 + Webpack +TS

```
// 安装 vue-cli，注意 Vue CLI 4.x 要求 Node.js 为 v10 以上版本
npm install -g @vue/cli
// 创建 Vue2 + Webpack + TS 模板工程
vue create tuiroomkit-demo
```

> ⚠ 注意：
> 执行生成模板工程脚本的过程中，生成模板的方式选择 Manually select features，其余配置选项参考图片。

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Babel, TS, Linter
? Choose a version of Vue.js that you want to start the project with 2.x
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected
polyfills, transpiling JSX)? Yes
? Pick a linter / formatter config: Standard
? Pick additional lint features: Lint on save
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated
config files
```

成功生成 Vue2 + Webpack + TS 模板工程后， 参考 Electron 无UI集成文档，往模版工程中集成 Electron。

> ⚠ 注意：
> 由于 TUIRoom 组件内已经引入了 trtc-electron-sdk ， 在参考 Electron 无UI集成文档 时，可以忽略：

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();
```

这一步骤。

步骤三：下载并引用 TUIRoomkit 组件

1. **下载 TUIRoom 组件代码**
   单击 Github，克隆或下载 TUIRoomKit 仓库代码。
2. **引用 TUIRoom 组件**

Vue3 项目中引入 TUIRoom 组件

1. 复制 `TUIRoomKit/Electron/vue3/packages/renderer/src/TUIRoom` 文件夹到已有项目 `packages/renderer/src/` 目录下，复制 `TUIRoomKit/Electron/vue3/packages/renderer/index.html` 文件夹到已有项目 `packages/renderer/` 目录下。
2. 在页面中引用 TUIRoom 组件。例如：在 `App.vue` 组件中引入 TUIRoom 组件。
   - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
   - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
  <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
  import { ref, onMounted } from 'vue';
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';
  // 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
  const TUIRoomRef = ref();

   onMounted(async () => {
     // 初始化 TUIRoom 组件
     // 主持人在创建房间前需要先初始化 TUIRoom 组件
     // 普通成员在进入房间前需要先初始化 TUIRoom 组件
     await TUIRoomRef.value.init({
       // 获取 sdkAppId 请您参考 步骤一
       sdkAppId: 0,
       // 用户在您业务中的唯一标示 Id
```

```
        userId: '',
        // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig
        // 注意 userSig 与 userId 为一一对应关系
        userSig: '',
        // 用户在您业务中使用的昵称
        userName: '',
        // 用户在您业务中使用的头像链接
        avatarUrl: '',
        // 用户在您业务中需要的皮肤主题颜色及是否支持切换皮肤主题
        theme: {
          defaultTheme:'black',
          isSupportSwitchTheme: true
        }
      })
      // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
      await handleCreateRoom();
    })

    // 主持人创建房间，该方法只在创建房间时调用
    async function handleCreateRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 string
      // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认为'FreeToSpeak'，注意目前仅支持自由发言模式
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id）
      const roomId = '123456';
      const roomMode = 'FreeToSpeak';
      const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
      };
      const createRoomInfo = {
        roomId,
        roomName: 'User defined room name' || roomId,
        roomMode,
        roomParam
      };
      await TUIRoomRef.value.createRoom(createRoomInfo);
    }

    // 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
    async function handleEnterRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 number
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id）
      const roomId = '123456';
      const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
      };
      const enterRoomInfo = {
        roomId,
        roomParam
      };
      await TUIRoomRef.value.enterRoom(enterRoomInfo);
    }
  </script>
```

```
<style>
html, body {
    width: 100%;
    height: 100%;
    margin: 0;
}

#app {
    width: 100%;
    height: 100%;
}
</style>
```

Vue2 项目中引入 TUIRoom 组件

1. 复制 `TUIRoomKit/Electron/vue2/src/TUIRoom` 文件夹到已有项目 src/ 目录下。
2. 在页面中引用 TUIRoom 组件。例如：在 `App.vue` 组件中引入 TUIRoom 组件。
   - TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
   - 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

> ⚠ **注意：**
> 在页面中复制以下代码之后，需要修改 TUIRoom 接口的参数为实际数据。

```
<template>
  <div id="app">
    <room-container ref="TUIRoomRef"></room-container>
  </div>
</template>

<script>
import RoomContainer from '@/TUIRoom/index.vue';
export default {
  name: 'App',
  components: { RoomContainer },
  data() {
    return {};
  },
  async mounted() {
    // 初始化 TUIRoom 组件
    // 主持人在创建房间前需要先初始化 TUIRoom 组件
    // 普通成员在进入房间前需要先初始化 TUIRoom 组件
    await this.$refs.TUIRoomRef.init({
      // 获取 sdkAppId 请您参考 步骤一
      sdkAppId: 0,
      // 用户在您业务中的唯一标示 Id
      userId: '',
      // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig, 注意
userSig 与 userId 为一一对应关系
      userSig: '',
```

```
      // 用户在您业务中使用的昵称
      userName: '',
      // 用户在您业务中使用的头像链接
      avatarUrl: '',
    });
    // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
    await this.handleCreateRoom();
  },
  methods: {
    // 主持人创建房间，该方法只在创建房间时调用
    async handleCreateRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 string
      // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认为'FreeToSpeak'，注意目前仅支持自由发言模式
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
      const roomId = '123456';
      const roomMode = 'FreeToSpeak';
      const roomParam = {
          isOpenCamera: true,
          isOpenMicrophone: true,
      }
      try {
        await this.$refs.TUIRoomRef.createRoom({ roomId, roomName: roomId, roomMode, roomParam });
      } catch (error) {
        alert('TUIRoomKit.createRoom error: ' + error.message);
      }
    },
    // 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
    async handleEnterRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 string
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
      const roomId = '123456';
      const roomParam = {
          isOpenCamera: true,
          isOpenMicrophone: true,
      }
      try {
        await this.$refs.TUIRoomRef.enterRoom({ roomId, roomParam });
      } catch (error) {
        alert('TUIRoomKit.enterRoom error: ' + error.message);
      }
    }
  },
};

</script>

<style lang="scss">
html, body {
  width: 100%;
  height: 100%;
  margin: 0;
}

#app {
  width: 100%;
```

```
  height: 100%;
  * {
    box-sizing: border-box;
  }
}
</style>
```

参数说明

这里详细介绍一下 init 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 步骤一 中的第三步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 步骤一 的第三步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 **辅助工具** 生成一个临时可用的 UserSig。

> ⚠ **注意：**
>
> - 在页面中复制以上代码之后，需要修改 TUIRoom 接口的参数为实际数据。
> - **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
>   - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
>   - UserSig 被错配成了加密密钥（SecretKey），UserSig 是用 SecretKey 把 SDKAppID、UserID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 UserSig。
>   - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID。
> - Github 中的示例代码使用了 getBasicInfo 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 Electron 应用的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 app 在每次使用 TUIRoomKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤四：配置开发环境

TUIRoom 组件引入之后，为了确保项目可以正常运行，需要进行以下配置：

配置 Vue3 + Vite + TS 开发环境

1. 安装依赖
   - 安装开发环境依赖

```
npm install sass typescript unplugin-auto-import unplugin-vue-components --save-dev
```

   - 安装生产环境依赖

```
npm install element-plus events mitt pinia trtc-electron-sdk tim-js-sdk tsignaling vue-i18n @tencentcloud/tuiroom-engine-electron --save
```

2. 注册 Pinia

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 `packages/renderer/src/main.ts` 文件。

```
// src/main.ts 文件
import { createPinia } from 'pinia';

const app = createApp(App);
// 注册Pinia
createApp(App)
  .use(createPinia())
  .mount('#app')
  .$nextTick(window.removeLoading)
```

3. 配置 element-plus 按需引入
   - TUIRoom 使用 element-plus UI 组件，为避免引入所有 element-plus组件，需要您在 packages/renderer/vite.config.ts 中配置 element-plus 组件按需加载, 可参考 packages/renderer/vite.config.ts 文件 。

> ⚠ **注意：**
> 以下配置项为增量配置，不要删除已经存在的 Vite 配置项。

```
// vite.config.ts
import AutoImport from 'unplugin-auto-import/vite';
import Components from 'unplugin-vue-components/vite';
import { ElementPlusResolver } from 'unplugin-vue-components/resolvers';
import { createSvg } from './src/TUIRoom/assets/icons/index.js'
const path = require('path');

export default defineConfig({
  // ...
  resolve: {
    alias: {
      '@': path.resolve(__dirname, 'src'),
      '@TUIRoom': path.resolve(__dirname, 'src/TUIRoom'),
    }
  },
  plugins: [
    // ...
    createSvg(path.join(path.resolve(__dirname, 'src/TUIRoom/assets/icons/svg/'), '/')),
    AutoImport({
      resolvers: [ElementPlusResolver()],
    }),
    Components({
      resolvers: [ElementPlusResolver({
        importStyle: 'sass',
      })],
    }),
  ],
  optimizeDeps: {
    include: ['@tencentcloud/tuiroom-engine-electron']
  },
  build: {
```

```
  // ...
  commonjsOptions: {
    exclude: ['@tencentcloud/tuiroom-engine-electron'],
    include: [],
  },
 }
 // ...
});
```

- 同时为了保证 element-plus 带 UI 组件能够正常显示样式，需要您在入口文件 packages/renderer/src/main.ts 中加载 element-plus 组件样式。

```
// src/main.ts
import 'element-plus/theme-chalk/el-message.css';
import 'element-plus/theme-chalk/el-message-box.css';
```

4. 引入 trtc-electron-sdk

为了在 UI 层以 import 方式引入 trtc-electron-sdk，统一代码风格，否则必须要以 require 的方式引入，需要您在 packages/renderer/vite.config.ts 中配置。以下配置项将 resolve 中的内容替换掉, 可参考 packages/renderer/vite.config.ts 文件。

```
// vite.config.ts

export default defineConfig({
 // ...
 plugins: [
 resolve(
   {
    "trtc-electron-sdk": `
     const TRTCCloud = require("trtc-electron-sdk");
     const TRTCParams = TRTCCloud.TRTCParams;
     const TRTCAppScene = TRTCCloud.TRTCAppScene;
     const TRTCVideoStreamType = TRTCCloud.TRTCVideoStreamType;
     const TRTCScreenCaptureSourceType = TRTCCloud.TRTCScreenCaptureSourceType;
     const TRTCVideoEncParam = TRTCCloud.TRTCVideoEncParam;
     const Rect = TRTCCloud.Rect;
     const TRTCAudioQuality = TRTCCloud.TRTCAudioQuality;
     const TRTCScreenCaptureSourceInfo = TRTCCloud.TRTCScreenCaptureSourceInfo;
     const TRTCDeviceInfo = TRTCCloud.TRTCDeviceInfo;
     const TRTCVideoQosPreference = TRTCCloud.TRTCVideoQosPreference;
     const TRTCQualityInfo = TRTCCloud.TRTCQualityInfo;
     const TRTCQuality = TRTCCloud.TRTCQuality;
     const TRTCStatistics = TRTCCloud.TRTCStatistics;
     const TRTCVolumeInfo = TRTCCloud.TRTCVolumeInfo;
     const TRTCDeviceType = TRTCCloud.TRTCDeviceType;
     const TRTCDeviceState = TRTCCloud.TRTCDeviceState;
     const TRTCBeautyStyle = TRTCCloud.TRTCBeautyStyle;
     const TRTCVideoResolution = TRTCCloud.TRTCVideoResolution;
     const TRTCVideoResolutionMode = TRTCCloud.TRTCVideoResolutionMode;
     const TRTCVideoMirrorType = TRTCCloud.TRTCVideoMirrorType;
     const TRTCVideoRotation = TRTCCloud.TRTCVideoRotation;
     const TRTCVideoFillMode = TRTCCloud.TRTCVideoFillMode;
     const TRTCRoleType = TRTCCloud.TRTCRoleType;
```

```
      const TRTCScreenCaptureProperty = TRTCCloud.TRTCScreenCaptureProperty;
      export {
        TRTCParams,
        TRTCAppScene,
        TRTCVideoStreamType,
        TRTCScreenCaptureSourceType,
        TRTCVideoEncParam,
        Rect,
        TRTCAudioQuality,
        TRTCScreenCaptureSourceInfo,
        TRTCDeviceInfo,
        TRTCVideoQosPreference,
        TRTCQualityInfo,
        TRTCStatistics,
        TRTCVolumeInfo,
        TRTCDeviceType,
        TRTCDeviceState,
        TRTCBeautyStyle,
        TRTCVideoResolution,
        TRTCVideoResolutionMode,
        TRTCVideoMirrorType,
        TRTCVideoRotation,
        TRTCVideoFillMode,
        TRTCRoleType,
        TRTCQuality,
        TRTCScreenCaptureProperty,
      };
      export default TRTCCloud.default;
    `
  }
 ),
 // ...
 ]
 // ...
});
```

5. **env.d.ts 文件配置**

   env.d.ts 文件配置需要您在 `packages/renderer/src/env.d.ts` **中配置。以下配置项为增量配置，不要删除已经存在的** `env.d.ts` **文件配置。**

```
// env.d.ts
declare module '*.vue' {
  import { DefineComponent } from 'vue'
  // eslint-disable-next-line @typescript-eslint/no-explicit-any, @typescript-eslint/ban-types
  const component: DefineComponent<{}, {}, any>
  export default component
}

declare module 'tsignaling/tsignaling-js' {
  import TSignaling from 'tsignaling/tsignaling-js';
  export default TSignaling;
}

declare module 'tim-js-sdk' {
```

```
  import TIM from 'tim-js-sdk';
  export default TIM;
}

declare const Aegis: any;
```

6. 配置中英文切换

TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
// src/main.ts
// 引入 locales/index.ts 文件，请确认引入路径是否正确
import VueI18n from './TUIRoom/locales/index';

createApp(App)
  .use(createPinia())
  .use(VueI18n)
  .mount('#app')
  .$nextTick(window.removeLoading);
```

配置 Vue2 + Webpack + TS 开发环境

1. 安装依赖
   - 安装开发环境依赖

```
npm install sass sass-loader -S -D
```

   - 安装生产环境依赖

```
npm install vue@^2.7 element-ui pinia trtc-electron-sdk tim-js-sdk tsignaling
@tencentcloud/tuiroom-engine-electron vue-i18n@^8 vue-i18n-bridge -S
```

2. 注册 Pinia

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 `src/main.ts` 文件。

```
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
  pinia,
  render: h => h(App),
}).$mount('#app');
```

3. 配置 element-ui

为了保证 element-ui 带 UI 组件能够正常显示样式，需要您在入口文件 `src/main.ts` 或 `src/main.js` 中注册 element-ui 组件并引用其样式文件。

```
import ElementUI from 'element-ui';
import 'element-ui/lib/theme-chalk/index.css';

Vue.use(ElementUI);
```

4. 配置中英文切换
   - TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
import i18n from './TUIRoom/locales/';

Vue.use(i18n);
```

配置 Vue2 + Webpack + JS 开发环境

TUIRoom 默认支持环境为 Vue2 + Webpack + TS 环境，如果您想在Vu2 + Webpack + JS 环境中集成 TUIRoom 组件，您需要做如下配置。

1. 配置 typescript 支持 TUIRoom 组件加载。

```
vue add typescript
```

配置 TS 开发环境的选项可参考图片

```
? Use class-style component syntax? No
? Use Babel alongside TypeScript (required for modern mode, auto-detected
polyfills, transpiling JSX)? Yes
? Convert all .js files to .ts? No
? Allow .js files to be compiled? Yes
? Skip type checking of all declaration files (recommended for apps)? No
```

2. 安装依赖
   - 安装开发环境依赖

```
npm install sass sass-loader -S -D
```

   - 安装生产环境依赖

```
npm install vue@^2.7 element-ui pinia trtc-electron-sdk tim-js-sdk tsignaling
@tencentcloud/tuiroom-engine-electron vue-i18n@^8 vue-i18n-bridge -S
```

3. 注册 Pinia

TUIRoom 使用 Pinia 进行房间数据管理，您需要在项目入口文件中注册 Pinia，项目入口文件为 `src/main.ts` 文件。

```
import { createPinia, PiniaVuePlugin } from 'pinia';

Vue.use(PiniaVuePlugin);
const pinia = createPinia();

new Vue({
```

```
  pinia,
  render: h => h(App),
}).$mount('#app');
```

4. 配置 element-ui

为了保证 element-ui 带 UI 组件能够正常显示样式，需要您在入口文件 `src/main.ts` 中注册 element-ui 组件并引用其样式文件。

```
import ElementUI from 'element-ui';
import 'element-ui/lib/theme-chalk/index.css';

Vue.use(ElementUI);
```

5. 配置中英文切换

TUIRoom 目前支持中英文语言切换，需要您在 main.ts 文件中注册 i18n 实例。

```
import i18n from './TUIRoom/locales/';

Vue.use(i18n);
```

步骤五：开发环境运行

Vue3 + Vite + TS 项目

1. 执行开发环境命令

```
npm run dev
```

> ⚠ **注意：**
> 因 TUIRoom 按需引入 element-plus 组件，会导致开发环境路由页面第一次加载时反应较慢，等待 element-plus 按需加载完成即可正常使用。element-plus 按需加载不会影响打包之后的页面加载。

2. 体验 TUIRoom 组件功能

Vue2 + Webpack + TS 项目

1. 执行开发环境命令

```
npm run start
```

> ⚠ **注意：**
> Electron 默认监听端口号为 8080，若出现打包完成 UI 没有渲染的情况，可能是默认端口被占用，在 `main.electron.js` 中修改对应端口号即可。

2. 体验 TUIRoom 组件功能

步骤六：构建安装包、运行

Vue3 + Vite + TS 项目

在命令行终端中，执行以下命令构建安装包，构建好的安装包位于 release 目录下，可以安装运行。

```
npm run build
```

Vue2 + Webpack + TS 项目

在命令行终端中，执行 package.json 中对应的构建命令，构建好的安装包位于 bin 目录下，可以安装运行。

> ⚠ **注意：**
> 只能使用 Mac 电脑构建 Mac 安装包，使用 Windows 电脑构建 Windows 安装包，暂不支持交叉编译。

常见问题

1. 运行时报错："does not provide an export named 'createBlock' "?

若按照上述步骤接入后，运行时出现以下错误，请您务必将 vite 版本固定在 3.0.0 以下，将 vue 版本固定在 3.4.9 以下。

```
/node_modules/.vite/deps/vue.js?v=b083d236' does not provide an export named 'createBlock' (at App.vue:2:74)
```

原因是项目提供的 electron 工程模版支持的 vite 版本指定在3.0.0以下，但是 vue 的 3.4.9 及以上版本中的属性需要升级 vite 的版本来获取支持，这里已经向 vue 官方提了 issue，https://github.com/vuejs/core/issues/10177 。

其他文档

- TUIRoomKit
- TUIRoomKit demo 快速跑通
- 界面定制(TUIRoomKit)

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云通信官方社群 进行技术交流和问题反馈。

Flutter

最近更新时间：2024-01-11 15:42:52

本文将指导您如何在最短时间内完成 TUIRoomKit 组件的接入。遵循本文档的步骤，您将在一个小时内完成以下关键步骤，并最终实现一个具备完整 UI 界面的音视频会议功能。

环境准备

平台	版本
Flutter	3.0.0 及以上版本。
Android	• 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 及以上的版本（Gradle 3.5.4 及以上的版本）。 • Android 4.1 及以上的手机设备。
iOS	iOS 12.0 及更高。

步骤一：开通服务

在使用 TUIRoomKit 前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

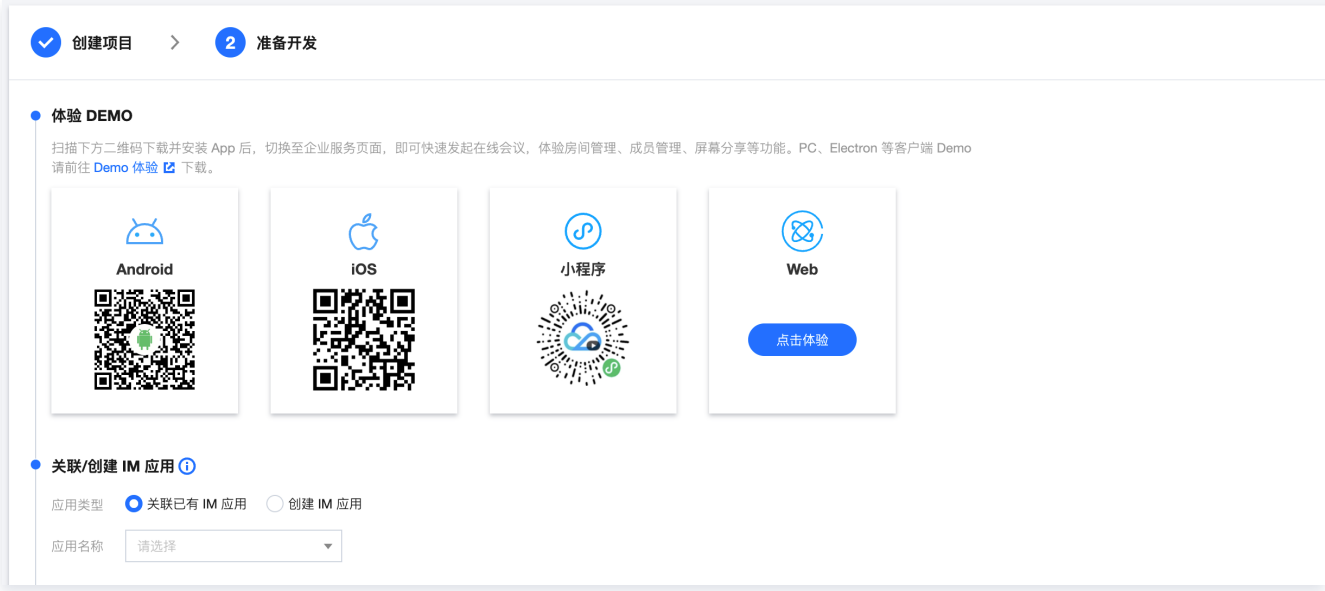
1. 登录 腾讯云视立方 SDK 控制台，单击**创建项目**，选择**多人音视频互动场景**和**集成方式**。



2. 在选定接入场景和集成方式以后，您需要开通**多人音视频房间 SDK** 使用的两项腾讯云基础的 PaaS 能力，即 即时通信 IM 和 实时音视频 TRTC，开通后，单击**创建项目并下一步**。



3. 在项目创建完成以后，您需要为该项目匹配一个 IM 应用。因为多人音视频房间 SDK 依赖了 IM SDK 提供的基础能力，在这里可以创建或者管理已有的 IM 应用。关联成功后，可以领取**7天的免费体验版**，用于后续的开发调试工作。如果您之前已经体验过，可以直接在该页面单击 购买正式版本 。



4. 单击**前往集成**，选择**项目配置**。查看详细的配置页面，找到 SDKAppID 和密钥并记录下来。它们会在后续的 步骤三：登录 TUIRoomKit 组件 中被用到，至此多人音视频房间 SDK 服务开通完成。



步骤二：集成 TUIRoomKit 组件

在您的工程 pubspec.yaml 文件中，添加 **rtc_conference_tui_kit** 插件依赖，或进入您的工程目录下执行如下命令添加依赖。

```
dependencies:
  rtc_conference_tui_kit: 最新版本
```

执行以下命令安装组件：

```
flutter pub get
```

步骤三：完成工程配置

- 由于 **rtc_conference_tui_kit** 组件使用了 GetX 状态管理库进行导航，您需要在您的应用程序中使用 GetMaterialApp 来代替 MaterialApp 。或者您也可以将您 MaterialApp 中的 navigatorKey 属性设置为 Get.key 以实现相同的效果。
- 使用 **Xcode** 打开您的工程，选择**项目** > **Building Settings** > **Deployment**，将其下的 **Strip Style** 设置为 Non-Global Symbols ，以保留所需要的全局符号信息。
- 如您需要在 iOS 端使用音视频功能，需要授权麦克风和摄像头的使用权限（ Android 端已在 SDK 中声明相关权限，您无需手动进行相关配置）。

  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。

  ```
  <key>NSCameraUsageDescription</key>
  <string>TUIRoom需要访问您的相机权限</string>
  <key>NSMicrophoneUsageDescription</key>
  <string>TUIRoom需要访问您的麦克风权限</string>
  ```

  完成以上添加后，在您的 **ios/Podfile** 中添加以下预处理器定义，用于启用相机与麦克风权限。

  ```
  post_install do |installer|
    installer.pods_project.targets.each do |target|
      flutter_additional_ios_build_settings(target)
        target.build_configurations.each do |config|
          config.build_settings['GCC_PREPROCESSOR_DEFINITIONS'] ||= [
        '$(inherited)',
        'PERMISSION_MICROPHONE=1',
        'PERMISSION_CAMERA=1',
        ]
  ```

```
    end
  end
end
```

步骤四：登录组件

在您的项目中添加如下代码，它的作用是通过调用 TUIRoomKit 中的相关接口完成组件的初始化。这个步骤非常关键，因为只有在初始化后才能正常使用 TUIRoomKit 的各项功能，故请您耐心检查相关参数是否配置正确：

```
import 'package:rtc_room_engine/rtc_room_engine.dart';

var result = await TUIRoomEngine.login(
  SDKAPPID, // 请替换为您的SDKAPPID
  'userId', // 请替换为您的User ID
  'userSig',// 请替换为您的userSig
  );

if (result.code == TUIError.success) {
  // login success
} else {
  // login error
}
```

参数说明

这里详细介绍一下 login 函数中所需要用到的几个关键参数：

- **SDKAppID**：在 步骤一 中的第4步中您已经获取到，这里不再赘述。
- **UserID**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
- **UserSig**：使用 步骤一 的第4步中获取的 SecretKey 对 SDKAppID、UserID 等信息进行加密，就可以得到 UserSig，它是一个鉴权用的票据，用于腾讯云识别当前用户是否能够使用 TRTC 的服务。您可以通过控制台中的 辅助工具 生成一个临时可用的 UserSig。

  更多信息请参见 如何计算及使用 UserSig 。

⚠ **注意：**

- **这个步骤也是目前我们收到的开发者反馈最多的步骤，常见问题如下：**
  - SDKAppID 设置错误，国内站的 SDKAppID 一般是以140开头的10位整数。
  - UserSig 被错配成了加密密钥（ SecretKey ），UserSig 是用 SecretKey 把 SDKAppID 、 UserID 以及过期时间等信息加密得来的，而不是直接把 SecretKey 配置成 UserSig 。
  - UserID 被设置成“1”、“123”、“111”等简单字符串，由于 **TRTC 不支持同一个 UserID 多端登录**，所以在多人协作开发时，形如 “1”、“123”、“111” 这样的 UserID 很容易被您的同事占用，导致登录失败，因此我们建议您在调试的时候设置一些辨识度高的 UserID 。
- Github 中的示例代码 使用了 genTestUserSig 函数在本地计算 UserSig 是为了更快地让您跑通当前的接入流程，但该方案会将您的 SecretKey 暴露在 App 的代码当中，这并不利于您后续升级和保护您的 SecretKey，所以我们强烈建议您将 UserSig 的计算逻辑放在服务端进行，并由 App 在每次使用 TUIRoomKit 组件时向您的服务器请求实时计算出的 UserSig。

步骤五：使用组件

设置用户信息

通过调用 TUIRoomKit 的 setSelfInfo 即可设置用户的用户名、头像。

```
import 'package:rtc_conference_tui_kit/rtc_conference_tuikit.dart';

var roomKit = TUIRoomKit.createInstance();
roomKit.setSelfInfo(userName, avatarURL);
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像 URL

创建房间

通过调用 TUIRoomKit 的 createRoom 方法，便可以创建房间。

```
import 'package:rtc_conference_tui_kit/rtc_conference_tuikit.dart';

var roomKit = TUIRoomKit.createInstance();
TUIRoomInfo roomInfo = TUIRoomInfo(roomId: '您的roomId');
var result = await roomKit.createRoom(roomInfo);
if (result.code == TUIError.success) {
  // create room success
} else {
  // create room error
}
```

参数	类型	含义
roomInfo	TUIRoomInfo	房间的基本信息

> **注意：**
> 在调用 createRoom 方法创建房间后，并不会自动进入房间，您需要手动调用进入房间接口 enterRoom 后才能进入房间。

加入房间

通过调用 TUIRoomKit 的 enterRoom 函数并指定房间号，便可以进入指定房间。(**调用该接口后会为您拉起UI界面进入房间**)

```
import 'package:rtc_conference_tui_kit/rtc_conference_tuikit.dart';

var roomKit = TUIRoomKit.createInstance();
var result = await roomKit.enterRoom('roomId',        // 您的room id
                           isOpenMicrophone, // 进房是否开启麦克风
                           isOpenCamera,     // 进房是否开启摄像头
                           userSpeaker);     // 进房是否使用扬声器播放声音
if (result.code == TUIError.success) {
  // enter room success
} else {
  // enter room success
}
```

参数	类型	含义

roomid	String	需要加入的房间号
isOpenMicrophone	bool	进房是否开启麦克风
isOpenCamera	bool	进房是否开启摄像头
userSpeaker	bool	进房是否使用扬声器播放声音

更多功能

TUIRoomEngine SDK 提供了丰富音视频房间功能，具体内容请参考 API 概览 。

交流与反馈

如果您在使用过程中有遇到什么问题，欢迎提交 issue ，我们也欢迎您加入我们的 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和反馈问题.

Windows

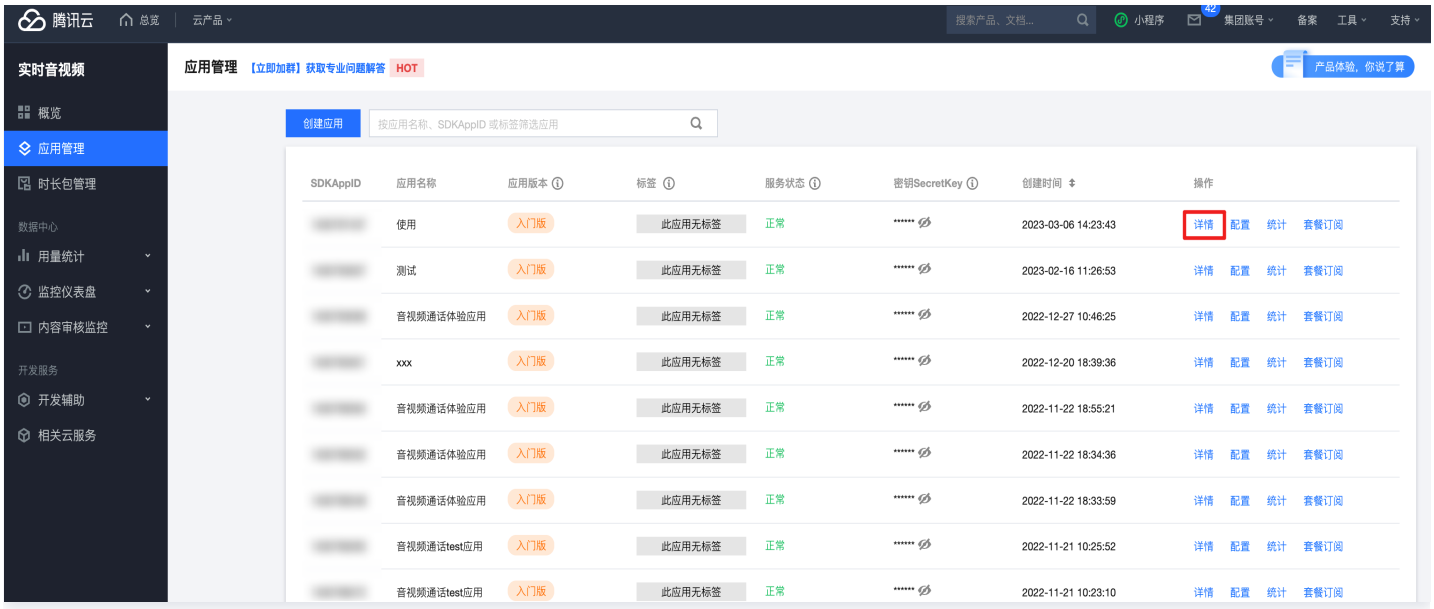
最近更新时间：2024-01-11 15:42:52

本文将介绍如何用最短的时间完成 TUIRoomKit 组件的接入，跟随本文档，您将完成如下几个关键步骤，并最终得到一个包含完备 UI 界面的开源音视频组件,通过集成 TUIRoomKit，您可以在业务中快速上线音视频房间，屏幕分享，聊天等功能。

步骤一：开通服务

在使用 TUIRoomKit 发起会议前，您需要开通 TUIRoomKit 专属的多人音视频互动服务，详细步骤如下：

1. 登录 实时音视频 TRTC 控制台，单击左侧应用管理页面，找到需要开通 TUIRoomKit 的应用（SDKAppID），点击详情按钮，进入应用概览界面。



2. 在应用概览页面找到 **含 UI 低代码集成接入** 卡片，选择 **多人音视频（TUIRoomKit）**，点击领取体验按钮，领取7天体验版 TUIRoomKit 进行接入测试。

注意：

- 领取体验版后仅开通 TUIRoomKit 7天的体验资格，测试过程中所产生的音视频时长等资源消耗，仍会按照实时音视频 TRTC 标准计费规则计费；
- 新账号首次可前往 试用中心 免费领取10000分钟音视频时长；
- 如果所选 SDKAppID 体验版领取次数已达上限，需要购买 TUIRoomKit 包月套餐才能开通服务，请点击**场景套餐订阅**按钮或前往 购买页 购买；

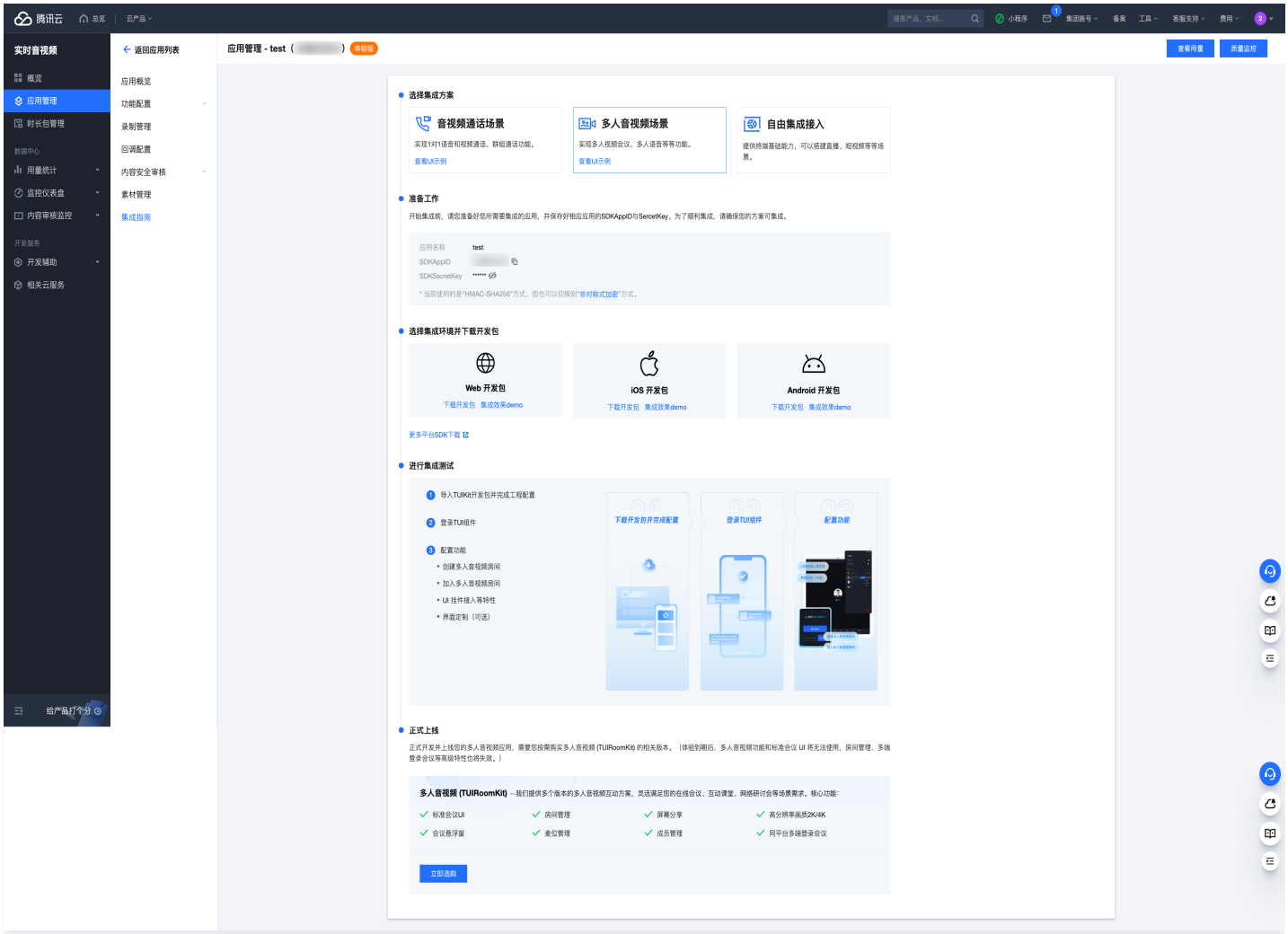




3. 领取完成后，可以看到体验版的基本信息，包括服务状态、版本信息和功能详情、到期时间。



4. 单击**集成指南**按钮，即可参照集成指南开始集成。至此 TUIRoomKit 服务开通完成。



步骤二：下载 TUIRoomKit 工程代码

1. **下载 TUIRoomKit 组件代码**

   单击 Github，克隆或下载 TUIRoomKit 仓库代码，Windows-Mac 目录为 Windows 工程所在目录。

2. **修改用户配置**

   进入 Windows-Mac 目录后，打开 `utils\usersig\win\GenerateTestUserSig.h` 文件，修改 SDKAppID 和 密钥。

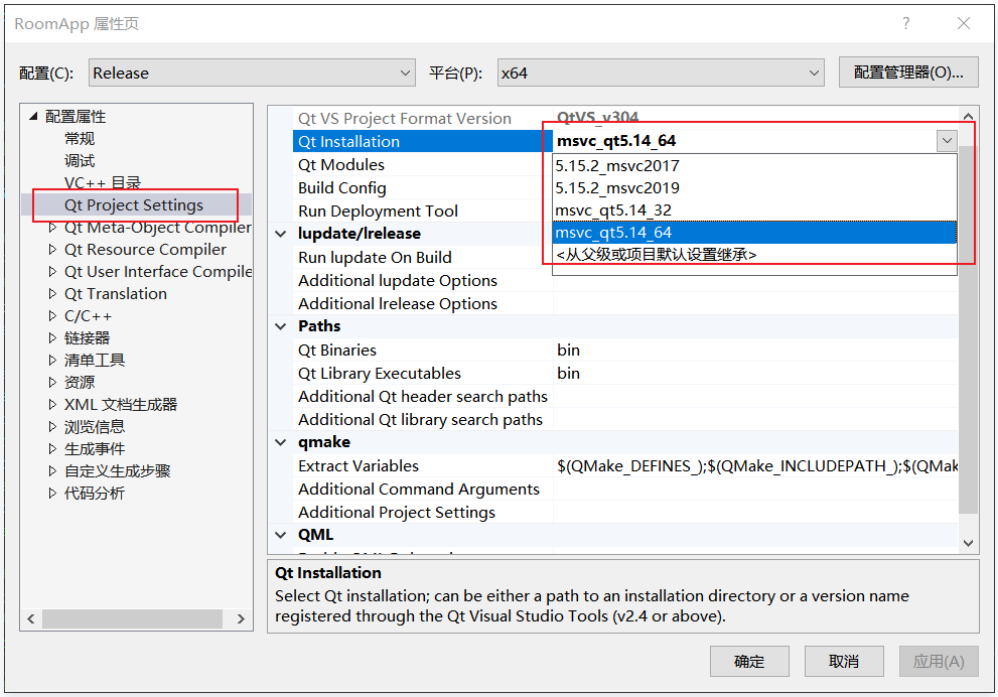
步骤三：配置开发环境

Windows 环境：

- Visual Studio 2015及以上集成开发环境。
- QT5.9.1及以上版本的 Qt 开发库。
- VS 下的 QT 开发插件 Qt Visual Studio Tools 2.2.0及以上。
- 最低支持系统：Windows 8.1。
- 请确保您的集成开发环境能够正常开发。

步骤四：配置 TUIRoomKit 工程

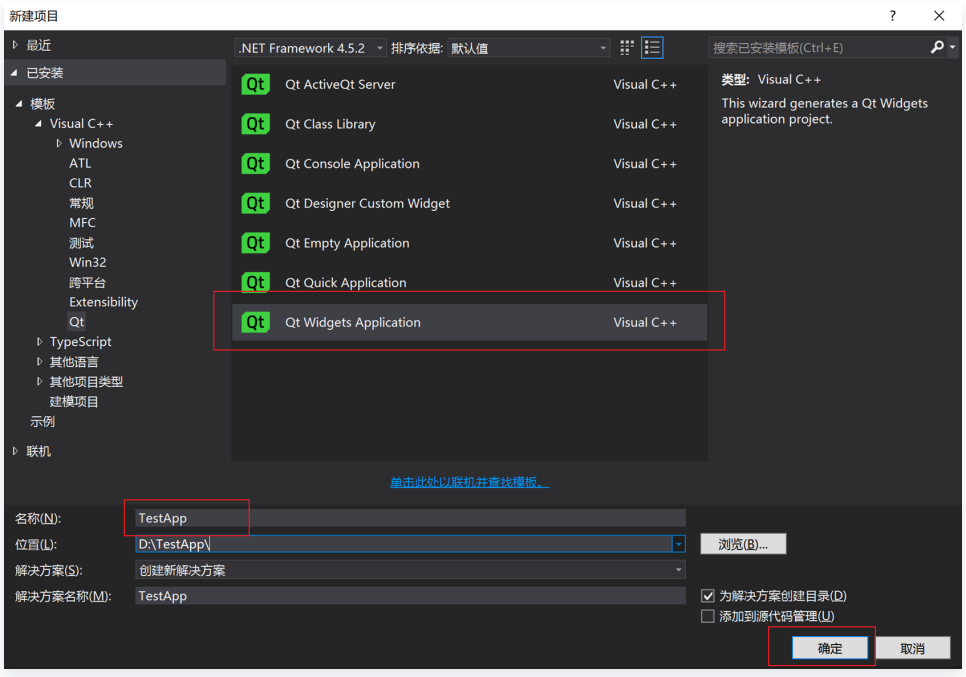
1. 进入 RoomKit 目录，使用 Visual Studio 打开 `RoomKit.vcxproj` 文件，打开 TUIRoomKit 工程。
2. 配置 QT 开发设置：



3. 编译 TUIRoomKit 工程，生成 RoomKit.exe 程序。

步骤五：外部启动 TUIRoomKit 程序

1. 打开 VS，选择 Qt Widgets Application 工程类型，创建 TestApp 工程。



2. 编写启动进程的程序，并在合适的位置调用 LoadRoomApp 函数。

```
#include <QProcess>
#include <QApplication>
void LoadRoomApp() {
  QString executable_file_path = QApplication::applicationDirPath();
```

```
    QString app_path = executable_file_path + "/RoomKit.exe";
    QProcess::startDetached(app_path);
}
```

```
#include "TestApp.h"

#include <QProcess>                    启动RoomApp
#include <QApplication>
void LoadRoomApp() {
    QString executable_file_path = QApplication::applicationDirPath();
    QString app_path = executable_file_path + "/RoomApp.exe";
    QProcess::startDetached(app_path);
}

TestApp::TestApp(QWidget *parent)
    : QMainWindow(parent) {
    ui.setupUi(this);
    LoadRoomApp();          TestApp启动的时候启动RoomApp
}
```

3. 编译项目，并将 RoomKit 编译的成果物复制到当前可执行程序目录，以 release x86 程序为例：
   复制 `TUIRoomKit\Windows-Mac\RoomKit\bin\Win32\Release` 目录下所有文件到当前程序目录下。
4. 执行程序，启动 TestApp 的同时启动 RoomKit。

其他文档

- TUIRoomEngine API
- TUIRoomKit demo 快速跑通

交流与反馈

您在接入或使用过程有任何需要或者反馈，欢迎加入 腾讯云实时音视频 TRTC 技术交流群 进行技术交流和问题反馈。

界面定制 (TUIRoomKit)
Android

最近更新时间：2023-11-21 15:54:01

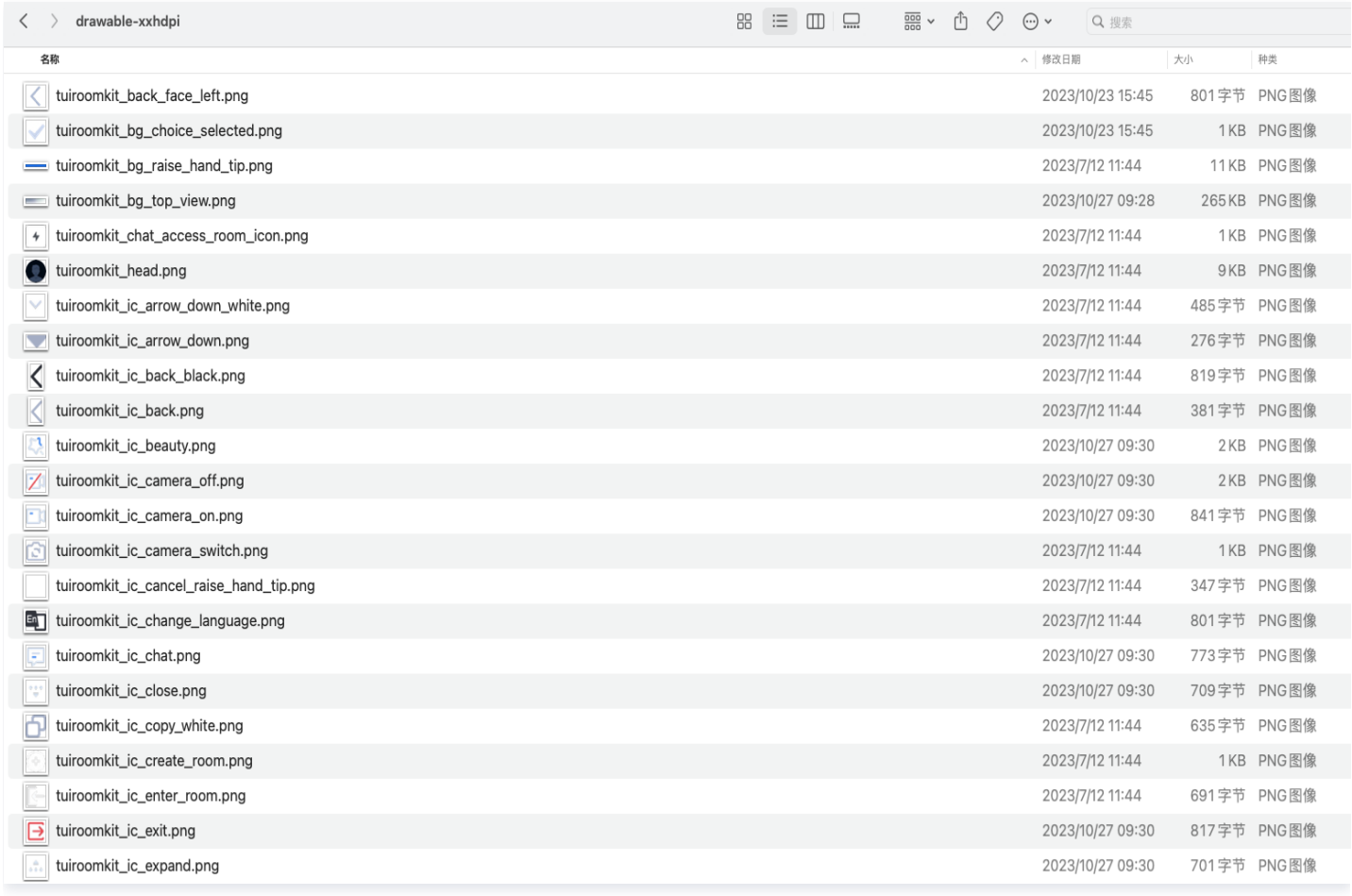
本文将介绍如何定制 TUIRoomKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUIRoomKit 的用户界面进行调整，TUIRoomKit 的界面源代码位于 Github 中的 `Android/tuiroomkit` 文件夹下面：

替换图标

您可以直接替换 `src/res/drawable-xxhdpi` 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换文案

您可以通过修改 values-zh 和 values-en 中的 `strings.xml` 文件来修改视频会议界面的字符串内容。

方案二：自定义部分 UI 方案

您可以通过修改 `src/main/java/com/tencent/cloud/tuikit/roomkit/view` 文件下的不同页面来调整多人视频会议界面的 UI 布局：

```
view
├── component
│   ├── BaseBottomDialog.java              //通用底部弹框
│   ├── ConfirmDialog.java                 //通用确认对话框
│   ├── QRCodeView.java                    //通用二维码空间
│   └── RoundRelativeLayout.java           //通用圆角 RelativeLayout
└── Page
    ├── RoomMaiActivity.java               //会议主界面Activity
    ├── RoomMainView.java                  //会议主界面View
    ├── RoomWindowManager.java             //会议主界面和悬浮窗切换的管理者
    └── Widget
        ├── BottomNavigationBar            //底部栏
        │   ├── BottomLayout.java
        │   └── BottomView.java
        ├── Chat                           //聊天弹窗
        │   └── ChatActivity.java
        ├── Dialog                         //特定弹框
        │   ├── ExitRoomDialog.java
        │   ├── InviteUserDialog.java
        │   └── RoomInfoDialog.java
        ├── FloatWindow                    //悬浮窗
        │   ├── ScreenSharingIndicate      //屏幕分享指示悬浮窗
        │   │   └── ScreenSharingIndicateFloatView.java
        │   └── VideoPlaying               //视频播放悬浮窗
        │       ├── RoomFloatViewService.java
        │       └── RoomVideoFloatView.java
        ├── LocalAudioIndicator            //本地本音状态显示
        │   ├── LocalAudioToggleView.java
        │   ├── LocalAudioToggleViewAction.java
        │   └── LocalAudioToggleViewResponder.java
        ├── MediaSettings                  //媒体设置，包括音频和视频的设置
        │   ├── MediaSettingPanel.java
        │   ├── VideoFrameRateChoicePanel.java
        │   └── VideoResolutionChoicePanel.java
        ├── RaiseHandControlPanel          //举手用户管理面板
        │   ├── RaiseHandApplicationListAdapter.java
        │   └── RaiseHandApplicationListPanel.java
        ├── TopNavigationBar               //顶部栏
        │   └── TopView.java
        ├── TransferOwnerControlPanel      //房间转让选择面板
        │   ├── TransferMasterPanel.java
        │   └── TransferOwnerAdapter.java
        └── UserControlPanel               //用户管理面板
            ├── UserListAdapter.java
            ├── UserListPanel.java
            └── UserManagementView.java
```

底部按钮 BottomView 的修改

为了方便您自定义底部的功能按钮，我们的 BottomView 是通过读取 list 来自动构造，以开关视频按钮为例子，代码如下：

```
private BottomItemData createCameraItem() {
  BottomItemData cameraItemData = new BottomItemData();
  //设置按钮类型用于区分不同按钮
```

```
    cameraItemData.setType(BottomItemData.Type.VIDEO);
    //设置按钮是否可点击
    if (isOwner()) {
      cameraItemData.setEnable(true);
    } else if (mRoomStore.roomInfo.enableSeatControl) {
      cameraItemData.setEnable(false);
    } else {
      cameraItemData.setEnable(mRoomStore.roomInfo.enableVideo);
    }
    //设置按钮默认图标
    cameraItemData.setIconId(R.drawable.tuiroomkit_ic_camera_off);
    //设置按钮背景图片
    cameraItemData.setBackground(R.drawable.tuiroomkit_bg_bottom_item_black);
    //设置按钮不可点击时的图标
    cameraItemData.setDisableIconId(R.drawable.tuiroomkit_ic_camera_off);
    //设置按钮默认图标
    cameraItemData.setName(mContext.getString(R.string.tuiroomkit_item_open_camera));

    //按钮点击效果，如果您的按钮点击时需要切换图片/名称等，需要构造此数据
    BottomSelectItemData camaraSelectItemData = new BottomSelectItemData();
    //设置按钮选中时的名称
    camaraSelectItemData.setSelectedName(mContext.getString(R.string.tuiroomkit_item_close_camera));
    //设置按钮未选中时的名称
    camaraSelectItemData.setUnSelectedName(mContext.getString(R.string.tuiroomkit_item_open_camera));
    //设置按钮是否选中
    camaraSelectItemData.setSelected(false);
    //设置按钮选中时的图标
    camaraSelectItemData.setSelectedIconId(R.drawable.tuiroomkit_ic_camera_on);
    //设置按钮未选中时的图标
    camaraSelectItemData.setUnSelectedIconId(R.drawable.tuiroomkit_ic_camera_off);
    //设置按钮选中/未选中时的点击事件
    camaraSelectItemData.setOnItemSelectListener(new BottomSelectItemData.OnItemSelectListener() {
        @Override
        public void onItemSelected(boolean isSelected) {
            enableCamera(isSelected);
        }
    });
    cameraItemData.setSelectItemData(camaraSelectItemData);
    return cameraItemData;
    }
```

方案三：自定义全部 UI 方案

TUIRoomKit 的整体功能是基于 TUIRoomEngine 这个无 UI SDK实现的，您可以完全基于 TUIRoomEngine 实现一套自己的 UI 界面。详情可见 TUIRoomEngine API 接口地址 。

iOS

最近更新时间：2023-11-17 15:02:42

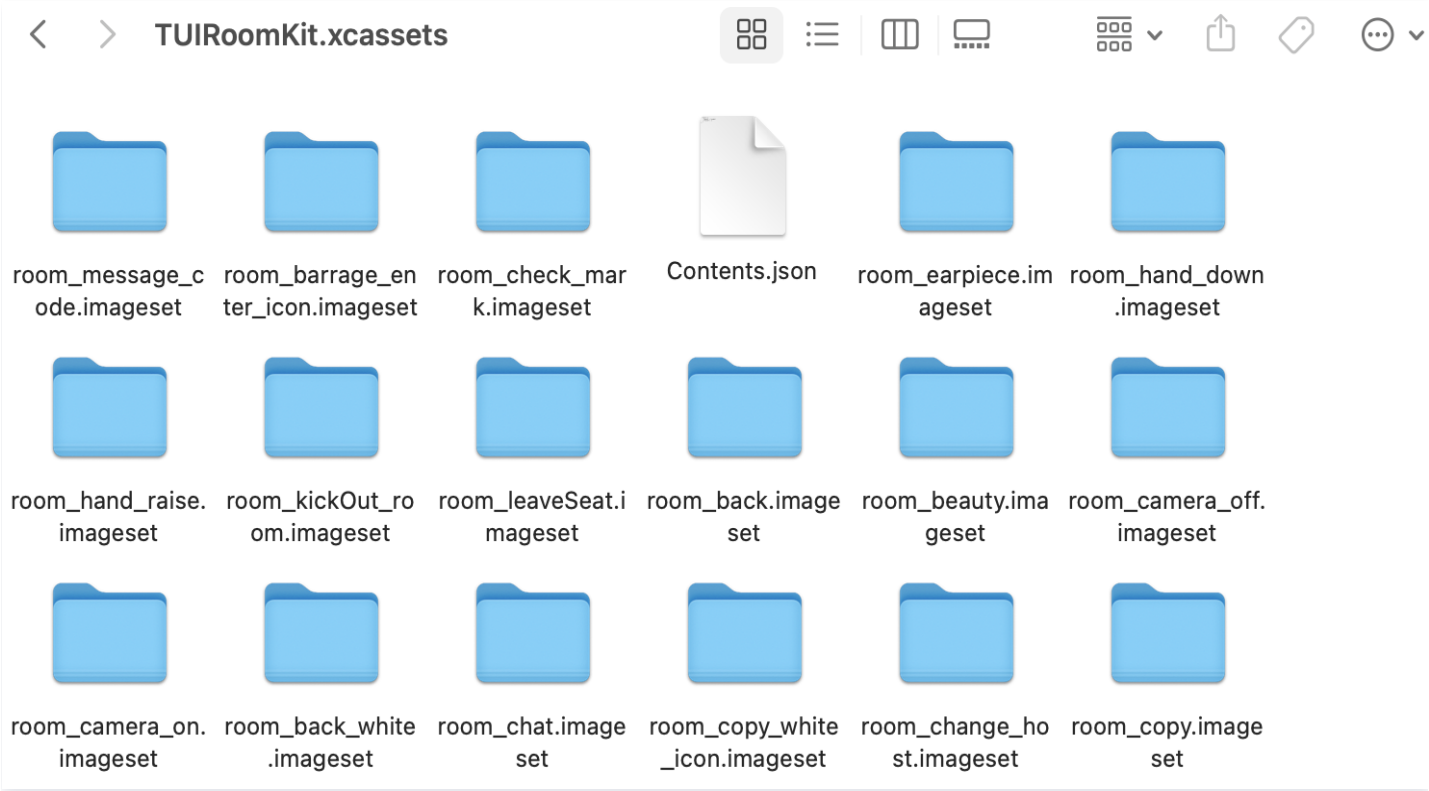
本文将介绍如何定制 TUIRoomKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自定义 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUIRoomKit 的用户界面进行调整，TUIRoomKit 的界面源代码位于 Github 中的 iOS/ 文件夹下面：

替换图标

您可以直接修改 **TUIRoomKit/Resources/TUIRoomKit.xcassets** 文件夹下的图标组件，以确保整个 app 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换文案

您可以通过修改 `TUIRoomKit/Resources/Localized` 文件中 values-zh 和 values-en 的 `strings.xml` 文件来修改视频会议界面的字符串内容。

调整 UI 布局

您可以通过修改 `TUIRoomKit/Source/View` 文件下的不同页面来调整多人视频会议界面的 UI 布局：

```
├── Component
│   ├── ButtonItemView.swift              //通用View
│   └── ListCellItemView.swift            //通用View
└── Page
```

```
    ├── RoomMainView.swift                    //会议主界面View
    ├── RoomMainViewController.swift          //会议主界面ViewController
    ├── RoomRouter.swift                      //路由
    └── Widget
        ├── BottomNavigationBar               //底部栏
        │   ├── BottomItemView.swift
        │   └── BottomView.swift
        ├── FloatWindow                       //悬浮窗
        │   ├── RoomUserStatusView.swift
        │   └── RoomVideoFloatView.swift
        ├── MediaSettings                     //设置
        │   ├── MediaSettingView.swift
        │   └── VideoChoicePanel.swift
        ├── TopNavigationBar                  //顶部栏
        │   ├── TopItemView.swift
        │   └── TopView.swift
        ├── UserControlPanel                  //用户列表
        │   ├── UserListManagerView.swift
        │   └── UserListView.swift
        ├── RaiseHandControlPanel             //举手用户管理列表
        │   └── RaiseHandApplicationListView.swift
        ├── TransferOwnerControlPanel         //房间转让列表
        │   └── TransferMasterView.swift
        ├── PopUpControlPanel                 //通用弹框
        │   ├── PopUpView.swift
        │   └── PopUpViewController.swift
        └── VideoSeat                         //视频显示
            ├── ScreenCaptureMaskView.swift
            ├── TUIVideoSeatCell.swift
            ├── TUIVideoSeatLayout.swift
            ├── TUIVideoSeatUserStatusView.swift
            └── TUIVideoSeatView.swift
```

底部按钮 BottomView 的修改

为了方便您自定义底部的功能按钮，我们的 BottomView 是通过读取 list 来自动构造，以开关视频按钮为例子，代码如下：

```
func createBottomData() {
    let muteVideoItem = ButtonItemData()
    //设置按钮默认标题
    muteVideoItem.normalTitle = .unMuteVideoText
    //设置按钮点击后标题
    muteVideoItem.selectedTitle = .muteVideoText
    //设置按钮默认图标
    muteVideoItem.normalIcon = "room_camera_on"
    //设置按钮点击后图标
    muteVideoItem.selectedIcon = "room_camera_off"
    //设置按钮图片资源获取处
    muteVideoItem.resourceBundle = tuiRoomKitBundle()
    //设置按钮是否点击
    muteVideoItem.isSelect = !(roomInfo.isOpenCamera)
    //设置按钮类型用于区分不同按钮
    muteVideoItem.buttonType = .muteVideoItemType
    //设置按钮点击事件
```

```
        muteVideoItem.action = { [weak self] sender in
            guard let self = self, let button = sender as? UIButton else { return }
            self.muteVideoAction(sender: button)
        }
    }
```

方案二：自定义 UI 方案

TUIRoomKit 的整体功能是基于 TUIRoomEngine 这个无 UI SDK 实现的，您可以完全基于 TUIRoomEngine 实现一套自己的 UI 界面。详情可见：
TUIRoomEngine API 接口地址 。

Web

最近更新时间：2023-07-13 17:06:32

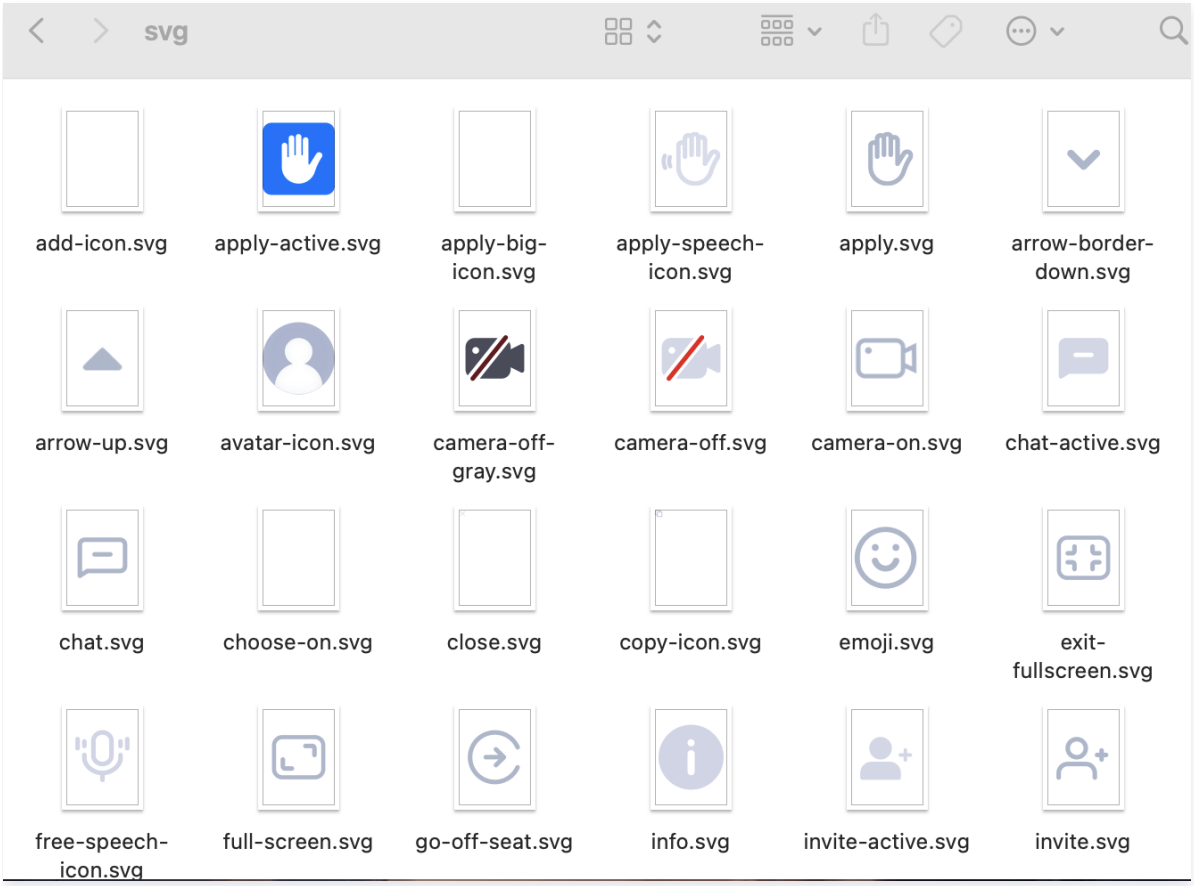
本文将介绍如何定制 TUIRoomkit 的用户界面，我们提供了两个方案供您选择：换肤方案和自定义 UI 方案。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUIRoom 的用户界面进行调整，TUIRoom 的界面源代码位于 Github 中的 Web/ 文件夹下面：

替换图标

您可以直接修改 **Web/vue3/src/TUIRoom/assests/icons/svg** 文件夹下的图标组件，以确保整个 app 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



调整 UI 布局

您可以通过修改 src/components/ 文件下的不同页面来调整多人视频会议界面的UI布局

```
- components/
    - base
    - Chat              聊天
    - common            公用图标组件
    - ManageMember      成员管理
    - RoomContent       Room 视频
    - RoomFooter        Room 页面 Footer部分
    - RoomHeader        Room 页面 Header部分
    - RoomHome          Home 页面
```

```
- RoomInvite        邀请成员
- RoomLogin         登录页面
- RoomMore          更多
- RoomSetting       设置
- RoomSidebar       抽屉组件
```

方案二：自定义 UI 方案

TUIRoom 的整体功能是基于 TUIRoomEngine 这个无 UI SDK实现的，您可以完全基于 TUIRoomEngine 实现一套自己的 UI 界面。详情可见

- TUIRoomEngine 接入指引
- TUIRoomEngine API 接口地址

Electron

最近更新时间：2023-04-18 19:41:29

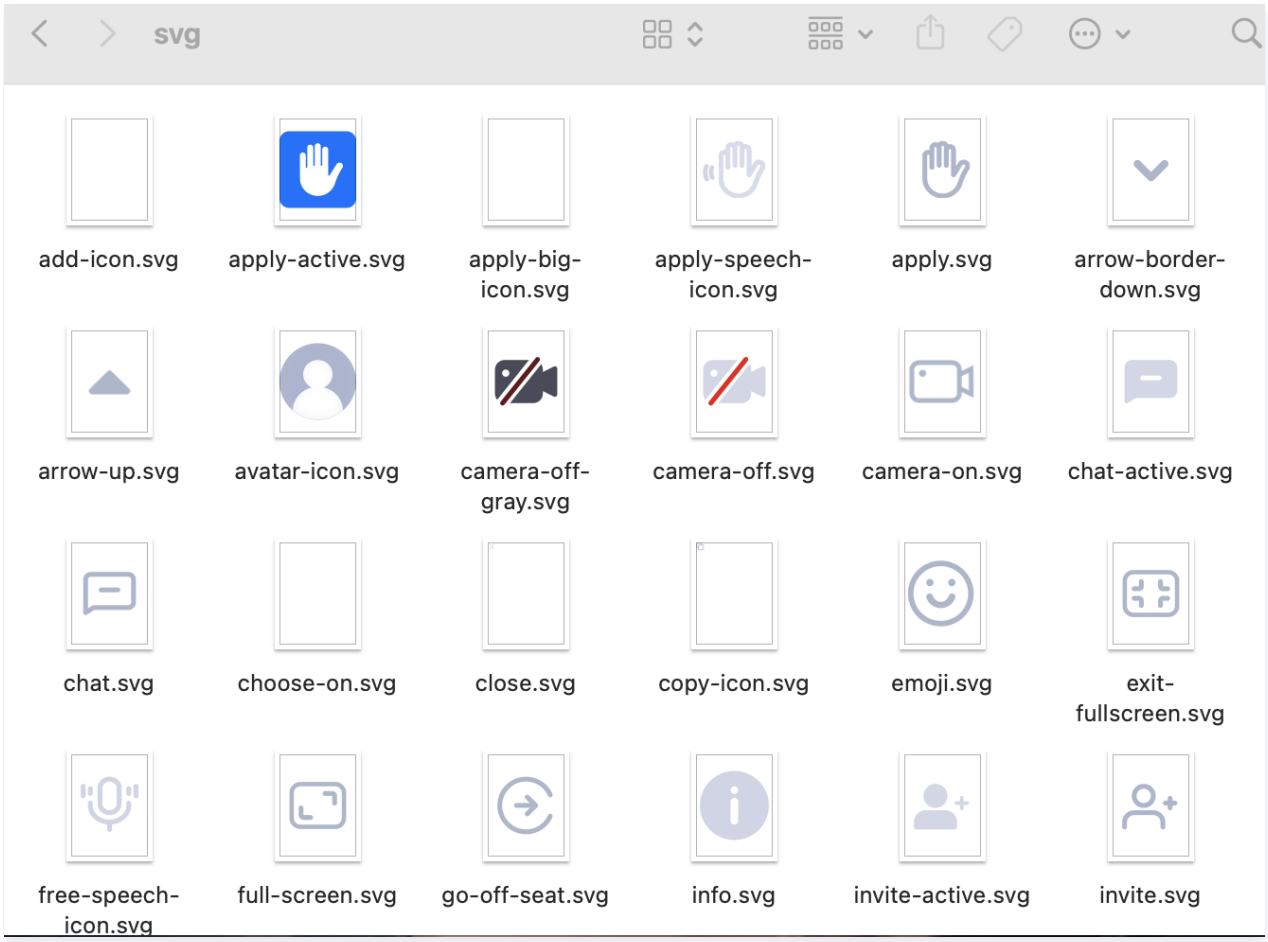
本文将介绍如何定制 TUIRoomkit 的用户界面，我们提供了两个方案供您选择：换肤方案和自定义 UI 方案。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUIRoomKit 的用户界面进行调整，TUIRoomKit 的界面源代码位于 Github 中的 Electron/ 文件夹下面：

替换图标

您可以直接修改 `packages/renderer/src/TUIRoom/assests/icons/svg` 文件夹下的图标组件，以确保整个 app 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



调整 UI 布局

您可以通过修改 `packages/renderer/src/TUIRoom/components/` 文件下的不同页面来调整多人视频会议界面的UI布局

```
- components/
    - base
    - Chat              聊天
    - common            公用图标组件
    - ManageMember      成员管理
    - RoomContent       Room视频
    - RoomFooter        Room页面Footer部分
```

```
- RoomHeader        Room页面Header部分
- RoomHome          Home页面
- RoomInvite        邀请成员
- RoomLogin         登录页面
- RoomMore          更多
- RoomSetting       设置
- RoomSidebar       抽屉组件
```

方案二：自定义 UI 方案

TUIRoom 的整体功能是基于 TUIRoomEngine 这个无 UI SDK实现的，您可以完全基于 TUIRoomEngine 实现一套自己的 UI 界面。详情可见

- TUIRoomEngine 接入指引
- TUIRoomEngine API 接口地址

Windows

最近更新时间：2023-04-18 19:41:30

本文将介绍如何定制 TUIRoomKit 的用户界面，我们提供了两个方案供您选择：换肤方案和自定义 UI 方案。

方案一：界面微调方案

- 您可以直接基于我们提供的 App 进行修改适配
- App 目录包含了 UI 相关的设计与逻辑，您可以根据需求，修改 RoomApp 源码进行二次开发，主要功能点如下：

功能点	文件目录
首页登录	Windows-Mac\RoomApp\App\LoginViewController.cpp
设备检测	Windows-Mac\RoomApp\App\PresetDeviceController.cpp
主页面	Windows-Mac\RoomApp\App\MainWindow.cpp
麦上列表	Windows-Mac\RoomApp\App\StageListController.cpp
成员列表	Windows-Mac\RoomApp\App\MemberListViewController.cpp
设置页面	Windows-Mac\RoomApp\App\SettingViewController.cpp
聊天室	Windows-Mac\RoomApp\App\ChatRoomViewController.cpp
屏幕分享	Windows-Mac\RoomApp\App\ScreenShareWindow.cpp
底部工具栏	Windows-Mac\RoomApp\App\BottomBarController.cpp

方案二：自定义 UI 方案

TUIRoomKit 的整体功能是基于 TUIRoomEngine 这个无 UI SDK实现的，您可以完全基于 TUIRoomEngine 实现一套自己的 UI 界面。详情可见 TUIRoomEngine API 接口地址 。

Flutter

最近更新时间：2023-11-17 11:00:41

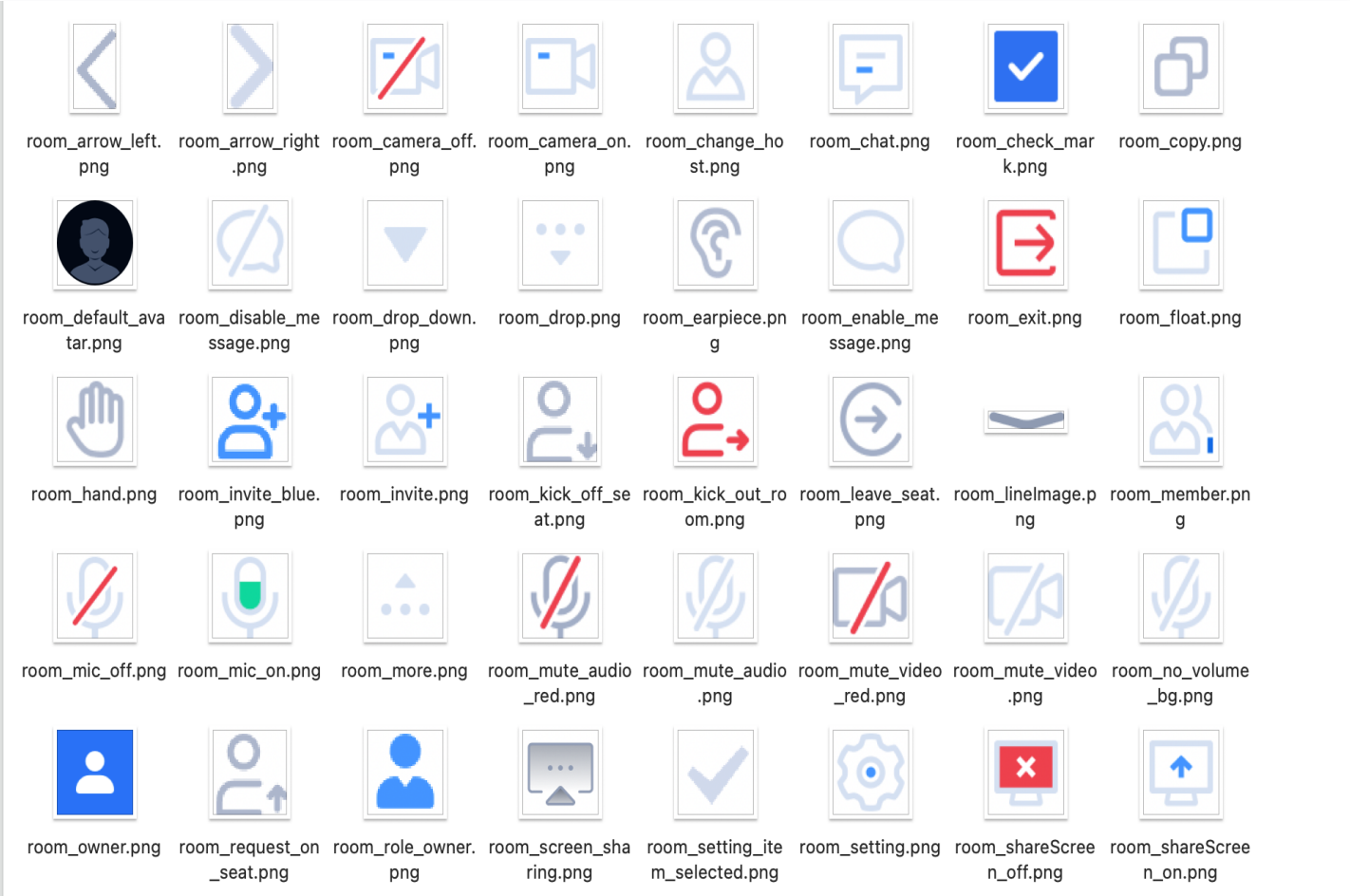
本文将介绍如何定制 TUIRoomKit 的用户界面，我们提供了两个方案供您选择：**界面微调方案**和**自实现 UI 方案**。

方案一：界面微调方案

通过直接修改我们提供的 UI 源代码，对 TUIRoomKit 的用户界面进行调整。

替换图标

您可以直接替换 rtc_conference_tui_kit/assets/images 文件夹下的图标，以确保整个 App 中的图标色调风格保持一致，请在替换时保持图标文件的名字不变。



替换文案

您可以通过修改 rtc_conference_tui_kit/lib/common/languages 目录下的 en_us.dart 和 zh_cn.dart 文件来修改视频会议界面的字符串内容。

方案二：自定义部分 UI 方案

TUIRoomKit 的各 UI 组件文件夹如下，您可根据自身的实际业务需求来修改相应文件夹下的 view.dart 来自定义 UI 。

```
├── lib
├── common
│   └── widgets
│       ├── bottom_sheet.dart (<1 KB)              // 通用底部弹窗
│       ├── copy_text_button.dart (1 KB)           // 通用文本复制按钮
│       ├── dialog.dart (3 KB)                     // 通用对话框弹窗
│       ├── drop_down_button.dart (<1 KB)          // 通用下拉按钮
│       ├── info_list.dart (1 KB)                  // 通用信息展示item
│       ├── invite_sheet.dart (1 KB)               // 通用邀请弹窗
│       ├── rounded_container.dart (<1 KB)         // 通用圆角container
│       ├── search_bar.dart (1 KB)                 // 通用搜索框
│       ├── single_select_list.dart (1 KB)         // 通用单选列表
│       ├── slider.dart (1 KB)                     // 通用滑动条
│       ├── switch.dart (<1 KB)                    // 通用开关按钮
│       ├── toast.dart (<1 KB)                     // 通用toast提示
│       └── user_info.dart (2 KB)                  // 通用用户信息item组件
├── pages
│   └── conference_main
│       ├── view.dart
│       └── widgets
│           ├── bottom_view
│           │   ├── view.dart (1 KB)               // 底部功能按钮文件
│           │   └── widgets
│           │       ├── base_button.dart (7 KB)    // 未展开的基础底部功能按钮
│           │       ├── bottom_button_item.dart (1 KB)  // 封装的通用底部单个item按钮
│           │       └── more_button.dart (1 KB)    // 展开后的所有底部功能按钮
│           ├── exit
│           │   └── view.dart (2 KB)               // 退出房间底部弹窗
│           ├── local_screen_sharing.dart (1 KB)   // 本地开屏幕共享后屏幕中间屏幕共享中提示组件
│           ├── raise_hand_list
│           │   ├── view.dart (2 KB)               // 举手申请列表（尽在举手发言模式房主端显示）
│           │   └── widgets
│           │       ├── title.dart (<1 KB)         // 列表标题栏
│           │       ├── user_item.dart (1 KB)      // 封装的列表组件的单个item
│           │       └── user_table.dart (1 KB)     // 列表组件
│           ├── setting
│           │   ├── view.dart (1 KB)               // 设置面板
│           │   └── widgets
│           │       ├── audio_setting.dart (1 KB)。    // 音频设置组件
│           │       ├── setting_info_select.dart (1 KB)。// 单选值设置组件，如分辨率设置
│           │       ├── setting_item.dart (<1 KB)。    // 设置面板单个设置项item
│           │       └── video_setting.dart (4 KB)      // 视频设置
│           ├── top_view
│           │   ├── view.dart (2 KB)               // 顶部功能按钮
│           │   └── widgets
│           │       ├── meeting_title.dart (1 KB)      // 顶部会议信息显示组件
│           │       ├── room_info_sheet.dart (2 KB)。   // 详细会议信息底部弹窗
│           │       └── top_button_item.dart (1 KB)    // 顶部功能单个通用item按钮
│           ├── transfer_host
│           │   ├── view.dart (1 KB)               // 转让房主面板
│           │   └── widgets
│           │       ├── title.dart (<1 KB)         // 转让房主面板标题栏
│           │       ├── user_item.dart (1 KB)      // 可转让的房主用户列表单个item
│           │       └── user_table.dart (1 KB)     // 可转让的房主用户列表
│           ├── user_list
```

```
│   │   ├── view.dart (3 KB)                    // 成员列表
│   │   └── widgets
│   │       ├── button_item.dart (1 KB)         // 成员列表底部按钮item
│   │       ├── user_control.dart (6 KB)        // 点击单个成员后弹出的成员管理页面
│   │       ├── user_control_item.dart (1 KB)   // 成员管理页面item
│   │       ├── user_item.dart (3 KB)           // 成员列表item
│   │       └── user_table.dart (3 KB)          // 成员列表页面列表
│   ├── video_seat
│   │   ├── video_item                          // 单个视频画面item文件夹
│   │   │   ├── view.dart (3 KB)                // 单个视频画面item
│   │   │   └── widgets
│   │   │       ├── video_user_info.dart (1 KB)     // 视频画面item上的用户信息
│   │   │       └── volume_bar.dart (<1 KB)         // 用户信息中的麦克风图标
│   │   ├── video_layout
│   │   │   ├── view.dart (2 KB)                // 单页视频画面中视频item的布局方式
│   │   │   └── widgets
│   │   │       └── with_draggable_window_widget.dart (1 KB)// 两人视频画面大小窗布局方式
│   │   └── video_page_turning
│   │       ├── view.dart (2 KB)                // 视频画面翻页管理页面
│   │       └── widgets
│   │           └── page_indicator.dart (1 KB)      // 页面指示器小圆点
```

方案三：自定义全部 UI 方案

`TUIRoomKit` 的整体功能是基于 `TUIRoomEngine` 这个无 UI SDK 实现的，您可以完全基于 `TUIRoomEngine` 实现一套自己的 UI 界面。详情可见 TUIRoomEngine API 接口地址 。

API 参考 (TUIRoomKit)
iOS&Mac
API 概览

最近更新时间：2023-09-28 15:21:01

API OVERVIEW

创建实例和事件回调

函数列表	描述
loginWithSDKAppId:userId:userSig:onSuccess:onError:	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout:onError:	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfoWithUserName:avatarUrl:onSuccess:onError:	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
setSelfInfo:onSuccess:onError:	设置本地用户信息
addObserver:	设置事件回调
removeObserver:	移除事件回调

房间相关主动接口

函数列表	描述
createRoom:onSuccess:onError:	创建房间
destroyRoom:onError:	解散房间
enterRoom:onSuccess:onError:	进入房间
exitRoom:onSuccess:onError:	离开房间
connectOtherRoom:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	连接其他房间
disconnectOtherRoom:onError:	断开与其他房间的连接
fetchRoomInfo:onError:	获取房间信息
updateRoomNameByAdmin:onSuccess:onError:	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin:onSuccess:onError:	设置房间麦控模式（只有管理员或房主能够调用）

本地用户视图渲染、视频管理

函数列表	描述
setLocalVideoView:view:	设置本地用户视频渲染的视图控件
openLocalCamera:quality:onSuccess:onError:	打开本地摄像头
closeLocalCamera	关闭本地摄像头
startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality:	更新本地视频编码质量设置
updateVideoQualityEx:params:	设置视频编码器的编码参数
setVideoResolutionMode:resolutionMode:	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
enableGravitySensor:	开启重力感应模式
startScreenCaptureByReplaykit:	开始屏幕分享（该接口仅支持移动端）
startScreenCapture:onSuccess:onError:	开始屏幕分享（该接口仅支持桌面端 Mac OS 系统）
stopScreenCapture	结束屏幕分享
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）
selectScreenCaptureTarget:	选取要分享的屏幕或窗口（该接口仅支持 Mac OS 系统）

本地用户音频管理

函数列表	描述
openLocalMicrophone:onSuccess:onError:	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio:	暂停发布本地的音频流
unmuteLocalAudio:onError:	恢复发布本地的音频流

远端用户视图渲染、视频管理

函数列表	描述
setRemoteVideoView:streamType:view:	设置远端用户视频渲染的视图控件
startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:	开始播放远端用户视频
stopPlayRemoteVideo:streamType:	停止播放远端用户视频
muteRemoteAudioStream:isMute:	将远端用户禁音

房间内用户信息

函数列表	描述
getUserList:onSuccess:onError:	获取房间内的成员列表
getUserInfo:onSuccess:onError:	获取成员信息

房间内用户管理

函数列表	描述
changeUserRole:role:onSuccess:onError:	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom:onSuccess:onError:	将远端用户踢出房间（只有管理员或房主能够调用）
addCategoryTagForUsers:userList:onSuccess:onError:	为用户增加标记（只有房主能够调用）
removeCategoryTagForUsers:userList:onSuccess:onError:	为用户移除标记（只有房主能够调用）
getUserListByTag:nextSequence:onSuccess:onError:	根据标记获取房间内用户信息

房间内用户发言管理

函数列表	描述
disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:	控制当前房间内所有用户是否可打开音频流、视频流采集设备的权限状态，例如：全员禁止打开麦克风、全员禁止打开摄像头、全员禁止打开屏幕分享（目前仅会议场景下可用，并且只有管理员或房主能够调用）
openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin:device:onSuccess:onError:	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	请求打开本地媒体设备（普通用户可用）

房间内麦位管理

函数列表	描述
setMaxSeatCount:onSuccess:onError:	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList:onError:	获取麦位列表
lockSeatByAdmin:lockMode:onSuccess:onError:	锁定麦位（只有管理员或房主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat:timeout:onAccepted:onRejected:onCan	上麦（上麦发言模式下，需要申请）

celled:onTimeout:onError:	
leaveSeat:onError:	下麦
takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin:userId:onSuccess:onError:	主持人/管理员 将用户踢下麦

文本消息

函数列表	描述
sendTextMessage:onSuccess:onError:	发送本文消息
sendCustomMessage:onSuccess:onError:	发送自定义消息
disableSendingMessageByAdmin:isDisable:onSuccess:onError:	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser:onSuccess:onError:	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

信令管理

函数列表	描述
cancelRequest:onSuccess:onError:	取消请求
responseRemoteRequest:agree:onSuccess:onError:	回复请求

高级功能

函数列表	描述
switchCamera:	切换前置或后置摄像头（仅适用于移动端）
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）
setBeautyLevel:beautyLevel:	设置美颜级别
setWhitenessLevel:	设置美白级别

调试相关

函数列表	描述
callExperimentalAPI:	调用实验性接口

错误事件回调

函数列表	描述
onError:message:	错误事件回调

登录状态事件回调

函数列表	描述
onKickedOffLine:	用户被踢下线
onUserSigExpired	用户凭证超时事件

房间内事件回调

函数列表	描述
onRoomNameChanged:roomName:	房间名称更改事件
onAllUserMicrophoneDisableChanged:isDisable:	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged:isDisable:	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged:isDisable:	房间内用户发送文本消息被禁用事件
onRoomDismissed:	房间被解散事件
onKickedOutOfRoom:reason:message:	被踢出房间事件
onRoomSpeechModeChanged:speechMode:	房间麦控模式发生变化

房间内用户事件回调

函数列表	描述
onRemoteUserEnterRoom:userInfo:	远端用户进房事件
onRemoteUserLeaveRoom:userInfo:	远端用户离开房间事件
onUserRoleChanged:userRole:	用户角色发生变化事件
onUserVideoStateChanged:streamType:hasVideo:reason:	用户视频状态发生变化事件
onUserAudioStateChanged:hasAudio:reason:	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged:userId:isDisable:	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged:	用户网络状态变化事件
onUserScreenCaptureStopped:	屏幕分享结束

版权所有：腾讯云计算（北京）有限责任公司

房间内麦位事件回调

函数列表	描述
onRoomMaxSeatCountChanged:maxSeatNumber:	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged:seated:left:	麦位列表发生变化事件
onKickedOffSeat:	收到用户被踢下麦事件

请求信令事件回调

函数列表	描述
onRequestReceived:	收到请求消息事件
onRequestCancelled:userId:	收到请求被取消的事件

房间内消息事件回调

函数列表	描述
onReceiveTextMessage:message:	收到普通文本消息事件
onReceiveCustomMessage:message:	收到自定义消息事件
onDeviceChanged:type:state:	本地设备添加事件

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomKit

最近更新时间：2023-10-30 15:46:02

简介

TUIRoomKit 是多人视频会议 SDK 提供的上层 UI 开源套件，目前 iOS 平台仅支持 Swift 语言，通过简单 API 调用即可唤起会议 UI。

TUIRoomKit 接口

API	描述
createInstance	初始化 TUIRoomKit 单例对象
destroyInstance	销毁 TUIRoomKit 单例对象
setSelfInfo	设置用户信息（头像、昵称）（可选）
createRoom	创建房间
enterRoom	进入房间

createInstance

初始化 TUIRoomKit 单例对象。

```
public class func createInstance() -> TUIRoomKit
```

destroyInstance

销毁 TUIRoomKit 单例对象。

```
public class func destroyInstance() -> Void
```

setSelfInfo（可选）

设置用户信息（头像、昵称）。

```
public func setSelfInfo(userName: String,
                        avatarURL: String,
                        onSuccess: @escaping TUISuccessBlock,
                          onError: @escaping TUIErrorBlock) -> Void
```

参数	类型	含义
userName	String	用户名
avatarURL	String	用户头像链接
onSuccess	TUISuccessBlock	成功回调
onError	TUIErrorBlock	失败回调

createRoom

版权所有：腾讯云计算（北京）有限责任公司

创建房间。

```
public func createRoom(roomInfo: TUIRoomInfo,
                       onSuccess: @escaping TUISuccessBlock,
                         onError: @escaping TUIErrorBlock) -> Void
```

参数含义如下表所示：

参数	类型	含义
roomInfo	TUIRoomInfo	房间基本信息
onSuccess	TUISuccessBlock	成功回调
onError	TUIErrorBlock	失败回调

enterRoom

进入房间。

```
public func enterRoom(roomId: String,
                  enableAudio: Bool,
                  enableVideo: Bool,
              isSoundOnSpeaker: Bool,
                    onSuccess: @escaping TUISuccessBlock,
                      onError: @escaping TUIErrorBlock) -> Void
```

参数含义如下表所示：

参数	类型	含义
roomId	String	房间字符串
enableAudio	Bool	进入房间前设置是否开启音频
enableVideo	Bool	进入房间前设置是否开启视频
isSoundOnSpeaker	Bool	进入房间前设置是否开启扬声器
onSuccess	TUISuccessBlock	成功回调
onError	TUIErrorBlock	失败回调

TUIRoomEngine

最近更新时间：2023-09-28 15:21:01

Copyright (c) 2021 Tencent. All rights reserved.

Module: TUIRoomEngine @ TUIKitEngine
Function: TUIRoomEngine 主功能接口
Version: 1.5

TUIRoomEngine

TUIRoomEngine

函数列表	描述
loginWithSDKAppId:userId:userSig:onSuccess:onError:	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout:onError:	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfoWithUserName:avatarUrl:onSuccess:onError:	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
setSelfInfo:onSuccess:onError:	设置本地用户信息
addObserver:	设置事件回调
removeObserver:	移除事件回调
createRoom:onSuccess:onError:	创建房间
destroyRoom:onError:	解散房间
enterRoom:onSuccess:onError:	进入房间
exitRoom:onSuccess:onError:	离开房间
connectOtherRoom:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	连接其他房间
disconnectOtherRoom:onError:	断开与其他房间的连接
fetchRoomInfo:onError:	获取房间信息
updateRoomNameByAdmin:onSuccess:onError:	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin:onSuccess:onError:	设置房间麦控模式（只有管理员或房主能够调用）
setLocalVideoView:view:	设置本地用户视频渲染的视图控件
openLocalCamera:quality:onSuccess:onError:	打开本地摄像头

closeLocalCamera	关闭本地摄像头
startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality:	更新本地视频编码质量设置
updateVideoQualityEx:params:	设置视频编码器的编码参数
setVideoResolutionMode:resolutionMode:	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
enableGravitySensor:	开启重力感应模式
startScreenCaptureByReplaykit:	开始屏幕分享（该接口仅支持移动端）
startScreenCapture:onSuccess:onError:	开始屏幕分享（该接口仅支持桌面端 Mac OS 系统）
stopScreenCapture	结束屏幕分享
getScreenCaptureSources	枚举可分享的屏幕和窗口（该接口仅支持 Mac OS 系统）
selectScreenCaptureTarget:	选取要分享的屏幕或窗口（该接口仅支持 Mac OS 系统）
openLocalMicrophone:onSuccess:onError:	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio:	暂停发布本地的音频流
unmuteLocalAudio:	恢复发布本地的音频流
setRemoteVideoView:streamType:view:	设置远端用户视频渲染的视图控件
startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:	开始播放远端用户视频
stopPlayRemoteVideo:streamType:	停止播放远端用户视频
muteRemoteAudioStream:isMute:	将远端用户禁音
getUserList:onSuccess:onError:	获取房间内的成员列表
getUserInfo:onSuccess:onError:	获取成员信息
changeUserRole:role:onSuccess:onError:	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom:onSuccess:onError:	将远端用户踢出房间（只有管理员或房主能够调用）
addCategoryTagForUsers:userList:onSuccess:onError:	为用户增加标记（只有房主能够调用）
removeCategoryTagForUsers:userList:onSuccess:onError:	为用户移除标记（只有房主能够调用）
getUserListByTag:nextSequence:onSuccess:onError:	根据标记获取房间内用户信息
disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:	控制当前房间内所有用户是否可打开音频流、视频流采集设备的权限状态，例如：全员禁止打开麦克风、全员禁止打开摄像

版权所有：腾讯云计算（北京）有限责任公司

	头、全员禁止打开屏幕分享（目前仅会议场景下可用，并且只有管理员或房主能够调用）
openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin:device:onSuccess:onError:	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	请求打开本地媒体设备（普通用户可用）
setMaxSeatCount:onSuccess:onError:	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList:onError:	获取麦位列表
lockSeatByAdmin:lockMode:onSuccess:onError:	锁定麦位（只有管理员或房主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	上麦（上麦发言模式下，需要申请）
leaveSeat:onError:	下麦
takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin:userId:onSuccess:onError:	主持人/管理员 将用户踢下麦
sendTextMessage:onSuccess:onError:	发送本文消息
sendCustomMessage:onSuccess:onError:	发送自定义消息
disableSendingMessageByAdmin:isDisable:onSuccess:onError:	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser:onSuccess:onError:	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）
cancelRequest:onSuccess:onError:	取消请求
responseRemoteRequest:agree:onSuccess:onError:	回复请求
switchCamera:	切换前置或后置摄像头（仅适用于移动端）
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）
setBeautyLevel:beautyLevel:	设置美颜级别
setWhitenessLevel:	设置美白级别
callExperimentalAPI:	调用实验性接口

loginWithSDKAppId:userId:userSig:onSuccess:onError:

loginWithSDKAppId:userId:userSig:onSuccess:onError:

```
+ (void)loginWithSDKAppId:    (NSInteger)sdkAppId
                   userId:    (NSString *)userId
                  userSig:    (NSString *)userSig
                onSuccess:    (TUISuccessBlock)onSuccess
                  onError:    (TUIErrorBlock)onError
```

登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作

参数	描述
onError	失败回调
onSuccess	成功回调
sdkAppId	它是腾讯云用于区分客户的唯一标识，进入腾讯云实时音视频 控制台 创建应用，即可看到SDKAppId
userId	用户ID，用于区分不同用户
userSig	用户签名，用于腾讯云流量的认证

logout:onError:

logout:onError:

```
+ (void)logout:    (TUISuccessBlock)onSuccess
           onError:    (TUIErrorBlock)onError
```

退出登录接口，会有主动离开房间操作、销毁资源

参数	描述
onError	失败回调
onSuccess	成功回调

setSelfInfoWithUserName:avatarUrl:onSuccess:onError:

setSelfInfoWithUserName:avatarUrl:onSuccess:onError:

```
+ (void)setSelfInfoWithUserName:    (NSString *)userName
```

```
avatarUrl:      (NSString *)avatarURL
onSuccess:      (TUISuccessBlock)onSuccess
onError:        (TUIErrorBlock)onError
```

设置本地用户名称和头像

参数	描述
avatarURL	用户头像URL地址
onError	失败回调
onSuccess	成功回调
userName	用户名称

getSelfInfo

getSelfInfo

获取本地用户登录的基本信息

返回值说明：

TUILoginUserInfo 用户登录信息

setSelfInfo:onSuccess:onError:

setSelfInfo:onSuccess:onError:

```
+ (void)setSelfInfo:    (TUILoginUserInfo *)userInfo
onSuccess:              (TUISuccessBlock)onSuccess
onError:                (TUIErrorBlock)onError
```

设置本地用户信息

参数	描述
onError	失败回调
onSuccess	成功回调
userInfo	本地用户信息

addObserver:

addObserver:

```
- (void)addObserver:    (id<TUIRoomObserver>)observer
```

设置事件回调

您可以通过 TUIRoomObserver 获得各类事件通知（比如：错误码，远端用户进房，音视频状态参数等）

参数	描述
observer	监听的实例

removeObserver:

removeObserver:

```
- (void)removeObserver:    (id<TUIRoomObserver>)observer
```

移除事件回调

参数	描述
observer	待移除的监听回调实例

createRoom:onSuccess:onError:

createRoom:onSuccess:onError:

```
- (void)createRoom:    (TUIRoomInfo *)roomInfo
         onSuccess:    (TUISuccessBlock)onSuccess
           onError:    (TUIErrorBlock)onError
```

创建房间

参数	描述
onError	失败回调
onSuccess	成功回调
roomInfo	房间信息，可以初始化房间的一些设置

destroyRoom:onError:

destroyRoom:onError:

```
- (void)destroyRoom:    (TUISuccessBlock)onSuccess
            onError:    (TUIErrorBlock)onError
```

解散房间

参数	描述
onError	失败回调
onSuccess	成功回调

enterRoom:onSuccess:onError:

enterRoom:onSuccess:onError:

```
- (void)enterRoom:    (NSString *)roomId
          onSuccess:  (TUIRoomInfoBlock)onSuccess
            onError:  (TUIErrorBlock)onError
```

进入房间

参数	描述
onError	失败回调
onSuccess	成功回调
roomId	房间ID

exitRoom:onSuccess:onError:

exitRoom:onSuccess:onError:

```
- (void)exitRoom:    (BOOL)syncWaiting
          onSuccess:  (TUISuccessBlock)onSuccess
            onError:  (TUIErrorBlock)onError
```

离开房间

参数	描述
onError	失败回调

onSuccess	成功回调
syncWaiting	是否同步等待接口返回

connectOtherRoom:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

connectOtherRoom:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

```
- (TUIRequest *)connectOtherRoom:(NSString *)roomId
                          userId:(NSString *)userId
                         timeout:(NSTimeInterval)timeout
                      onAccepted:(TUIRequestAcceptedBlock)onAccepted
                      onRejected:(TUIRequestRejectedBlock)onRejected
                     onCancelled:(TUIRequestCancelledBlock)onCancelled
                       onTimeout:(TUIRequestTimeoutBlock)onTimeout
                         onError:(TUIRequestErrorBlock)onError
```

连接其他房间

参数	描述
onAccepted	邀请被接受的回调
onCancelled	邀请被取消的回调
onError	邀请发生错误的回调
onRejected	邀请被拒绝的回调
onTimeout	邀请超时未处理的回调
roomId	房间ID
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

注意
用于直播场景下的申请跨房连麦

返回值说明：

请求体

disconnectOtherRoom:onError:

disconnectOtherRoom:onError:

```
- (void)disconnectOtherRoom:    (TUISuccessBlock)onSuccess
                     onError:   (TUIErrorBlock)onError
```

断开与其他房间的连接

参数	描述
onError	失败回调
onSuccess	成功回调

> **注意**
> 用于直播场景下的断开跨房连麦

fetchRoomInfo:onError:

fetchRoomInfo:onError:

```
- (void)fetchRoomInfo:    (TUIRoomInfoBlock)onSuccess
              onError:    (TUIErrorBlock)onError
```

获取房间信息

参数	描述
onError	失败回调
onSuccess	成功获取到房间信息回调，回调会包含 TUIRoomInfo 房间信息

updateRoomNameByAdmin:onSuccess:onError:

updateRoomNameByAdmin:onSuccess:onError:

```
- (void)updateRoomNameByAdmin:    (NSString *)roomName
                     onSuccess:   (TUISuccessBlock)onSuccess
                     onError:     (TUIErrorBlock)onError
```

更新房间名称（只有管理员或房主能够调用）

参数	描述
onError	失败回调
onSuccess	成功回调
roomName	房间名称

updateRoomSpeechModeByAdmin:onSuccess:onError:

updateRoomSpeechModeByAdmin:onSuccess:onError:

```
- (void)updateRoomSpeechModeByAdmin:(TUISpeechMode)mode
                          onSuccess:(TUISuccessBlock)onSuccess
                            onError:(TUIErrorBlock)onError
```

设置房间麦控模式（只有管理员或房主能够调用）

参数	描述
mode	kFreeToSpeak: 自由发言模式, 用户可以自由开启麦克风和扬声器; kApplyToSpeak: 申请发言模式，用户requestOpenLocalMicrophone 或 requestOpenLocalCamera 向房主或管理员申请后，方可打开麦克风和摄像头开始发言 kkSpeakAfterTakingSeat: 上麦发言模式，KConference房间内，所有人在发言前，必须 takeSeat，才能进行麦克风和摄像头操作。
onError	错误回调
onSuccess	成功回调

setLocalVideoView:view:

setLocalVideoView:view:

```
- (void)setLocalVideoView:(TUIVideoStreamType)streamType
                     view:(TUIVideoView *__nullable)view
```

设置本地用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义
view	视频渲染视图

openLocalCamera:quality:onSuccess:onError:

openLocalCamera:quality:onSuccess:onError:

```
- (void)openLocalCamera:    (BOOL)isFront
                quality:    (TUIVideoQuality)quality
              onSuccess:    (TUISuccessBlock)onSuccess
                onError:    (TUIErrorBlock)onError
```

打开本地摄像头

参数	描述
isFront	YES: 前置 NO: 后置
onError	失败回调
onSuccess	成功回调

closeLocalCamera

closeLocalCamera

关闭本地摄像头

startPushLocalVideo

startPushLocalVideo

开始推送本地视频。默认开启

stopPushLocalVideo

stopPushLocalVideo

停止推送本地视频

updateVideoQuality:

updateVideoQuality:

```
- (void)updateVideoQuality:(TUIVideoQuality)quality
```

更新本地视频编码质量设置

updateVideoQualityEx:params:

updateVideoQualityEx:params:

```
- (void)updateVideoQualityEx:(TUIVideoStreamType)streamType
                      params:(TUIVideoEncoderParams *)params
```

设置视频编码器的编码参数

参数	描述
params	视频编码器的编码参数，定义可参考 TUIVideoEncoderParams 的定义
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义

setVideoResolutionMode:resolutionMode:

setVideoResolutionMode:resolutionMode:

```
- (void)setVideoResolutionMode:(TUIVideoStreamType)streamType
                  resolutionMode:(TUIResolutionMode)resolutionMode
```

设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）

参数	描述
resolutionMode	分辨率模式，定义可参考 TUIResolutionMode 的定义
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义

enableGravitySensor:

enableGravitySensor:

```
- (void)enableGravitySensor:(BOOL)enable
```

开启重力感应模式

参数	描述

enable	YES: 开启 NO: 关闭

注意
开启该模式后，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

startScreenCaptureByReplaykit:

startScreenCaptureByReplaykit:

```
- (void)startScreenCaptureByReplaykit:    (NSString *)appGroup
```

开始屏幕分享（该接口仅支持移动端）

startScreenCapture:onSuccess:onError:

startScreenCapture:onSuccess:onError:

```
- (void)startScreenCapture:    (TUIVideoView *)view
              onSuccess:    (TUISuccessBlock)onSuccess
                onError:    (TUIErrorBlock)onError
```

开始屏幕分享（该接口仅支持桌面端 Mac OS 系统）

参数	描述
onError	错误回调
onSuccess	成功回调
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
该接口可以抓取整个 Mac OS 系统的屏幕内容，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。

stopScreenCapture

stopScreenCapture

结束屏幕分享

getScreenCaptureSources

getScreenCaptureSources

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

> **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:

selectScreenCaptureTarget:

```
- (void)selectScreenCaptureTarget:    (NSString *)targetId
```

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
targetId	指定分享源

openLocalMicrophone:onSuccess:onError:

openLocalMicrophone:onSuccess:onError:

```
- (void)openLocalMicrophone:    (TUIAudioQuality)quality
                  onSuccess:    (TUISuccessBlock)onSuccess
                    onError:    (TUIErrorBlock)onError
```

打开本地麦克风

参数	描述
onError	失败回调
onSuccess	成功回调
quality	音频质量

closeLocalMicrophone

closeLocalMicrophone

关闭本地麦克风

muteLocalAudio:

muteLocalAudio

暂停发布本地的音频流

unmuteLocalAudio:onError:

unmuteLocalAudio:onError:

```
- (void)unmuteLocalAudio:    (TUISuccessBlock)onSuccess
                  onError:   (TUIErrorBlock)onError
```

恢复发布本地的音频流

参数	描述
onError	失败回调
onSuccess	成功回调

setRemoteVideoView:streamType:view:

setRemoteVideoView:streamType:view:

```
- (void)setRemoteVideoView:    (NSString *)userId
                 streamType:   (TUIVideoStreamType)streamType
                       view:   (TUIVideoView *__nullable)view
```

设置远端用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义

userId	远端用户ID
view	视频渲染视图

startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:

startPlayRemoteVideo:streamType:onPlaying:onLoading:onError:

```
- (void)startPlayRemoteVideo:(NSString *)userId
                  streamType:(TUIVideoStreamType)streamType
                   onPlaying:(TUIPlayOnPlayingBlock)onPlaying
                   onLoading:(TUIPlayOnLoadingBlock)onLoading
                     onError:(TUIPlayOnErrorBlock)onError
```

开始播放远端用户视频

参数	描述
onError	错误回调
onLoading	加载回调
onPlaying	播放回调
streamType	视频流的类型 详细定义可以参考 TUIVideoStreamType 的定义
userId	用户ID

stopPlayRemoteVideo:streamType:

stopPlayRemoteVideo:streamType:

```
- (void)stopPlayRemoteVideo:(NSString *)userId
                 streamType:(TUIVideoStreamType)streamType
```

停止播放远端用户视频

参数	描述
streamType	视频流的类型 详细定义可以参考 TUIVideoStreamType 的定义
userId	用户ID

muteRemoteAudioStream:isMute:

muteRemoteAudioStream:isMute:

```
- (void)muteRemoteAudioStream:  (NSString *)userId
                       isMute:  (BOOL)isMute
```

将远端用户禁音

参数	描述
isMute	是否禁音
userId	用户ID

getUserList:onSuccess:onError:

getUserList:onSuccess:onError:

```
- (void)getUserList:  (NSInteger)nextSequence
          onSuccess:  (TUIUserListResponseBlock)onSuccess
            onError:  (TUIErrorBlock)onError
```

获取房间内的成员列表

参数	描述
nextSequence	分页拉取标志，第一次拉取填0，回调成功 如果callback返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0
onError	错误回调
onSuccess	成功回调，回调中包含 TUIUserInfo 数组

getUserInfo:onSuccess:onError:

getUserInfo:onSuccess:onError:

```
- (void)getUserInfo:  (NSString *)userId
          onSuccess:  (TUIUserInfoBlock)onSuccess
            onError:  (TUIErrorBlock)onError
```

获取成员信息

参数	描述

onError	错误回调
onSuccess	成功回调，回调中包含 TUIUserInfo 数组
userId	用户ID

changeUserRole:role:onSuccess:onError:

changeUserRole:role:onSuccess:onError:

```
- (void)changeUserRole:(NSString *)userId
                  role:(TUIRole)role
             onSuccess:(TUISuccessBlock)onSuccess
               onError:(TUIErrorBlock)onError
```

修改用户角色（只有管理员或房主能够调用）

参数	描述
onError	错误回调
onSuccess	成功回调
role	用户角色 详细定义可以参考 TUIRole 的定义
userId	用户ID

kickRemoteUserOutOfRoom:onSuccess:onError:

kickRemoteUserOutOfRoom:onSuccess:onError:

```
- (void)kickRemoteUserOutOfRoom:(NSString *)userId
                      onSuccess:(TUISuccessBlock)onSuccess
                        onError:(TUIErrorBlock)onError
```

将远端用户踢出房间（只有管理员或房主能够调用）

参数	描述
onError	错误回调
onSuccess	成功回调
userId	用户ID

addCategoryTagForUsers:userList:onSuccess:onError:

addCategoryTagForUsers:userList:onSuccess:onError:

```
- (void)addCategoryTagForUsers:    (NSInteger)tag
                      userList:    (NSArray<NSString *> *)userList
                     onSuccess:    (TUISuccessBlock)onSuccess
                       onError:    (TUIErrorBlock)onError
```

为用户增加标记（只有房主能够调用）

参数	描述
onError	错误回调
onSuccess	成功回调
tag	标记类型。数字类型，大于等于 1000，您可以自定义。
userList	用户列表

removeCategoryTagForUsers:userList:onSuccess:onError:

removeCategoryTagForUsers:userList:onSuccess:onError:

```
- (void)removeCategoryTagForUsers:    (NSInteger)tag
                         userList:    (NSArray<NSString *> *)userList
                        onSuccess:    (TUISuccessBlock)onSuccess
                          onError:    (TUIErrorBlock)onError
```

为用户移除标记（只有房主能够调用）

参数	描述
onError	错误回调
onSuccess	成功回调
tag	标记类型。数字类型，大于等于 1000，您可以自定义。
userList	用户列表

getUserListByTag:nextSequence:onSuccess:onError:

getUserListByTag:nextSequence:onSuccess:onError:

```
- (void)getUserListByTag:(NSInteger)tag
           nextSequence:(NSInteger)nextSequence
              onSuccess:(TUIUserListResponseBlock)onSuccess
                onError:(TUIErrorBlock)onError
```

根据标记获取房间内用户信息

参数	描述
nextSequence	分页拉取标志，第一次拉取填0，回调成功 如果callback返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0
onError	错误回调
onSuccess	成功回调
tag	标记类型。数字类型，大于等于 1000，您可以自定义。

disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:

disableDeviceForAllUserByAdmin:isDisable:onSuccess:onError:

```
- (void)disableDeviceForAllUserByAdmin:(TUIMediaDevice)device
                             isDisable:(BOOL)isDisable
                             onSuccess:(TUISuccessBlock)onSuccess
                               onError:(TUIErrorBlock)onError
```

控制当前房间内所有用户是否可打开音频流、视频流采集设备的权限状态，例如：全员禁止打开麦克风、全员禁止打开摄像头、全员禁止打开屏幕分享（目前仅会议场景下可用，并且只有管理员或房主能够调用）

参数	描述
device	设备. 详细定义参考: TUIMediaDevice
isDisable	否禁用
onError	操作失败回调
onSuccess	操作成功回调

openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

openRemoteDeviceByAdmin:device:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

```
- (TUIRequest *)openRemoteDeviceByAdmin:(NSString *)userId
                                 device:(TUIMediaDevice)device
                                timeout:(NSTimeInterval)timeout
                             onAccepted:(nullable TUIRequestAcceptedBlock)onAccepted
                             onRejected:(nullable TUIRequestRejectedBlock)onRejected
                            onCancelled:(nullable TUIRequestCancelledBlock)onCancelled
                              onTimeout:(nullable TUIRequestTimeoutBlock)onTimeout
                                onError:(nullable TUIRequestErrorBlock)onError
```

请求远端用户打开媒体设备（只有管理员或房主能够调用）

参数	描述
device	媒体设备。详细定义参考: TUIMediaDevice
onAccepted	邀请被接受的回调
onCancelled	邀请被取消的回调
onError	邀请发生错误的回调
onRejected	邀请被拒绝的回调
onTimeout	邀请超时未处理的回调
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

closeRemoteDeviceByAdmin:device:onSuccess:onError:

closeRemoteDeviceByAdmin:device:onSuccess:onError:

```
- (void)closeRemoteDeviceByAdmin:(NSString *)userId
                          device:(TUIMediaDevice)device
                       onSuccess:(TUISuccessBlock)onSuccess
                         onError:(TUIErrorBlock)onError
```

关闭远端用户媒体设备（只有管理员或房主能够调用）

参数	描述
device	媒体设备。详细定义参考: TUIMediaDevice
onError	调用失败回调
onSuccess	调用成功回调
userId	用户ID

applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

applyToAdminToOpenLocalDevice:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

```
- (TUIRequest *)applyToAdminToOpenLocalDevice:(TUIMediaDevice)device
                                      timeout:(NSTimeInterval)timeout
                                   onAccepted:(nullable TUIRequestAcceptedBlock)onAccepted
                                   onRejected:(nullable TUIRequestRejectedBlock)onRejected
                                  onCancelled:(nullable TUIRequestCancelledBlock)onCancelled
                                    onTimeout:(nullable TUIRequestTimeoutBlock)onTimeout
                                      onError:(nullable TUIRequestErrorBlock)onError
```

请求打开本地媒体设备（普通用户可用）

参数	描述
device	用户ID device 媒体设备。详细定义参考: TUIMediaDevice
onAccepted	邀请被接受的回调
onCancelled	邀请被取消的回调
onError	邀请发生错误的回调
onRejected	邀请被拒绝的回调
onTimeout	邀请超时未处理的回调
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

返回值说明：

TUIRequest 请求体

setMaxSeatCount:onSuccess:onError:

setMaxSeatCount:onSuccess:onError:

```
- (void)setMaxSeatCount:    (NSUInteger)maxSeatCount
              onSuccess:    (TUISuccessBlock)onSuccess
                onError:    (TUIErrorBlock)onError
```

设置最大麦位数（仅支持进房前和创建房间时设置）

参数	描述
maxSeatCount	最大麦位数
onError	错误回调
onSuccess	成功回调

getSeatList:onError:

getSeatList:onError:

```
- (void)getSeatList:    (TUISeatListResponseBlock)onSuccess
                onError:    (TUIErrorBlock)onError
```

获取麦位列表

参数	描述
onError	错误回调
onSuccess	成功回调

lockSeatByAdmin:lockMode:onSuccess:onError:

lockSeatByAdmin:lockMode:onSuccess:onError:

```
- (void)lockSeatByAdmin:    (NSInteger)seatIndex
                lockMode:    (TUISeatLockParams *)lockParams
              onSuccess:    (TUISuccessBlock)onSuccess
                onError:    (TUIErrorBlock)onError
```

锁定麦位（只有管理员或房主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）

参数	描述
lockParams	锁麦参数。详情参考:$TUISeatLockParam$
onError	错误回调
onSuccess	成功回调
seatIndex	麦位编号

takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

takeSeat:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

```
- (TUIRequest *)takeSeat:(NSInteger)seatIndex
               timeout:(NSTimeInterval)timeout
            onAccepted:(TUIRequestAcceptedBlock)onAccepted
            onRejected:(TUIRequestRejectedBlock)onRejected
           onCancelled:(TUIRequestCancelledBlock)onCancelled
             onTimeout:(TUIRequestTimeoutBlock)onTimeout
               onError:(TUIRequestErrorBlock)onError
```

上麦（上麦发言模式下，需要申请）

参数	描述
onAccepted	邀请被接受的回调
onCancelled	邀请被取消的回调
onError	邀请发生错误的回调
onRejected	邀请被拒绝的回调
onTimeout	邀请超时未处理的回调
seatIndex	麦位编号。会议场景下无需关心该参数，填0即可。
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

注意

开启上麦发言模式时，需要向主持人或管理员发起申请才允许上麦。

开启自由发言模式，直播场景可以自由上麦，上麦后开麦发言，会议场景无需调用该接口，即可开麦发言。

返回值说明：

TUIRequest 请求体

leaveSeat:onError:

leaveSeat:onError:

```
- (void)leaveSeat:    (TUISuccessBlock)onSuccess
          onError:    (TUIErrorBlock)onError
```

下麦

参数	描述
onError	错误回调
onSuccess	成功回调

takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

takeUserOnSeatByAdmin:userId:timeout:onAccepted:onRejected:onCancelled:onTimeout:onError:

```
- (TUIRequest *)takeUserOnSeatByAdmin:    (NSInteger)seatIndex
                               userId:    (NSString *)userId
                              timeout:    (NSTimeInterval)timeout
                           onAccepted:    (TUIRequestAcceptedBlock)onAccepted
                           onRejected:    (TUIRequestRejectedBlock)onRejected
                          onCancelled:    (TUIRequestCancelledBlock)onCancelled
                            onTimeout:    (TUIRequestTimeoutBlock)onTimeout
                              onError:    (TUIRequestErrorBlock)onError
```

主持人/管理员 邀请用户上麦

参数	描述
onAccepted	邀请被接受的回调
onCancelled	邀请被取消的回调
onError	邀请发生错误的回调
onRejected	邀请被拒绝的回调
onTimeout	邀请超时未处理的回调
seatIndex	麦位编号。会议场景无需关心，填0即可。

timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

kickUserOffSeatByAdmin:userId:onSuccess:onError:

kickUserOffSeatByAdmin:userId:onSuccess:onError:

```
- (void)kickUserOffSeatByAdmin:(NSInteger)seatIndex
                        userId:(NSString *)userId
                     onSuccess:(TUISuccessBlock)onSuccess
                       onError:(TUIErrorBlock)onError
```

主持人/管理员 将用户踢下麦

参数	描述
onError	失败回调
onSuccess	成功回调
seatIndex	麦位编号。会议场景无需关心，填0即可。
userId	用户ID

sendTextMessage:onSuccess:onError:

sendTextMessage:onSuccess:onError:

```
- (void)sendTextMessage:(NSString *)message
              onSuccess:(TUISuccessBlock)onSuccess
                onError:(TUIErrorBlock)onError
```

发送本文消息

参数	描述
message	消息内容
onError	错误回调
onSuccess	成功回调

sendCustomMessage:onSuccess:onError:

sendCustomMessage:onSuccess:onError:

```
- (void)sendCustomMessage:    (NSString *)message
                onSuccess:    (TUISuccessBlock)onSuccess
                  onError:    (TUIErrorBlock)onError
```

发送自定义消息

参数	描述
message	消息内容
onError	错误回调
onSuccess	成功回调

disableSendingMessageByAdmin:isDisable:onSuccess:onError:

disableSendingMessageByAdmin:isDisable:onSuccess:onError:

```
- (void)disableSendingMessageByAdmin:    (NSString *)userId
                           isDisable:    (BOOL)isDisable
                           onSuccess:    (TUISuccessBlock)onSuccess
                             onError:    (TUIErrorBlock)onError
```

禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
isDisable	是否禁用
onError	错误回调
onSuccess	成功回调
userId	用户ID

disableSendingMessageForAllUser:onSuccess:onError:

disableSendingMessageForAllUser:onSuccess:onError:

```
- (void)disableSendingMessageForAllUser:    (BOOL)isDisable
```

```
onSuccess:    (TUISuccessBlock)onSuccess
onError:      (TUIErrorBlock)onError
```

禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
isDisable	是否禁用
onError	错误回调
onSuccess	成功回调

cancelRequest:onSuccess:onError:

cancelRequest:onSuccess:onError:

```
- (void)cancelRequest:    (NSString *)requestId
onSuccess:               (TUISuccessBlock)onSuccess
onError:                 (TUIErrorBlock)onError
```

取消请求

参数	描述
onError	失败回调
onSuccess	成功回调
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
可以使用此接口来取消已发出的请求

responseRemoteRequest:agree:onSuccess:onError:

responseRemoteRequest:agree:onSuccess:onError:

```
- (void)responseRemoteRequest:    (NSString *)requestId
agree:                           (BOOL)agree
onSuccess:                       (TUISuccessBlock)onSuccess
onError:                         (TUIErrorBlock)onError
```

回复请求

参数	描述
agree	是否同意 YES: 同意请求, NO: 拒绝请求
onError	失败回调
onSuccess	成功回调
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
在收到信令请求时，可以使用此接口来回复接收到的请求

switchCamera:

switchCamera:

```
- (NSInteger)switchCamera:      (BOOL)frontCamera
```

切换前置或后置摄像头（仅适用于移动端）

返回值说明：

0：操作成功；负数：操作失败

getDevicesList:

getDevicesList:

```
- (NSArray<TXMediaDeviceInfo *> * _Nullable)getDevicesList:      (TUIMediaDeviceType)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:(TUIMediaDeviceType)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。

setBeautyLevel:beautyLevel:

setBeautyLevel:beautyLevel:

```
- (void)setBeautyLevel:(NSInteger)beautyStyle
            beautyLevel:(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。
beautyStyle	美颜风格，TXBeautyStyleSmooth：光滑；TXBeautyStyleNature：自然；TXBeautyStylePitu：优图。

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。

callExperimentalAPI:

callExperimentalAPI:

```
- (void)callExperimentalAPI:(NSString *)jsonStr
```

调用实验性接口

参数	描述
jsonStr	接口信息

TUIRoomObserver

最近更新时间：2023-09-14 16:46:12

Copyright (c) 2021 Tencent. All rights reserved.
Module: TUIRoomObserver @ TUIKitEngine
Function: TUIRoomEngine的事件回调接口

TUIRoomObserver

TUIRoomObserver

函数列表	描述
onError:message:	错误事件回调
onKickedOffLine:	用户被踢下线
onUserSigExpired	用户凭证超时事件
onRoomNameChanged:roomName:	房间名称更改事件
onAllUserMicrophoneDisableChanged:isDisable:	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged:isDisable:	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged:isDisable:	房间内用户发送文本消息被禁用事件
onRoomDismissed:	房间被解散事件
onKickedOutOfRoom:reason:message:	被踢出房间事件
onRoomSpeechModeChanged:speechMode:	房间麦控模式发生变化
onRemoteUserEnterRoom:userInfo:	远端用户进房事件
onRemoteUserLeaveRoom:userInfo:	远端用户离开房间事件
onUserRoleChanged:userRole:	用户角色发生变化事件
onUserVideoStateChanged:streamType:hasVideo:reason:	用户视频状态发生变化事件
onUserAudioStateChanged:hasAudio:reason:	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged:userId:isDisable:	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged:	用户网络状态变化事件
onUserScreenCaptureStopped:	屏幕分享结束

onRoomMaxSeatCountChanged:maxSeatNumber:	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged:seated:left:	麦位列表发生变化事件
onKickedOffSeat:	收到用户被踢下麦事件
onRequestReceived:	收到请求消息事件
onRequestCancelled:userId:	收到请求被取消的事件
onReceiveTextMessage:message:	收到普通文本消息事件
onReceiveCustomMessage:message:	收到自定义消息事件
onDeviceChanged:type:state:	本地设备添加事件

onError:message:

onError:message:

```
- (void)onError:    (TUIError)errorCode
        message:    (NSString *)message
```

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。

参数	描述
errorCode	错误码，请参考：TUIError
message	错误信息

onKickedOffLine:

onKickedOffLine:

```
- (void)onKickedOffLine:    (NSString *)message
```

用户被踢下线

参数	描述
message	被踢下线的描述

onUserSigExpired

onUserSigExpired

用户凭证超时事件

onRoomNameChanged:roomName:

onRoomNameChanged:roomName:

```
- (void)onRoomNameChanged:    (NSString *)roomId
                  roomName:   (NSString *)roomName
```

房间名称更改事件

参数	描述
roomId	房间ID
roomName	房间名称

onAllUserMicrophoneDisableChanged:isDisable:

onAllUserMicrophoneDisableChanged:isDisable:

```
- (void)onAllUserMicrophoneDisableChanged:    (NSString *)roomId
                                  isDisable:  (BOOL)isDisable
```

房间内所有用户麦克风被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onAllUserCameraDisableChanged:isDisable:

onAllUserCameraDisableChanged:isDisable:

```
- (void)onAllUserCameraDisableChanged:    (NSString *)roomId
                                isDisable:  (BOOL)isDisable
```

房间内所有用户摄像头被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onSendMessageForAllUserDisableChanged:isDisable:

onSendMessageForAllUserDisableChanged:isDisable:

```
- (void)onSendMessageForAllUserDisableChanged:    (NSString *)roomId
                                    isDisable:    (BOOL)isDisable
```

房间内用户发送文本消息被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onRoomDismissed:

onRoomDismissed:

```
- (void)onRoomDismissed:    (NSString *)roomId
```

房间被解散事件

参数	描述
roomId	房间ID

onKickedOutOfRoom:reason:message:

onKickedOutOfRoom:reason:message:

```
- (void)onKickedOutOfRoom:    (NSString *)roomId
                   reason:    (TUIKickedOutOfRoomReason)reason
                  message:    (NSString *)message
```

被踢出房间事件

参数	描述

message	被踢出的描述
reason	被踢出原因
roomId	房间ID

onRoomSpeechModeChanged:speechMode:

onRoomSpeechModeChanged:speechMode:

```
- (void)onRoomSpeechModeChanged:    (NSString *)roomId
                     speechMode:    (TUISpeechMode)mode
```

房间麦控模式发生变化

参数	描述
mode	房间模式
roomId	房间ID

onRemoteUserEnterRoom:userInfo:

onRemoteUserEnterRoom:userInfo:

```
- (void)onRemoteUserEnterRoom:    (NSString *)roomId
                     userInfo:    (TUIUserInfo *)userInfo
```

远端用户进房事件

参数	描述
roomId	房间ID
userInfo	用户信息

onRemoteUserLeaveRoom:userInfo:

onRemoteUserLeaveRoom:userInfo:

```
- (void)onRemoteUserLeaveRoom:    (NSString *)roomId
                     userInfo:    (TUIUserInfo *)userInfo
```

远端用户离开房间事件

参数	描述
roomId	房间ID
userInfo	用户信息

onUserRoleChanged:userRole:

onUserRoleChanged:userRole:

```
- (void)onUserRoleChanged:    (NSString *)userId
                  userRole:   (TUIRole)userRole
```

用户角色发生变化事件

参数	描述
userId	用户ID
userRole	用户角色 可参考 TUIRole 枚举定义

onUserVideoStateChanged:streamType:hasVideo:reason:

onUserVideoStateChanged:streamType:hasVideo:reason:

```
- (void)onUserVideoStateChanged:    (NSString *)userId
                      streamType:   (TUIVideoStreamType)streamType
                        hasVideo:   (BOOL)hasVideo
                          reason:   (TUIChangeReason)reason
```

用户视频状态发生变化事件

参数	描述
hasVideo	是否有视频流
reason	视频流发生变化原因 TUIChangeReasonBySelf：自己切换 TUIChangeReasonByAdmin：被管理员切换
streamType	视频流类型
userId	用户ID

onUserAudioStateChanged:hasAudio:reason:

onUserAudioStateChanged:hasAudio:reason:

```
- (void)onUserAudioStateChanged:    (NSString *)userId
                        hasAudio:   (BOOL)hasAudio
                          reason:   (TUIChangeReason)reason
```

用户音频状态发生变化事件

参数	描述
hasAudio	是否有音频流
reason	视频流发生变化原因 TUIChangeReasonBySelf: 自己切换 TUIChangeReasonByAdmin: 被管理员切换
userId	用户ID

onUserVoiceVolumeChanged

onUserVoiceVolumeChanged

用户音量变化事件

参数	描述
volumeMap	用户音量字典 key: userId, value: 用于承载所有正在说话的用户的音量大小，取值范围 0 - 100。

onSendMessageForUserDisableChanged:userId:isDisable:

onSendMessageForUserDisableChanged:userId:isDisable:

```
- (void)onSendMessageForUserDisableChanged:    (NSString *)roomId
                                    userId:   (NSString *)userId
                                 isDisable:   (BOOL)muted
```

用户文本消息发送能力发生变化事件

参数	描述
isDisable	是否被禁止发送文本消息 YES: 用户被禁止发送消息 NO: 用户被解除禁止，可以发送消息
userId	用户ID

onUserNetworkQualityChanged:

onUserNetworkQualityChanged:

```
- (void)onUserNetworkQualityChanged:    (NSArray<TUINetworkInfo *> *)networkList
```

用户网络状态变化事件

参数	描述
networkList	用户网络状态数组，可参考 TUINetworkInfo 对象

onUserScreenCaptureStopped:

onUserScreenCaptureStopped:

```
- (void)onUserScreenCaptureStopped:    (NSInteger)reason
```

屏幕分享结束

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）

onRoomMaxSeatCountChanged:maxSeatNumber:

onRoomMaxSeatCountChanged:maxSeatNumber:

```
- (void)onRoomMaxSeatCountChanged:    (NSString *)roomId
                    maxSeatNumber:    (NSInteger)maxSeatNumber
```

房间内最大麦位数发生变化事件（仅在会议类型房间生效）

参数	描述
maxSeatCount	房间内最大麦位数量
roomId	房间ID

onSeatListChanged:seated:left:

onSeatListChanged:seated:left:

```
- (void)onSeatListChanged:    (NSArray<TUISeatInfo *> *)seatList
```

seated:	(NSArray<TUISeatInfo *> *)seatedList
left:	(NSArray<TUISeatInfo *> *)leftList

麦位列表发生变化事件

参数	描述
leftList	新下麦的用户列表
seatList	目前麦上最新的用户列表，包含新上麦的用户
seatedList	新上麦的用户列表

onKickedOffSeat:

onKickedOffSeat:

- (void)onKickedOffSeat:	(NSString *)userId

收到用户被踢下麦事件

参数	描述
userId	操作踢人的（主持人/管理员）用户id

onRequestReceived:

onRequestReceived:

- (void)onRequestReceived:	(TUIRequest *)request

收到请求消息事件

参数	描述
request	请求内容，可参考 TUIRequest 对象

onRequestCancelled:userId:

onRequestCancelled:userId:

- (void)onRequestCancelled:	(NSString *)requestId
userId:	(NSString *)userId

收到请求被取消的事件

参数	描述
requestId	请求ID
userId	取消信令的用户ID

onReceiveTextMessage:message:

onReceiveTextMessage:message:

```
- (void)onReceiveTextMessage:(NSString *)roomId
                     message:(TUIMessage *)message
```

收到普通文本消息事件

参数	描述
message	消息内容, 请参考: TUIMessage 定义
roomId	房间ID

onReceiveCustomMessage:message:

onReceiveCustomMessage:message:

```
- (void)onReceiveCustomMessage:(NSString *)roomId
                       message:(TUIMessage *)message
```

收到自定义消息事件

参数	描述
message	消息内容, 请参考: TUIMessage 定义
roomId	房间ID

onDeviceChanged:type:state:

onDeviceChanged:type:state:

```
- (void)onDeviceChanged:(NSString *)deviceId
                   type:(TUIMediaDeviceType)type
```

state: （TUIMediaDeviceState）state

本地设备添加事件

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。

注意
当本地设备（包括摄像头、麦克风以及扬声器）添加时，SDK 便会抛出此事件回调

TUICommonDefine

最近更新时间：2023-10-13 16:39:31

TUICommonDefine

结构体类型

结构体类型	描述
TUINetworkInfo	网络质量信息
TUIMessage	消息

枚举类型

枚举类型	描述
TUIError	错误码枚举定义
TUINetworkQuality	网络质量

回调类型

回调类型	描述
TUISuccessBlock	成功回调
TUIErrorBlock	失败回调

TUIError

TUIError

错误码枚举定义

枚举	取值	描述
TUIErrorSuccess	0	操作成功
TUIErrorFailed	-1	暂未归类的通用错误
TUIErrorFreqLimit	-2	请求被限频，请稍后重试
TUIErrorSDKAppIDNotFound	-1000	未找到SDKAppID，请在腾讯云视立方SDK控制台确认应用信息。

TUIErrorInvalidParameter	-1001	调用 API 时，传入的参数不合法，检查入参是否合法
TUIErrorSdkNotInitialized	-1002	未登录,请调用Login接口
TUIErrorPermissionDenied	-1003	获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限。Room场景下请使用以下错误码来处理: 摄像头没有系统授权: ERR_CAMERA_NOT_AUTHORIZED 麦克风没有系统授权: ERR_MICROPHONE_NOT_AUTHORIZED
TUIErrorRequirePayment	-1004	该功能需要开通额外的套餐，请在腾讯云视立方SDK按需开通对应套餐: https://console.cloud.tencent.com/vcube/project/manage
TUIErrorCameraStartFail	-1100	系统问题，打开摄像头失败。检查摄像头设备是否正常
TUIErrorCameraNotAuthorized	-1101	摄像头没有系统授权, 检查系统授权
TUIErrorCameraOccupied	-1102	摄像头被占用，检查是否有其他进程使用摄像头
TUIErrorCameraDeviceEmpty	-1103	当前无摄像头设备，请插入摄像头设备解决该问题
TUIErrorMicrophoneStartFail	-1104	系统问题，打开麦克风失败。检查麦克风设备是否正常
TUIErrorMicrophoneNotAuthorized	-1105	麦克风没有系统授权，检查系统授权
TUIErrorMicrophoneOccupied	-1106	麦克风被占用
TUIErrorMicrophoneDeviceEmpty	-1107	当前无麦克风设备
TUIErrorGetScreenSharingTargetFailed	-1108	获取屏幕分享源（屏幕和窗口）失败，检查屏幕录制权限
TUIErrorStartScreenSharingFailed	-1109	开启屏幕分享失败，检查房间内是否有人正在屏幕分享
TUIErrorRoomIdNotExist	-2100	进房时房间不存在，或许已被解散
TUIErrorOperationInvalidBeforeEnterRoom	-2101	需要进房后才可使用此功能
TUIErrorExitNotSupportedForRoomOwner	-2102	房主不支持退房操作，Conference(会议)房间类型: 可以先转让房主，再退房。LivingRoom(直播)房间类型: 房主只能解散房间
TUIErrorOperationNotSupportedInCurrentRoomType	-2103	当前房间类型下不支持该操作
TUIErrorOperationNotSupportedInCurrentSpeechMode	-2104	当前发言模式下不支持该操作

TUIErrorRoomIdInvalid	-2105	创建房间ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
TUIErrorRoomIdOccupied	-2106	房间ID 已被使用，请选择别的房间ID
TUIErrorRoomNameInvalid	-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
TUIErrorAlreadyInOtherRoom	-2108	当前用户已在别的房间内，需要先退房才能加入新的房间: 单个roomEngine实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的roomEngine实例。
TUIErrorUserNotExist	-2200	用户不存在
TUIErrorUserNotEntered	-2201	用户不在当前房间内
TUIErrorUserNeedOwnerPermission	-2300	需要房主权限才能操作
TUIErrorUserNeedAdminPermission	-2301	需要房主或者管理员权限才能操作
TUIErrorRequestNoPermission	-2310	信令请求无权限，比如取消非自己发起的邀请。
TUIErrorRequestIdInvalid	-2311	信令请求ID 无效或已经被处理过。
TUIErrorMaxSeatCountLimit	-2340	最大麦位超出套餐包数量限制
TUIErrorAlreadyInSeat	-2341	当前用户已经在麦位上
TUIErrorSeatOccupied	-2342	当前麦位已经有人了
TUIErrorSeatLocked	-2343	当前麦位被锁
TUIErrorSeatIndexNotExist	-2344	麦位编号不存在
TUIErrorUserNotInSeat	-2345	当前用户没有在麦上
TUIErrorAllSeatOccupied	-2346	上麦人数已满
TUIErrorOpenMicrophoneNeedSeatUnlock	-2360	当前麦位音频被锁
TUIErrorOpenMicrophoneNeedPermissionFromAdmin	-2361	需要向房主或管理员申请后打开麦克风
TUIErrorOpenCameraNeedSeatUnlock	-2370	当前麦位视频被锁, 需要由房主解锁麦位后，才能打开摄像头
TUIErrorOpenCameraNeedPermissionFromAdmin	-2371	需要向房主或管理员申请后打开摄像头
TUIErrorSendMessageDisabledForAll	-2380	当前房间已开启全员禁言
TUIErrorSendMessageDisabledForCurrent	-2381	当前房间内，你已被禁言

TUINetworkQuality

TUINetworkQuality

网络质量

枚举	取值	描述
TUINetworkQualityUnknown	0	未定义
TUINetworkQualityExcellent	1	当前网络非常好
TUINetworkQualityGood	2	当前网络比较好
TUINetworkQualityPoor	3	当前网络一般
TUINetworkQualityBad	4	当前网络较差
TUINetworkQualityVeryBad	5	当前网络很差
TUINetworkQualityDown	6	当前网络不满足 TRTC 的最低要求

TUINetworkInfo

TUINetworkInfo

网络质量信息

枚举类型	描述
delay	网络延迟，单位 ms
downLoss	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
quality	网络质量
upLoss	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
userId	用户ID

TUIMessage

TUIMessage

消息

枚举类型	描述

avatarURL	消息发送者头像
message	消息文本
messageId	消息 ID
timestamp	消息时间
userId	消息发送者
userName	消息发送者昵称

TUISuccessBlock

TUISuccessBlock

成功回调

```
void (^TUISuccessBlock)(void)
```

TUIErrorBlock

TUIErrorBlock

失败回调

```
void (^TUIErrorBlock)(TUIError code, NSString* _Nonnull message)
```

参数	描述
code	错误码
message	错误信息

TUIRoomDefine

最近更新时间：2023-09-28 15:21:01

TUIRoomDefine

结构体类型

函数列表	描述
TUIRoomInfo	房间信息
TUILoginUserInfo	用户登录信息
TUIUserInfo	房间内用户信息
TUIRoomVideoEncoderParams	视频编码参数
TUISeatInfo	房间内座位信息
TUISeatLockParams	锁定麦位操作参数
TUIUserVoiceVolume	房间内用户音量
TUIRequest	信令请求

枚举类型

枚举类型	描述
TUIRoomType	房间类型
TUISpeechMode	麦控模式
TUIMediaDevice	房间内媒体设备类型
TUIRole	房间内角色类型
TUIVideoQuality	视频质量
TUIAudioQuality	音频质量
TUIVideoStreamType	视频流类型
TUIChangeReason	音视频状态更改原因（分类: 自己主动修改 或者 被房主、管理员修改）
TUIKickedOutOfRoomReason	用户被踢出房间原因（分类: 被主持人或管理员踢出、在其它设备进入房间被踢出 或者 被服务端踢出）
TUIResolutionMode	分辨率模式（横屏 or 竖屏）
TUICaptureSourceType	屏幕分享捕获源类型
TUIMediaDeviceType	设备类型

TUIMediaDeviceState	设备操作
TUIRequestAction	请求类型

TUIRoomType

TUIRoomType

房间类型

枚举	取值	描述
TUIRoomTypeConference	1	会议类型房间，适用于会议，教育场景，该房间中可以开启自由发言，申请发言、上麦发言等不同模式
TUIRoomTypeLivingRoom	2	直播类型房间，适用于直播场景，该房间可以开启自由发言，上麦发言模式

TUISpeechMode

TUISpeechMode

麦控模式

枚举	取值	描述
TUISpeechModeFreeToSpeak	1	自由发言模式
TUISpeechModeApplyToSpeak	2	申请发言模式。（仅在会议类型房间下生效）
TUISpeechModeApplySpeakAfterTakingSeat	3	上麦发言模式。

TUIMediaDevice

TUIMediaDevice

房间内媒体设备类型

枚举	取值	描述
TUIMediaDeviceMicrophone	1	麦克风
TUIMediaDeviceCamera	2	摄像头
TUIMediaDeviceScreenSharing	3	屏幕共享

版权所有：腾讯云计算（北京）有限责任公司

TUIRole

TUIRole

房间内角色类型

枚举	取值	描述
TUIRoleRoomOwner	0	房主，一般指房间的创建者，房间内最高权限拥有者。
TUIRoleAdministrator	1	房间管理员
TUIRoleGeneralUser	2	房间内普通成员

TUIVideoQuality

TUIVideoQuality

视频质量

枚举	取值	描述
TUIVideoQuality360P	1	低清360P
TUIVideoQuality540P	2	标清540P
TUIVideoQuality720P	3	高清720P
TUIVideoQuality1080P	4	超清1080P

TUIAudioQuality

TUIAudioQuality

音频质量

枚举	取值	描述
TUIAudioQualitySpeech	0	人声模式
TUIAudioQualityDefault	1	默认模式
TUIAudioQualityMusic	2	音乐模式

TUIVideoStreamType

版权所有：腾讯云计算（北京）有限责任公司

TUIVideoStreamType

视频流类型

枚举	取值	描述
TUIVideoStreamTypeCameraStream	0	高清摄像头视频流
TUIVideoStreamTypeScreenStream	1	屏幕分享视频流
TUIVideoStreamTypeCameraStreamLow	2	低清摄像头视频流

TUIChangeReason

TUIChangeReason

音视频状态更改原因（分类: 自己主动修改 或者 被房主、管理员修改）

枚举	取值	描述
TUIChangeReasonBySelf	0	自己操作
TUIChangeReasonByAdmin	1	房主或管理员操作

TUIKickedOutOfRoomReason

TUIKickedOutOfRoomReason

用户被踢出房间原因（分类: 被主持人或管理员踢出、在其它设备进入房间被踢出 或者 被服务端踢出）

枚举	取值	描述
TUIKickedOutOfRoomReasonByAdmin	0	被主持人或管理员踢出
TUIKickedOutOfRoomReasonByLoggedOnOtherDevice	1	在其它设备进入房间被踢出
TUIKickedOutOfRoomReasonByServer	2	被服务端踢出

TUIResolutionMode

TUIResolutionMode

分辨率模式（横屏 or 竖屏）

枚举	取值	描述
TUIResolutionModeLandscape	0	横屏
TUIResolutionModePortrait	1	竖屏

TUICaptureSourceType

TUICaptureSourceType

屏幕分享捕获源类型

枚举	取值	描述
TUICaptureSourceTypeUnknown	-1	未定义
TUICaptureSourceTypeWindow	0	窗口
TUICaptureSourceTypeScreen	1	屏幕

TUIMediaDeviceType

TUIMediaDeviceType

设备类型

枚举	取值	描述
TUIMediaDeviceTypeUnknown	-1	未定义的设备类型
TUIMediaDeviceTypeAudioInput	0	麦克风类型设备
TUIMediaDeviceTypeAudioOutput	1	扬声器类型设备
TUIMediaDeviceTypeVideoCamera	2	摄像头类型设备

TUIMediaDeviceState

TUIMediaDeviceState

设备操作

枚举	取值	描述
TUIMediaDeviceStateAdd	0	设备已被插入
TUIMediaDeviceStateRemove	1	设备已被移除
TUIMediaDeviceStateActive	2	设备已启用

TUIRequestAction

TUIRequestAction

请求类型

枚举	取值	描述
TUIRequestActionInvalidAction	0	无效请求
TUIRequestActionOpenRemoteCamera	1	请求远端用户打开摄像头
TUIRequestActionOpenRemoteMicrophone	2	请求远端用户打开麦克风
TUIRequestActionConnectOtherRoom	3	请求连接到其他房间
TUIRequestActionTakeSeat	4	请求上麦
TUIRequestActionRemoteUserOnSeat	5	请求远端用户上麦
TUIRequestActionApplyToAdminToOpenLocalCamera	6	向管理员请求打开本地摄像头
TUIRequestActionApplyToAdminToOpenLocalMicrophone	7	向管理员请求打开本地麦克风

TUIRoomInfo

TUIRoomInfo

房间信息

TUIRoomEngine 只支持字符串房间ID

枚举类型	描述
createTime	房间创建时间（只读）
isCameraDisableForAllUser	是否禁止打开摄像头（创建房间可选参数），默认值：NO。
isMessageDisableForAllUser	是否禁止发送消息（创建房间可选参数），默认值：NO。
isMicrophoneDisableForAllUser	是否禁止打开麦克风（创建房间可选参数），默认值：NO。
maxSeatCount	最大麦位数

memberCount	房间内成员数量（只读）
name	房间名称（创建房间可选参数，默认房间ID）
ownerId	主持人ID: 默认为房间创建者（只读）
roomId	房间ID(创建房间必填参数)
roomType	房间类型（创建房间可选参数，默认Group类型），请参考：TUIRoomType。
speechMode	房间麦控模式

TUILoginUserInfo

TUILoginUserInfo

用户登录信息

枚举类型	描述
avatarUrl	用户头像URL
customInfo	自定义信息
userId	用户ID
userName	用户名称

TUIUserInfo

TUIUserInfo

房间内用户信息

枚举类型	描述
avatarUrl	用户头像URL
hasAudioStream	是否有音频流，默认值：NO。
hasScreenStream	是否有屏幕分享流，默认值：NO。
hasVideoStream	是否有视频流，默认值：NO。
roomCustomInfo	房间成员自定义信息
userId	用户ID
userName	用户名称
userRole	用户角色类型, 请参考：TUIRole。

TUIRoomVideoEncoderParams

TUIRoomVideoEncoderParams

视频编码参数

枚举类型	描述
bitrate	目标视频码率
fps	视频采集帧率
resolutionMode	分辨率模式, 请参考： TUIResolutionMode 。
videoResolution	视频质量, 请参考： TUIVideoQuality 。

TUISeatInfo

TUISeatInfo

房间内座位信息

枚举类型	描述
index	麦位序号
isAudioLocked	麦位是否被禁止打开麦克风，默认值：NO。
isLocked	麦位是否被锁定，默认值：NO。
isVideoLocked	麦位是否被禁止打开摄像头，默认值：NO。
userId	用户ID

TUISeatLockParams

TUISeatLockParams

锁定麦位操作参数

枚举类型	描述
lockAudio	锁定麦位麦克风，默认值：NO。
lockSeat	锁定麦位，默认值：NO。
lockVideo	锁定麦位摄像头，默认值：NO。

版权所有：腾讯云计算（北京）有限责任公司

TUIUserVoiceVolume

TUIUserVoiceVolume

房间内用户音量

枚举类型	描述
userId	用户ID
volume	音量 用于承载所有正在说话的用户的音量大小，取值范围 0 – 100

TUIRequest

TUIRequest

信令请求

枚举类型	描述
content	信令内容
requestAction	请求类型
requestId	请求ID
timestamp	时间戳
userId	用户ID

Android API 概览

最近更新时间：2023-09-28 15:21:01

API OVERVIEW

创建实例和事件回调

函数列表	描述
createInstance	创建 TUIRoomEngine 实例
destroyInstance	销毁 TUIRoomEngine 实例
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfo	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
addObserver	设置事件回调
removeObserver	移除事件回调

房间相关主动接口

函数列表	描述
createRoom	创建房间
destroyRoom	解散房间
enterRoom	进入房间
exitRoom	离开房间
connectOtherRoom	连接其他房间
disconnectOtherRoom	断开与其他房间的连接
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin	设置房间麦控模式（只有管理员或房主能够调用）

本地用户视图渲染、视频管理

函数列表	描述
setLocalVideoView	设置本地用户视频渲染的视图控件
openLocalCamera	打开本地摄像头
closeLocalCamera	关闭本地摄像头
startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality	更新本地视频编码质量设置
updateVideoQualityEx	设置视频编码器的编码参数
setVideoResolutionMode	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
enableGravitySensor	开启重力感应模式
startScreenSharing	开始屏幕分享（该接口仅支持移动端）
stopScreenSharing	结束屏幕分享

本地用户音频管理

函数列表	描述
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio	暂停发布本地的音频流
unmuteLocalAudio	恢复发布本地的音频流

远端用户视图渲染、视频管理

函数列表	描述
setRemoteVideoView	设置远端用户视频渲染的视图控件
startPlayRemoteVideo	开始播放远端用户视频
stopPlayRemoteVideo	停止播放远端用户视频
muteRemoteAudioStream	将远端用户禁音

房间内用户信息

函数列表	描述
getUserList	获取房间内的成员列表

getUserInfo	获取成员信息

房间内用户管理

函数列表	描述
changeUserRole	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或房主能够调用）
addCategoryTagForUsers	为用户增加标记（只有房主能够调用）
removeCategoryTagForUsers	为用户移除标记（只有房主能够调用）
getUserListByTag	根据标记获取房间内用户信息

房间内用户发言管理

函数列表	描述
disableDeviceForAllUserByAdmin	全体用户媒体设备管理
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）

房间内麦位管理

函数列表	描述
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList	获取麦位列表
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat	上麦（上麦发言模式下，需要申请）
leaveSeat	下麦
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin	主持人/管理员 将用户下麦

文本消息

函数列表	描述

sendTextMessage	发送本文消息
sendCustomMessage	发送自定义消息
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

信令管理

函数列表	描述
cancelRequest	取消请求
responseRemoteRequest	回复请求

高级功能

函数列表	描述
switchCamera	切换前置或后置摄像头（仅适用于移动端）
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别

调试相关

函数列表	描述
callExperimentalAPI	调用实验性接口

错误事件回调

函数列表	描述
onError	错误事件回调

登录状态事件回调

函数列表	描述
onKickedOffLine	用户被踢下线
onUserSigExpired	用户凭证超时事件

房间内事件回调

函数列表	描述

onRoomNameChanged	房间名称更改事件
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged	房间内用户发送文本消息被禁用事件
onRoomDismissed	房间被解散事件
onKickedOutOfRoom	被踢出房间事件
onRoomSpeechModeChanged	房间麦控模式发生变化

房间内用户事件回调

函数列表	描述
onRemoteUserEnterRoom	远端用户进房事件
onRemoteUserLeaveRoom	远端用户离开房间事件
onUserRoleChanged	用户角色发生变化事件
onUserVideoStateChanged	用户视频状态发生变化事件
onUserAudioStateChanged	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件
onUserScreenCaptureStopped	屏幕分享结束

房间内麦位事件回调

函数列表	描述
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged	麦位列表发生变化事件
onKickedOffSeat	收到用户被踢下麦事件

请求信令事件回调

函数列表	描述
onRequestReceived	收到请求消息事件
onRequestCancelled	收到请求被取消的事件

房间内消息事件回调

函数列表	描述
onReceiveTextMessage	收到普通文本消息事件
onReceiveCustomMessage	收到自定义消息事件

TUIRoomKit

最近更新时间：2024-03-06 14:51:31

Copyright (c) 2021 Tencent. All rights reserved.

Module: TUIRoomKit
Function: 多人音视频主功能接口
Version: 1.5

TUIRoomKit

TUIRoomKit

函数列表	描述
createInstance	创建 TUIRoomKit 实例（单例模式）。
destroyInstance	销毁 TUIRoomKit 实例。
setSelfInfo	设置个人信息，包括用户名和头像。
createRoom	创建房间。
enterRoom	进入房间。

createInstance

初始化 TUIRoomKit 单例对象。

```
public static TUIRoomKit createInstance();
```

destroyInstance

销毁 TUIRoomKit 实例。

```
public static void destroyInstance();
```

setSelfInfo

设置个人信息，包括用户名和头像。

```
public abstract void setSelfInfo(String userName, String avatarURL,
                                 TUIRoomDefine.ActionCallback callback);
```

参数	描述
userName	个人的用户名。

avatarURL	个人的头像链接。
callback	设置个人信息是否成功的回调。

createRoom

创建房间。

```
public abstract void createRoom(TUIRoomDefine.RoomInfo roomInfo, TUIRoomDefine.ActionCallback
callback);
```

参数	描述
roomInfo	创建房间的参数，包含房间号、房间名称等，其中 roomId 是必填项，其余可为默认值。
callback	创建房间是否成功的回调。

enterRoom

进入房间。

```
public abstract void enterRoom(String roomId,
                               boolean enableAudio,
                               boolean enableVideo,
                               boolean isSoundOnSpeaker,
                               TUIRoomDefine.GetRoomInfoCallback callback);
```

参数	描述
roomId	进入房间的房间号。
enableAudio	true 进房打开麦克风，并推送本地音频数据到远端，其他成员可以正常听到本地声音； false 进房只打开麦克风，不推送本地音频数据到远端，其他成员无法听到本地声音。
enableVideo	true 进房打开摄像头，并推送本地视频数据到远端，其他成员可以正常看到本地画面； false 进房不打开摄像头，也不推送本地视频数据到远端，其他成员无法看到本地画面。
isSoundOnSpeaker	是否使用扬声器播放声音，true 则使用扬声器，false 则使用听筒。
callback	进入房间是否成功的回调。

TUIRoomEngine

最近更新时间：2023-09-28 15:21:02

Copyright (c) 2021 Tencent. All rights reserved.

Module: TUIRoomEngine @ TUIKitEngine
Function: TUIRoomEngine 主功能接口
Version: 1.5

TUIRoomEngine

TUIRoomEngine

函数列表	描述
createInstance	创建 TUIRoomEngine 实例
destroyInstance	销毁 TUIRoomEngine 实例
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfo	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
setSelfInfo	设置本地用户信息
addObserver	设置事件回调
removeObserver	移除事件回调
createRoom	创建房间
destroyRoom	解散房间
enterRoom	进入房间
exitRoom	离开房间
connectOtherRoom	连接其他房间
disconnectOtherRoom	断开与其他房间的连接
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin	设置房间麦控模式（只有管理员或房主能够调用）
setLocalVideoView	设置本地用户视频渲染的视图控件

openLocalCamera	打开本地摄像头
closeLocalCamera	关闭本地摄像头
startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality	更新本地视频编码质量设置
updateVideoQualityEx	设置视频编码器的编码参数
setVideoResolutionMode	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
enableGravitySensor	开启重力感应模式
startScreenSharing	开始屏幕分享（该接口仅支持移动端）
stopScreenSharing	结束屏幕分享
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio	暂停发布本地的音频流
unmuteLocalAudio	恢复发布本地的音频流
setRemoteVideoView	设置远端用户视频渲染的视图控件
startPlayRemoteVideo	开始播放远端用户视频
stopPlayRemoteVideo	停止播放远端用户视频
muteRemoteAudioStream	将远端用户禁音
getUserList	获取房间内的成员列表
getUserInfo	获取成员信息
changeUserRole	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或房主能够调用）
addCategoryTagForUsers	为用户增加标记（只有房主能够调用）
removeCategoryTagForUsers	为用户移除标记（只有房主能够调用）
getUserListByTag	根据标记获取房间内用户信息
disableDeviceForAllUserByAdmin	全体用户媒体设备管理
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）

版权所有：腾讯云计算（北京）有限责任公司

getSeatList	获取麦位列表
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat	上麦（上麦发言模式下，需要申请）
leaveSeat	下麦
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin	主持人/管理员 将用户下麦
sendTextMessage	发送本文消息
sendCustomMessage	发送自定义消息
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）
cancelRequest	取消请求
responseRemoteRequest	回复请求
switchCamera	切换前置或后置摄像头（仅适用于移动端）
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别
callExperimentalAPI	调用实验性接口

createInstance

createInstance

创建 TUIRoomEngine 实例

注意
如果您使用 createInstance 获取 TUIRoomEngine 实例，并请使用 destroyInstance 释放对象指针。

destroyInstance

destroyInstance

销毁 TUIRoomEngine 实例

> **注意**
> 使用 destroyInstance 释放对象指针。

login

login

```
void login (Context context
            int sdkAppId
            String userId
            String userSig
            TUIRoomDefine.ActionCallback callback)
```

登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
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。
sdkAppId	它是腾讯云用于区分客户的唯一标识，进入腾讯云实时音视频 控制台 创建应用，即可看到SDKAppId
userId	用户ID，用于区分不同用户
userSig	用户签名，用于腾讯云流量的认证

logout

logout

```
void logout (TUIRoomDefine.ActionCallback callback)
```

退出登录接口，会有主动离开房间操作、销毁资源

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

setSelfInfo

setSelfInfo

```
void setSelfInfo (String userName
                  String avatarURL
                  TUIRoomDefine.ActionCallback callback)
```

设置本地用户名称和头像

参数	描述
avatarURL	用户头像URL地址
callback	调用接口的回调，用于通知接口调用的成功或者失败
userName	用户名称

getSelfInfo

getSelfInfo

获取本地用户登录的基本信息

返回值说明：

LoginUserInfo 用户登录信息

setSelfInfo

setSelfInfo

```
void setSelfInfo (TUIRoomDefine.LoginUserInfo userInfo
                  TUIRoomDefine.ActionCallback callback)
```

设置本地用户信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userInfo	本地用户信息

addObserver

addObserver

void addObserver	(TUIRoomObserver observer)

设置事件回调

您可以通过 TUIRoomObserver 获得各类事件通知（比如：错误码，远端用户进房，音视频状态参数等）

参数	描述
observer	监听的实例

removeObserver

removeObserver

void removeObserver	(TUIRoomObserver observer)

移除事件回调

参数	描述
observer	待移除的监听回调实例

createRoom

createRoom

void createRoom	(TUIRoomDefine.RoomInfo roomInfo TUIRoomDefine.ActionCallback callback)

创建房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomInfo	房间信息，可以初始化房间的一些设置

destroyRoom

destroyRoom

void	(TUIRoomDefine.ActionCallback callback)

版权所有：腾讯云计算（北京）有限责任公司

destroyRoom

解散房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

enterRoom

enterRoom

void enterRoom	(String roomId TUIRoomDefine.GetRoomInfoCallback callback)

进入房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomId	房间ID

exitRoom

exitRoom

void exitRoom	(boolean syncWaiting TUIRoomDefine.ActionCallback callback)

离开房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
syncWaiting	是否同步等待接口返回

connectOtherRoom

connectOtherRoom

Request	(String roomId

connectOtherRoom

String userId

int timeout

TUIRoomDefine.RequestCallback callback)

连接其他房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomId	房间ID
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

注意
用于直播场景下的申请跨房连麦

返回值说明：

TUIRequest 请求体

disconnectOtherRoom

disconnectOtherRoom

void disconnectOtherRoom (TUIRoomDefine.ActionCallback callback)

断开与其他房间的连接

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

注意
用于直播场景下的断开跨房连麦

fetchRoomInfo

fetchRoomInfo

void fetchRoomInfo (TUIRoomDefine.GetRoomInfoCallback callback)

获取房间信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

updateRoomNameByAdmin

updateRoomNameByAdmin

```
void updateRoomNameByAdmin (String roomName
                            TUIRoomDefine.ActionCallback callback)
```

更新房间名称（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomName	房间名称

updateRoomSpeechModeByAdmin

updateRoomSpeechModeByAdmin

```
void updateRoomSpeechModeByAdmin (TUIRoomDefine.SpeechMode mode
                                  TUIRoomDefine.ActionCallback callback)
```

设置房间麦控模式（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
mode	FREE_TO_SPEAK: 自由发言模式, 用户可以自由开启麦克风和扬声器; APPLY_TO_SPEAK: 申请发言模式，用户requestOpenLocalMicrophone 或 requestOpenLocalCamera 向房主或管理员申请后，方可打开麦克风和摄像头开始发言 SPEAK_AFTER_TAKING_SEAT: 上麦发言模式，KConference房间内，所有人在发言前，必须 takeSeat，才能进行麦克风和摄像头操作。

setLocalVideoView

setLocalVideoView

```
void setLocalVideoView (TUIRoomDefine.VideoStreamType streamType
                        TUIVideoView view)
```

设置本地用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 VideoStreamType 的定义
view	视频渲染视图

openLocalCamera

openLocalCamera

```
void openLocalCamera (boolean isFront
                      TUIRoomDefine.VideoQuality quality
                      TUIRoomDefine.ActionCallback callback)
```

打开本地摄像头

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isFront	true: 前置 false: 后置。该参数只在移动端生效

closeLocalCamera

closeLocalCamera

关闭本地摄像头

startPushLocalVideo

startPushLocalVideo

开始推送本地视频。默认开启

stopPushLocalVideo

stopPushLocalVideo

停止推送本地视频

updateVideoQuality

updateVideoQuality

void updateVideoQuality (TUIRoomDefine.VideoQuality quality)

更新本地视频编码质量设置

updateVideoQualityEx

updateVideoQualityEx

void updateVideoQualityEx (TUIRoomDefine.VideoStreamType streamType
TUIRoomDefine.VideoEncoderParams params)

设置视频编码器的编码参数

参数	描述
params	视频编码器的编码参数，定义可参考 VideoEncoderParams 的定义
streamType	视频流的类型，定义可参考 VideoStreamType 的定义

setVideoResolutionMode

setVideoResolutionMode

void setVideoResolutionMode (TUIRoomDefine.VideoStreamType streamType
TUIRoomDefine.ResolutionMode resolutionMode)

设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）

参数	描述
resolutionMode	分辨率模式，定义可参考 ResolutionMode 的定义
streamType	视频流的类型，定义可参考 VideoStreamType 的定义

版权所有：腾讯云计算（北京）有限责任公司

enableGravitySensor

enableGravitySensor

void enableGravitySensor (boolean enable)

开启重力感应模式

参数	描述
enable	true: 开启 false: 关闭

注意
开启该模式后，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

startScreenSharing

startScreenSharing

开始屏幕分享（该接口仅支持移动端）

stopScreenSharing

stopScreenSharing

结束屏幕分享

openLocalMicrophone

openLocalMicrophone

void openLocalMicrophone (TUIRoomDefine.AudioQuality quality
TUIRoomDefine.ActionCallback callback)

打开本地麦克风

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

closeLocalMicrophone

closeLocalMicrophone

关闭本地麦克风

muteLocalAudio

muteLocalAudio

暂停发布本地的音频流

unmuteLocalAudio

unmuteLocalAudio

void unmuteLocalAudio (TUIRoomDefine.ActionCallback callback)

恢复发布本地的音频流

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

setRemoteVideoView

setRemoteVideoView

void setRemoteVideoView (String userId

TUIRoomDefine.VideoStreamType streamType

TUIVideoView view)

设置远端用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 VideoStreamType 的定义
userId	远端用户ID

view	视频渲染视图

startPlayRemoteVideo

startPlayRemoteVideo

```
void startPlayRemoteVideo (String userId
                           TUIRoomDefine.VideoStreamType streamType
                           TUIRoomDefine.PlayCallback callback)
```

开始播放远端用户视频

参数	描述
callback	调用接口的回调，用于通知播放时的状态回调
streamType	视频流的类型 详细定义可以参考 VideoStreamType 的定义
userId	用户ID

stopPlayRemoteVideo

stopPlayRemoteVideo

```
void stopPlayRemoteVideo (String userId
                          TUIRoomDefine.VideoStreamType streamType)
```

停止播放远端用户视频

参数	描述
streamType	视频流的类型 详细定义可以参考 VideoStreamType 的定义
userId	用户ID

muteRemoteAudioStream

muteRemoteAudioStream

```
void muteRemoteAudioStream (String userId
                            boolean isMute)
```

将远端用户禁音

参数	描述
isMute	是否禁音
userId	用户ID

getUserList

getUserList

```
void getUserList (long nextSequence
                  TUIRoomDefine.GetUserListCallback callback)
```

获取房间内的成员列表

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败。成功回调中包含 UserInfo list
nextSequence	分页拉取标志，第一次拉取填0，回调成功 如果callback返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0

getUserInfo

getUserInfo

```
void getUserInfo (String userId
                  TUIRoomDefine.GetUserInfoCallback callback)
```

获取成员信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userId	用户ID

changeUserRole

changeUserRole

```
void changeUserRole (String userId
```

```
TUIRoomDefine.Role role
TUIRoomDefine.ActionCallback callback)
```

修改用户角色（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
role	用户角色 详细定义可以参考 Role 的定义
userId	用户ID

kickRemoteUserOutOfRoom

kickRemoteUserOutOfRoom

```
void kickRemoteUserOutOfRoom (String userId
                              TUIRoomDefine.ActionCallback callback)
```

将远端用户踢出房间（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userId	用户ID

addCategoryTagForUsers

addCategoryTagForUsers

```
void addCategoryTagForUsers (int tag
                             List<String> userList
                             TUIRoomDefine.ActionCallback callback)
```

为用户增加标记（只有房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
tag	标记类型。数字类型，大于等于 1000，您可以自定义。

userList	用户列表

removeCategoryTagForUsers

removeCategoryTagForUsers

```
void removeCategoryTagForUsers (int tag
                                List<String> userList
                                TUIRoomDefine.ActionCallback callback)
```

为用户移除标记（只有房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
tag	标记类型。数字类型，大于等于 1000，您可以自定义。
userList	用户列表

getUserListByTag

getUserListByTag

```
void getUserListByTag (int tag
                       long nextSequence
                       TUIRoomDefine.GetUserListCallback callback)
```

根据标记获取房间内用户信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
nextSequence	分页拉取标志，第一次拉取填0，回调成功 如果callback返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0
tag	标记类型。数字类型，大于等于 1000，您可以自定义。

disableDeviceForAllUserByAdmin

disableDeviceForAllUserByAdmin

void disableDeviceForAllUserByAdmin	(TUIRoomDefine.MediaDevice device boolean isDisable TUIRoomDefine.ActionCallback callback)

全体用户媒体设备管理

参数	描述
callback	操作回调
device	设备。详细定义参考:MediaDevice
isDisable	否禁用

openRemoteDeviceByAdmin

openRemoteDeviceByAdmin

Request openRemoteDeviceByAdmin	(String userId TUIRoomDefine.MediaDevice device int timeout TUIRoomDefine.RequestCallback callback)

请求远端用户打开媒体设备（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知请求状态的回调，详细定义参考: RequestCallback
device	媒体设备。详细定义参考:MediaDevice
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

closeRemoteDeviceByAdmin

closeRemoteDeviceByAdmin

void closeRemoteDeviceByAdmin	(String userId

TUIRoomDefine.MediaDevice device

TUIRoomDefine.ActionCallback callback)

关闭远端用户媒体设备（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
device	媒体设备。详细定义参考:MediaDevice
userId	用户ID

applyToAdminToOpenLocalDevice

applyToAdminToOpenLocalDevice

Request applyToAdminToOpenLocalDevice (TUIRoomDefine.MediaDevice device

int timeout

TUIRoomDefine.RequestCallback callback)

请求打开本地媒体设备（普通用户可用）

参数	描述
callback	调用接口的回调，用于通知请求的回调状态，详细定义参考: RequestCallback
device	用户ID device 媒体设备。详细定义参考:MediaDevice
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

返回值说明：

TUIRequest 请求体

setMaxSeatCount

setMaxSeatCount

void setMaxSeatCount (int maxSeatCount

TUIRoomDefine.ActionCallback callback)

设置最大麦位数（仅支持进房前和创建房间时设置）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
maxSeatCount	最大麦位数

getSeatList

getSeatList

void getSeatList (TUIRoomDefine.GetSeatListCallback callback)

获取麦位列表

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败，成功回调中会包含 SeatInfo list信息

lockSeatByAdmin

lockSeatByAdmin

void lockSeatByAdmin (int seatIndex

TUIRoomDefine.SeatLockParams lockParams

TUIRoomDefine.ActionCallback callback)

锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
lockParams	锁麦参数。详情参考:TUIRoomDefine.SeatLockParams
seatIndex	麦位编号

takeSeat

takeSeat

Request takeSeat (int seatIndex

int timeout

TUIRoomDefine.RequestCallback callback)

上麦（上麦发言模式下，需要申请）

参数	描述
callback	调用接口的回调，用于通知请求的回调状态，详细定义参考: RequestCallback
seatIndex	麦位编号
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

注意
开启上麦发言模式时，需要向主持人或管理员发起申请才允许上麦。
开启自由发言模式，直播场景可以自由上麦，上麦后开麦发言，会议场景无需调用该接口，即可开麦发言。

返回值说明：

TUIRequest 请求体

leaveSeat

leaveSeat

void leaveSeat (TUIRoomDefine.ActionCallback callback)

下麦

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

takeUserOnSeatByAdmin

takeUserOnSeatByAdmin

Request takeUserOnSeatByAdmin (int seatIndex
String userId
int timeout
TUIRoomDefine.RequestCallback callback)

主持人/管理员 邀请用户上麦

参数	描述

callback	调用接口的回调，用于通知请求的回调状态，详细定义参考: RequestCallback
seatIndex	麦位编号。会议场景无需关心，填0即可。
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

kickUserOffSeatByAdmin

kickUserOffSeatByAdmin

```
void kickUserOffSeatByAdmin (int seatIndex
                             String userId
                             TUIRoomDefine.ActionCallback callback)
```

主持人/管理员 将用户下麦

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
seatIndex	麦位编号
userId	用户ID

sendTextMessage

sendTextMessage

```
void sendTextMessage (String message
                      TUIRoomDefine.ActionCallback callback)
```

发送本文消息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
message	消息内容

sendCustomMessage

sendCustomMessage

```
void sendCustomMessage (String message
                        TUIRoomDefine.ActionCallback callback)
```

发送自定义消息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
message	消息内容

disableSendingMessageByAdmin

disableSendingMessageByAdmin

```
void disableSendingMessageByAdmin (String userId
                                   boolean isDisable
                                   TUIRoomDefine.ActionCallback callback)
```

禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isDisable	是否禁用
userId	用户ID

disableSendingMessageForAllUser

disableSendingMessageForAllUser

```
void disableSendingMessageForAllUser (boolean isDisable
                                      TUIRoomDefine.ActionCallback callback)
```

禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isDisable	是否禁用

cancelRequest

cancelRequest

```
void cancelRequest (String requestId
                    TUIRoomDefine.ActionCallback callback)
```

取消请求

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
可以使用此接口来取消已发出的请求

responseRemoteRequest

responseRemoteRequest

```
void responseRemoteRequest (String requestId
                            boolean agree
                            TUIRoomDefine.ActionCallback callback)
```

回复请求

参数	描述
agree	是否同意 true: 同意请求, false: 拒绝请求
callback	调用接口的回调，用于通知接口调用的成功或者失败
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
在收到信令请求时，可以使用此接口来回复接收到的请求

switchCamera

switchCamera

int switchCamera (boolean frontCamera)

切换前置或后置摄像头（仅适用于移动端）

返回值说明：

0：操作成功；负数：操作失败

setBeautyLevel

setBeautyLevel

void setBeautyLevel (int beautyStyle

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。
beautyStyle	美颜风格，TXBeautyStyleSmooth：光滑；TXBeautyStyleNature：自然；TXBeautyStylePitu：优图。

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。

callExperimentalAPI

callExperimentalAPI

void callExperimentalAPI (String jsonStr)

调用实验性接口

参数	描述
jsonStr	接口信息

TUIRoomObserver

最近更新时间：2023-09-14 16:46:13

Copyright (c) 2021 Tencent. All rights reserved.
Module: TUIRoomObserver @ TUIKitEngine
Function: TUIRoomEngine的事件回调接口

TUIRoomObserver

TUIRoomObserver

函数列表	描述
onError	错误事件回调
onKickedOffLine	用户被踢下线
onUserSigExpired	用户凭证超时事件
onRoomNameChanged	房间名称更改事件
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged	房间内用户发送文本消息被禁用事件
onRoomDismissed	房间被解散事件
onKickedOutOfRoom	被踢出房间事件
onRoomSpeechModeChanged	房间麦控模式发生变化
onRemoteUserEnterRoom	远端用户进房事件
onRemoteUserLeaveRoom	远端用户离开房间事件
onUserRoleChanged	用户角色发生变化事件
onUserVideoStateChanged	用户视频状态发生变化事件
onUserAudioStateChanged	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件
onUserScreenCaptureStopped	屏幕分享结束
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）

onSeatListChanged	麦位列表发生变化事件
onKickedOffSeat	收到用户被踢下麦事件
onRequestReceived	收到请求消息事件
onRequestCancelled	收到请求被取消的事件
onReceiveTextMessage	收到普通文本消息事件
onReceiveCustomMessage	收到自定义消息事件

onError

onError

```
void onError (TUICommonDefine.Error errorCode
              String message)
```

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。

参数	描述
errorCode	错误码，请参考：Error
message	错误信息

onKickedOffLine

onKickedOffLine

```
void onKickedOffLine (String message)
```

用户被踢下线

参数	描述
message	被踢下线的描述

onUserSigExpired

onUserSigExpired

用户凭证超时事件

onRoomNameChanged

onRoomNameChanged

```
void
onRoomNameChanged        (String roomId

                          String roomName)
```

房间名称更改事件

参数	描述
roomId	房间ID
roomName	房间名称

onAllUserMicrophoneDisableChanged

onAllUserMicrophoneDisableChanged

```
void
onAllUserMicrophoneDisableChanged        (String roomId

                                         boolean isDisable)
```

房间内所有用户麦克风被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onAllUserCameraDisableChanged

onAllUserCameraDisableChanged

```
void
onAllUserCameraDisableChanged        (String roomId

                                     boolean isDisable)
```

房间内所有用户摄像头被禁用事件

参数	描述

isDisable	是否被禁用
roomId	房间ID

onSendMessageForAllUserDisableChanged

onSendMessageForAllUserDisableChanged

```
void onSendMessageForAllUserDisableChanged(String roomId, boolean isDisable)
```

房间内用户发送文本消息被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onRoomDismissed

onRoomDismissed

```
void onRoomDismissed(String roomId)
```

房间被解散事件

参数	描述
roomId	房间ID

onKickedOutOfRoom

onKickedOutOfRoom

```
void onKickedOutOfRoom(String roomId, TUIRoomDefine.KickedOutOfRoomReason reason, String message)
```

被踢出房间事件

参数	描述
message	被踢出的描述
reason	被踢出原因
roomId	房间ID

onRoomSpeechModeChanged

onRoomSpeechModeChanged

```
void onRoomSpeechModeChanged (String roomId
                              TUIRoomDefine.SpeechMode speechMode)
```

房间麦控模式发生变化

参数	描述
mode	房间模式
roomId	房间ID

onRemoteUserEnterRoom

onRemoteUserEnterRoom

```
void onRemoteUserEnterRoom (String roomId
                            TUIRoomDefine.UserInfo userInfo)
```

远端用户进房事件

参数	描述
roomId	房间ID
userInfo	用户信息

onRemoteUserLeaveRoom

onRemoteUserLeaveRoom

```
void onRemoteUserLeaveRoom (String roomId
```

TUIRoomDefine.UserInfo userInfo)

远端用户离开房间事件

参数	描述
roomId	房间ID
userInfo	用户信息

onUserRoleChanged

onUserRoleChanged

void
onUserRoleChanged (String userId

TUIRoomDefine.Role user)

用户角色发生变化事件

参数	描述
userId	用户ID
userRole	用户角色 可参考 Role 枚举定义

onUserVideoStateChanged

onUserVideoStateChanged

void
onUserVideoStateChanged (String userId

TUIRoomDefine.VideoStreamType streamType

boolean hasVideo

TUIRoomDefine.ChangeReason reason)

用户视频状态发生变化事件

参数	描述
hasVideo	是否有视频流
reason	视频流发生变化原因 BY_SELF：自己切换 BY_ADMIN：被管理员切换
streamType	视频流类型

userId	用户ID

onUserAudioStateChanged

onUserAudioStateChanged

```
void
onUserAudioStateChanged (String userId
                         boolean hasAudio
                         TUIRoomDefine.ChangeReason reason)
```

用户音频状态发生变化事件

参数	描述
hasAudio	是否有音频流
reason	视频流发生变化原因 BY_SELF：自己切换 BY_ADMIN：被管理员切换
userId	用户ID

onUserVoiceVolumeChanged

onUserVoiceVolumeChanged

```
void
onUserVoiceVolumeChanged (Map<String, Integer> volumeMap)
```

用户音量变化事件

参数	描述
volumeMap	用户音量字典 key: userId, value: 用于承载所有正在说话的用户的音量大小，取值范围 0 – 100。

onSendMessageForUserDisableChanged

onSendMessageForUserDisableChanged

```
void
onSendMessageForUserDisableChanged (String roomId
                                    String userId
                                    boolean isDisable)
```

用户文本消息发送能力发生变化事件

参数	描述
isDisable	是否被禁止发送文本消息 true: 用户被禁止发送消息 false: 用户被解除禁止，可以发送消息
userId	用户ID

onUserNetworkQualityChanged

onUserNetworkQualityChanged

```
void
onUserNetworkQualityChanged (Map<String, TUICommonDefine.NetworkInfo> networkMap)
```

用户网络状态变化事件

参数	描述
networkMap	用户网络状态Map，可参考 NetworkInfo 对象

onUserScreenCaptureStopped

onUserScreenCaptureStopped

```
void
onUserScreenCaptureStopped (int reason)
```

屏幕分享结束

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）

onRoomMaxSeatCountChanged

onRoomMaxSeatCountChanged

```
void
onRoomMaxSeatCountChanged (String roomId
                           int maxSeatCount)
```

房间内最大麦位数发生变化事件（仅在会议类型房间生效）

参数	描述
maxSeatCount	房间内最大麦位数量
roomId	房间ID

onSeatListChanged

onSeatListChanged

void onSeatListChanged (List<TUIRoomDefine.SeatInfo> seatList

List<TUIRoomDefine.SeatInfo> seatedList

List<TUIRoomDefine.SeatInfo> leftList)

麦位列表发生变化事件

参数	描述
leftList	新下麦的用户列表
seatList	目前麦上最新的用户列表，包含新上麦的用户
seatedList	新上麦的用户列表

onKickedOffSeat

onKickedOffSeat

void onKickedOffSeat (String userId)

收到用户被踢下麦事件

参数	描述
userId	操作踢人的（主持人/管理员）用户id

onRequestReceived

onRequestReceived

void onRequestReceived (TUIRoomDefine.Request request)

收到请求消息事件

参数	描述
request	请求内容，可参考 Request 对象

onRequestCancelled

onRequestCancelled

```
void
onRequestCancelled
(String requestId
String userId)
```

收到请求被取消的事件

参数	描述
requestId	请求ID
userId	取消信令的用户ID

onReceiveTextMessage

onReceiveTextMessage

```
void
onReceiveTextMessage
(String roomId
TUICommonDefine.Message message)
```

收到普通文本消息事件

参数	描述
message	消息内容, 请参考: Message 定义
roomId	房间ID

onReceiveCustomMessage

onReceiveCustomMessage

```
void
onReceiveCustomMessage
(String roomId
TUICommonDefine.Message message)
```

收到自定义消息事件

参数	描述
message	消息内容, 请参考: Message 定义
roomId	房间ID

TUICommonDefine

最近更新时间：2023-09-14 16:46:13

TUICommonDefine

结构体类型

函数列表	描述
NetworkInfo	网络质量信息
Message	消息

枚举类型

枚举类型	描述
Error	错误码枚举定义
NetworkQuality	网络质量

TUIError

TUIError

错误码枚举定义

枚举	取值	描述
SUCCESS	0	操作成功
FAILED	-1	暂未归类的通用错误
FREQ_LIMIT	-2	请求被限频，请稍后重试
SDKAPPID_NOT_FOUND	-1000	未找到SDKAppID，请在腾讯云视立方SDK控制台确认应用信息。
INVALID_PARAMETER	-1001	调用 API 时，传入的参数不合法，检查入参是否合法
SDK_NOT_INITIALIZED	-1002	未登录,请调用Login接口
PERMISSION_DENIED	-1003	获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限。Room场景下请使用以下错误码来处理: 摄像头没有系统授权: ERR_CAMERA_NOT_AUTHORIZED

		麦克风没有系统授权: ERR_MICROPHONE_NOT_AUTHORIZED
REQUIRE_PAYMENT	-1004	该功能需要开通额外的套餐，请在腾讯云视立方SDK按需开通对应套餐: https://console.cloud.tencent.com/vcube/project/manage
CAMERA_START_FAIL	-1100	系统问题，打开摄像头失败。检查摄像头设备是否正常
CAMERA_NOT_AUTHORIZED	-1101	摄像头没有系统授权, 检查系统授权
CAMERA_OCCUPIED	-1102	摄像头被占用，检查是否有其他进程使用摄像头
CAMERA_DEVICE_EMPTY	-1103	当前无摄像头设备，请插入摄像头设备解决该问题
MICROPHONE_START_FAIL	-1104	系统问题，打开麦克风失败。检查麦克风设备是否正常
MICROPHONE_NOT_AUTHORIZED	-1105	麦克风没有系统授权，检查系统授权
MICROPHONE_OCCUPIED	-1106	麦克风被占用
MICROPHONE_DEVICE_EMPTY	-1107	当前无麦克风设备
GET_SCREEN_SHARING_TARGET_FAILED	-1108	获取屏幕分享源（屏幕和窗口）失败，检查屏幕录制权限
START_SCREEN_SHARING_FAILED	-1109	开启屏幕分享失败，检查房间内是否有人正在屏幕分享
ROOM_ID_NOT_EXIST	-2100	进房时房间不存在，或许已被解散
OPERATION_INVALID_BEFORE_ENTER_ROOM	-2101	需要进房后才可使用此功能
EXIT_NOT_SUPPORTED_FOR_ROOM_OWNER	-2102	房主不支持退房操作，Conference(会议)房间类型: 可以先转让房主，再退房。LivingRoom(直播)房间类型: 房主只能解散房间
OPERATION_NOT_SUPPORTED_IN_CURRENT_ROOM_TYPE	-2103	当前房间类型下不支持该操作
OPERATION_NOT_SUPPORTED_IN_CURRENT_SPEECH_MODE	-2104	当前发言模式下不支持该操作
ROOM_ID_INVALID	-2105	创建房间ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
ROOM_ID_OCCUPIED	-2106	房间ID 已被使用，请选择别的房间ID
ROOM_NAME_INVALID	-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
ALREADY_IN_OTHER_ROOM	-2108	当前用户已在别的房间内，需要先退房才能加入新的房间:

		单个roomEngine实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的roomEngine实例。
USER_NOT_EXIST	-2200	用户不存在
USER_NOT_ENTERED	-2201	用户不在当前房间内
NEED_OWNER_PERMISSION	-2300	需要房主权限才能操作
NEED_ADMIN_PERMISSION	-2301	需要房主或者管理员权限才能操作
REQUEST_NO_PERMISSION	-2310	信令请求无权限，比如取消非自己发起的邀请。
REQUEST_ID_INVALID	-2311	信令请求ID 无效或已经被处理过。
MAX_SEAT_COUNT_LIMIT	-2340	最大麦位超出套餐包数量限制
ALREADY_IN_SEAT	-2341	当前用户已经在麦位上
SEAT_OCCUPIED	-2342	当前麦位已经有人了
SEAT_LOCKED	-2343	当前麦位被锁
SEAT_INDEX_NOT_EXIST	-2344	麦位编号不存在
USER_NOT_IN_SEAT	-2345	当前用户没有在麦上
ALL_SEAT_OCCUPIED	-2346	上麦人数已满
OPEN_MICROPHONE_NEED_SEAT_UNLOCK	-2360	当前麦位音频被锁
OPEN_MICROPHONE_NEED_PERMISSION_FROM_ADMIN	-2361	需要向房主或管理员申请后打开麦克风
OPEN_CAMERA_NEED_SEAT_UNLOCK	-2370	当前麦位视频被锁，需要由房主解锁麦位后，才能打开摄像头
OPEN_CAMERA_NEED_PERMISSION_FROM_ADMIN	-2371	需要向房主或管理员申请后打开摄像头
SEND_MESSAGE_DISABLED_FOR_ALL	-2380	当前房间已开启全员禁言
SEND_MESSAGE_DISABLED_FOR_CURRENT	-2381	当前房间内，你已被禁言

TUINetworkQuality

TUINetworkQuality

网络质量

枚举	取值	描述
UNKNOWN	0	未定义

EXCELLENT	1	当前网络非常好
GOOD	2	当前网络比较好
POOR	3	当前网络一般
BAD	4	当前网络较差
VERY_BAD	5	当前网络很差
DOWN	6	当前网络不满足 TRTC 的最低要求

TUINetworkInfo

TUINetworkInfo

网络质量信息

枚举类型	描述
delay	网络延迟，单位 ms
downLoss	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
quality	网络质量
upLoss	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
userId	用户ID

TUIMessage

TUIMessage

消息

枚举类型	描述
avatarUrl	消息发送者头像
message	消息文本
messageId	消息 ID
timestamp	消息时间
userId	消息发送者

userName	消息发送者昵称

TUIRoomDefine

最近更新时间：2023-09-28 15:21:02

TUIRoomDefine

结构体类型

函数列表	描述
RoomInfo	房间信息
LoginUserInfo	用户登录信息
UserInfo	房间内用户信息
RoomVideoEncoderParams	视频编码参数
SeatInfo	房间内座位信息
SeatLockParams	锁定麦位操作参数
UserVoiceVolume	房间内用户音量
Request	信令请求

枚举类型

枚举类型	描述
RoomType	房间类型
SpeechMode	麦控模式
MediaDevice	房间内媒体设备类型
Role	房间内角色类型
VideoQuality	视频质量
AudioQuality	音频质量
VideoStreamType	视频流类型
ChangeReason	音视频状态更改原因（分类: 自己主动修改 或者 被房主、管理员修改）
KickedOutOfRoomReason	用户被踢出房间原因（分类: 被主持人或管理员踢出、在其它设备进入房间被踢出 或者 被服务端踢出）
ResolutionMode	分辨率模式（横屏 or 竖屏）
RequestAction	请求类型

TUIRoomType

TUIRoomType

房间类型

枚举	取值	描述
CONFERENCE	1	会议类型房间，适用于会议，教育场景，该房间中可以开启自由发言，申请发言、上麦发言等不同模式
LIVE_ROOM	2	直播类型房间，适用于直播场景，该房间可以开启自由发言，上麦发言模式

TUISpeechMode

TUISpeechMode

麦控模式

枚举	取值	描述
FREE_TO_SPEAK	1	自由发言模式
APPLY_TO_SPEAK	2	申请发言模式。（仅在会议类型房间下生效）
SPEAK_AFTER_TAKING_SEAT	3	上麦发言模式。

TUIMediaDevice

TUIMediaDevice

房间内媒体设备类型

枚举	取值	描述
MICROPHONE	1	麦克风
CAMERA	2	摄像头
SCREEN_SHARING	3	屏幕共享

TUIRole

TUIRole

房间内角色类型

枚举	取值	描述
ROOM_OWNER	0	房主，一般指房间的创建者，房间内最高权限拥有者。
MANAGER	1	房间管理员
GENERAL_USER	2	房间内普通成员

TUIVideoQuality

TUIVideoQuality

视频质量

枚举	取值	描述
Q_360P	1	低清360P
Q_540P	2	标清540P
Q_720P	3	高清720P
Q_1080P	4	超清1080P

TUIAudioQuality

TUIAudioQuality

音频质量

枚举	取值	描述
SPEECH	0	人声模式
DEFAULT	1	默认模式
MUSIC	2	音乐模式

TUIVideoStreamType

TUIVideoStreamType

视频流类型

枚举	取值	描述
CAMERA_STREAM	0	高清摄像头视频流
SCREEN_STREAM	1	屏幕分享视频流
CAMERA_STREAM_LOW	2	低清摄像头视频流

TUIChangeReason

TUIChangeReason

音视频状态更改原因（分类: 自己主动修改 或者 被房主、管理员修改）

枚举	取值	描述
BY_SELF	0	自己操作
BY_ADMIN	1	房主或管理员操作

TUIKickedOutOfRoomReason

TUIKickedOutOfRoomReason

用户被踢出房间原因（分类: 被主持人或管理员踢出、在其它设备进入房间被踢出 或者 被服务端踢出）

枚举	取值	描述
BY_ADMIN	0	被主持人或管理员踢出
BY_LOGGED_ON_OTHER_DEVICE	1	在其它设备进入房间被踢出
BY_SERVER	2	被服务端踢出

TUIResolutionMode

TUIResolutionMode

分辨率模式（横屏 or 竖屏）

枚举	取值	描述
LANDSCAPE	0	横屏
PORTRAIT	1	竖屏

版权所有：腾讯云计算（北京）有限责任公司

TUIRequestAction

TUIRequestAction

请求类型

枚举	取值	描述
INVALID_ACTION	0	无效请求
REQUEST_TO_OPEN_REMOTE_CAMERA	1	请求远端用户打开摄像头
REQUEST_TO_OPEN_REMOTE_MICROPHONE	2	请求远端用户打开麦克风
REQUEST_TO_CONNECT_OTHER_ROOM	3	请求连接到其他房间
REQUEST_TO_TAKE_SEAT	4	请求上麦
REQUEST_REMOTE_USER_ON_SEAT	5	请求远端用户上麦
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_CAMERA	6	向管理员请求打开本地摄像头
REQUEST_APPLY_TO_ADMIN_TO_OPEN_LOCAL_MICROPHONE	7	向管理员请求打开本地麦克风

TUIRoomInfo

TUIRoomInfo

房间信息

TUIRoomEngine 只支持字符串房间ID

枚举类型	描述
createTime	房间创建时间（只读）
isCameraDisableForAllUser	是否禁止打开摄像头（创建房间可选参数），默认值：false。
isMessageDisableForAllUser	是否禁止发送消息（创建房间可选参数），默认值：false。
isMicrophoneDisableForAllUser	是否禁止打开麦克风（创建房间可选参数），默认值：false。
maxSeatCount	最大麦位数
memberCount	房间内成员数量（只读）
name	房间名称（创建房间可选参数，默认房间ID）
ownerId	主持人ID: 默认为房间创建者（只读）

roomId	房间ID(创建房间必填参数)
roomType	房间类型（创建房间可选参数，默认Group类型），请参考：RoomType。
speechMode	房间麦控模式

TUIUserInfo

TUIUserInfo

用户登录信息

枚举类型	描述
avatarUrl	用户头像URL
customInfo	自定义信息
userId	用户ID
userName	用户名称

TUIUserInfo

TUIUserInfo

房间内用户信息

枚举类型	描述
avatarUrl	用户头像URL
hasAudioStream	是否有音频流，默认值：false。
hasScreenStream	是否有屏幕分享流，默认值：false。
hasVideoStream	是否有视频流，默认值：false。
roomCustomInfo	房间成员自定义信息
userId	用户ID
userName	用户名称
userRole	用户角色类型, 请参考：Role。

TUIRoomVideoEncoderParams

TUIRoomVideoEncoderParams

视频编码参数

枚举类型	描述
bitrate	目标视频码率
fps	视频采集帧率
resolutionMode	分辨率模式, 请参考： ResolutionMode 。
videoResolution	视频质量, 请参考： VideoQuality 。

TUISeatInfo

TUISeatInfo

房间内座位信息

枚举类型	描述
index	麦位序号
isAudioLocked	麦位是否被禁止打开麦克风，默认值：false。
isLocked	麦位是否被锁定，默认值：false。
isVideoLocked	麦位是否被禁止打开摄像头，默认值：false。
userId	用户ID

TUISeatLockParams

TUISeatLockParams

锁定麦位操作参数

枚举类型	描述
lockAudio	锁定麦位麦克风，默认值：false。
lockSeat	锁定麦位，默认值：false。
lockVideo	锁定麦位摄像头，默认值：false。

TUIUserVoiceVolume

TUIUserVoiceVolume

房间内用户音量

枚举类型	描述
userId	用户ID
volume	音量 用于承载所有正在说话的用户的音量大小，取值范围 0 – 100

TUIRequest

TUIRequest

信令请求

枚举类型	描述
content	信令内容
requestAction	请求类型
requestId	请求ID
timestamp	时间戳
userId	用户ID

Windows API 概览

最近更新时间：2023-09-28 15:21:02

API OVERVIEW

创建实例和事件回调

函数列表	描述
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfo	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
addObserver	设置事件回调
removeObserver	移除事件回调

房间相关主动接口

函数列表	描述
createRoom	创建房间
destroyRoom	解散房间
enterRoom	进入房间
exitRoom	离开房间
connectOtherRoom	连接其他房间
disconnectOtherRoom	断开与其他房间的连接
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin	设置房间管理模式（只有管理员或房主能够调用）

本地用户视图渲染、视频管理

函数列表	描述
setLocalVideoView	设置本地用户视频渲染的视图控件
openLocalCamera	打开本地摄像头

closeLocalCamera	关闭本地摄像头
startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality	更新本地视频编码质量设置
updateVideoQualityEx	设置视频编码器的编码参数
setVideoResolutionMode	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
startScreenSharing	开始屏幕分享。
stopScreenSharing	结束屏幕分享
getScreenSharingTargetList	获取分享对象列表
selectScreenSharingTarget	选择屏幕分享对象

本地用户音频管理

函数列表	描述
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio	暂停发布本地的音频流
unmuteLocalAudio	恢复发布本地的音频流

远端用户视图渲染、视频管理

函数列表	描述
setRemoteVideoView	设置远端用户视频渲染的视图控件
startPlayRemoteVideo	开始播放远端用户视频
stopPlayRemoteVideo	停止播放远端用户视频
muteRemoteAudioStream	将远端用户禁音

房间内用户信息

函数列表	描述
getUserList	获取房间内的成员列表
getUserInfo	获取成员信息

房间内用户管理

函数列表	描述
changeUserRole	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或房主能够调用）

房间内用户发言管理

函数列表	描述
disableDeviceForAllUserByAdmin	全体用户媒体设备管理
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）

房间内麦位管理

函数列表	描述
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList	获取麦位列表
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat	上麦（上麦发言模式下，需要申请）
leaveSeat	下麦
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin	主持人/管理员 将用户下麦

文本消息

函数列表	描述
sendTextMessage	发送本文消息
sendCustomMessage	发送自定义消息
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

信令管理

函数列表	描述
cancelRequest	取消请求
responseRemoteRequest	回复请求

高级功能

函数列表	描述
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）
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别

调试相关

函数列表	描述
callExperimentalAPI	调用实验性接口

错误事件回调

函数列表	描述
onError	错误事件回调

登录状态事件回调

函数列表	描述
onKickedOffLine	用户被踢下线
onUserSigExpired	用户凭证超时事件

房间内事件回调

函数列表	描述
onRoomNameChanged	房间名称更改事件
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged	房间内用户发送文本消息被禁用事件

onRoomDismissed	房间被解散事件
onKickedOutOfRoom	被踢出房间事件
onRoomSpeechModeChanged	房间麦控模式发生变化

房间内用户事件回调

函数列表	描述
onRemoteUserEnterRoom	远端用户进房事件
onRemoteUserLeaveRoom	远端用户离开房间事件
onUserRoleChanged	用户角色发生变化事件
onUserVideoStateChanged	用户视频状态发生变化事件
onUserAudioStateChanged	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件
onUserScreenCaptureStopped	屏幕分享结束

房间内麦位事件回调

函数列表	描述
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）
onSeatListChanged	麦位列表发生变化事件
onKickedOffSeat	收到用户被踢下麦事件

请求信令事件回调

函数列表	描述
onRequestReceived	收到请求消息事件
onRequestCancelled	收到请求被取消的事件

房间内消息事件回调

函数列表	描述
onReceiveTextMessage	收到普通文本消息事件

onReceiveCustomMessage	收到自定义消息事件
onDeviceChanged	本地设备添加事件

TUIRoomEngine 基本类型定义

函数列表	描述
onSuccess	接口调用成功
onError	接口调用失败
onPlaying	视频播放成功
onLoading	视频正在加载。
onAccepted	请求被接受
onRejected	请求被拒绝
onCancelled	请求被取消
onTimeout	请求超时

TUIRoomEngine

最近更新时间：2023-09-28 15:21:02

Copyright (c) 2021 Tencent. All rights reserved.

Module: TUIRoomEngine @ TUIKitEngine
Function: TUIRoomEngine 主功能接口
Version: 1.5

TUIRoomEngine

TUIRoomEngine

函数列表	描述
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作
logout	退出登录接口，会有主动离开房间操作、销毁资源
setSelfInfo	设置本地用户名称和头像
getSelfInfo	获取本地用户登录的基本信息
setSelfInfo	设置本地用户信息
addObserver	设置事件回调
removeObserver	移除事件回调
createRoom	创建房间
destroyRoom	解散房间
enterRoom	进入房间
exitRoom	离开房间
connectOtherRoom	连接其他房间
disconnectOtherRoom	断开与其他房间的连接
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间名称（只有管理员或房主能够调用）
updateRoomSpeechModeByAdmin	设置房间管理模式（只有管理员或房主能够调用）
setLocalVideoView	设置本地用户视频渲染的视图控件
openLocalCamera	打开本地摄像头
closeLocalCamera	关闭本地摄像头

startPushLocalVideo	开始推送本地视频。默认开启
stopPushLocalVideo	停止推送本地视频
updateVideoQuality	更新本地视频编码质量设置
updateVideoQualityEx	设置视频编码器的编码参数
setVideoResolutionMode	设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）
startScreenSharing	开始屏幕分享。
stopScreenSharing	结束屏幕分享
getScreenSharingTargetList	获取分享对象列表
selectScreenSharingTarget	选择屏幕分享对象
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio	暂停发布本地的音频流
unmuteLocalAudio	恢复发布本地的音频流
setRemoteVideoView	设置远端用户视频渲染的视图控件
startPlayRemoteVideo	开始播放远端用户视频
stopPlayRemoteVideo	停止播放远端用户视频
muteRemoteAudioStream	将远端用户禁音
getUserList	获取房间内的成员列表
getUserInfo	获取成员信息
changeUserRole	修改用户角色（只有管理员或房主能够调用）
kickRemoteUserOutOfRoom	将远端用户踢出房间（只有管理员或房主能够调用）
disableDeviceForAllUserByAdmin	全体用户媒体设备管理
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或房主能够调用）
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或房主能够调用）
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）
getSeatList	获取麦位列表
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）
takeSeat	上麦（上麦发言模式下，需要申请）

版权所有：腾讯云计算（北京）有限责任公司

leaveSeat	下麦
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦
kickUserOffSeatByAdmin	主持人/管理员 将用户下麦
sendTextMessage	发送本文消息
sendCustomMessage	发送自定义消息
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）
cancelRequest	取消请求
responseRemoteRequest	回复请求
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）
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别
callExperimentalAPI	调用实验性接口

login

login

```
void login (int sdkAppId
            const char* userId
            const char* userSig
            TUICallback* callback)
```

登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
sdkAppId	它是腾讯云用于区分客户的唯一标识，进入腾讯云实时音视频 控制台 创建应用，即可看到SDKAppId
userId	用户ID，用于区分不同用户
userSig	用户签名，用于腾讯云流量的认证

logout

logout

void logout （TUICallback * callback)

退出登录接口，会有主动离开房间操作、销毁资源

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

setSelfInfo

setSelfInfo

void setSelfInfo (const char* userName
const char* avatarURL
TUICallback * callback)

设置本地用户名称和头像

参数	描述
avatarURL	用户头像URL地址
callback	调用接口的回调，用于通知接口调用的成功或者失败
userName	用户名称

getSelfInfo

getSelfInfo

获取本地用户登录的基本信息

返回值说明：

TUILoginUserInfo：用户登录信息

setSelfInfo

setSelfInfo

void setSelfInfo (const TUILoginUserInfo & userInfo

TUICallback * callback)

设置本地用户信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userInfo	本地用户信息

addObserver

addObserver

void addObserver (TUIRoomObserver * observer)

设置事件回调

您可以通过 TUIRoomObserver 获得各类事件通知（比如：错误码，远端用户进房，音视频状态参数等）

参数	描述
observer	监听的实例

removeObserver

removeObserver

void removeObserver (TUIRoomObserver * observer)

移除事件回调

参数	描述
observer	待移除的监听回调实例

createRoom

createRoom

void createRoom (const TUIRoomInfo & roomInfo

TUICallback * callback)

创建房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomInfo	房间信息，可以初始化房间的一些设置

destroyRoom

destroyRoom

```
void
destroyRoom
```
(TUICallback* callback)

解散房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

enterRoom

enterRoom

```
void
enterRoom
```
(const char* roomId

TUIValueCallback<TUIRoomInfo>* callback)

进入房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomId	房间ID

exitRoom

exitRoom

```
void
exitRoom
```
(bool syncWaiting

TUICallback* callback)

离开房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
syncWaiting	是否同步等待接口返回

connectOtherRoom

connectOtherRoom

```
TUIRequest
connectOtherRoom
(const char* roomId
const char* userId
int timeout
TUIRequestCallback* callback)
```

连接其他房间

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomId	房间ID
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

注意
用于直播场景下的申请跨房连麦

返回值说明：

TUIRequest 请求体

disconnectOtherRoom

disconnectOtherRoom

```
void
disconnectOtherRoom
(TUICallback* callback)
```

断开与其他房间的连接

参数	描述

callback	调用接口的回调，用于通知接口调用的成功或者失败

注意
用于直播场景下的断开跨房连麦

fetchRoomInfo

fetchRoomInfo

void fetchRoomInfo (TUIValueCallback<TUIRoomInfo>* callback)

获取房间信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

updateRoomNameByAdmin

updateRoomNameByAdmin

void updateRoomNameByAdmin (const char* roomName
TUICallback* callback)

更新房间名称（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
roomName	房间名称

updateRoomSpeechModeByAdmin

updateRoomSpeechModeByAdmin

void updateRoomSpeechModeByAdmin (TUISpeechMode mode
TUICallback* callback)

设置房间管理模式（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
mode	kFreeToSpeak: 自由发言模式, 用户可以自由开启麦克风和扬声器; kApplyToSpeak: 申请发言模式，用户requestOpenLocalMicrophone 或 requestOpenLocalCamera 向房主或管理员申请后，方可打开麦克风和摄像头开始发言 kkSpeakAfterTakingSeat: 上麦发言模式。KConference房间内，所有人在发言前，必须takeSeat，才能进行麦克风和摄像头操作。

setLocalVideoView

setLocalVideoView

```
void setLocalVideoView (TUIVideoStreamType streamType
                        const TUIVideoView& view)
```

设置本地用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义
view	视频渲染视图

openLocalCamera

openLocalCamera

```
void openLocalCamera (bool isFront
                      TUIVideoQuality quality
                      TUICallback * callback)
```

打开本地摄像头

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isFront	true: 前置 false: 后置。该参数只在移动端生效

closeLocalCamera

closeLocalCamera

关闭本地摄像头

startPushLocalVideo

startPushLocalVideo

开始推送本地视频。默认开启

stopPushLocalVideo

stopPushLocalVideo

停止推送本地视频

updateVideoQuality

updateVideoQuality

void updateVideoQuality (TUIVideoQuality quality)

更新本地视频编码质量设置

updateVideoQualityEx

updateVideoQualityEx

void updateVideoQualityEx (TUIVideoStreamType streamType
const TUIVideoEncoderParams & params)

设置视频编码器的编码参数

参数	描述
params	视频编码器的编码参数，定义可参考 TUIVideoEncoderParams 的定义
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义

setVideoResolutionMode

setVideoResolutionMode

```
void setVideoResolutionMode (TUIVideoStreamType streamType
                             TUIResolutionMode resolutionMode)
```

设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）

参数	描述
resolutionMode	分辨率模式，定义可参考 TUIResolutionMode 的定义
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义

startScreenSharing

startScreenSharing

```
void startScreenSharing (const TUISourceId& sourceId
                         TUICallback* callback)
```

开始屏幕分享。

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
sourceId	屏幕分享窗口或屏幕的句柄，可以调用GetScreenSharingTargetList获取，移动端填空即可。

stopScreenSharing

stopScreenSharing

结束屏幕分享

getScreenSharingTargetList

getScreenSharingTargetList

```
void getScreenSharingTargetList (TUIListCallback<TUIShareTarget>* list)
```

获取分享对象列表

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

参数	描述
list	调用接口的回调，用于通知接口调用的成功或者失败

selectScreenSharingTarget

selectScreenSharingTarget

void selectScreenSharingTarget (const TUISourceId& sourceId)

选择屏幕分享对象

当您通过 getScreenSharingTargetList 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

参数	描述
sourceId	屏幕分享窗口或屏幕的句柄，可以调用 getScreenSharingTargetList 获取

openLocalMicrophone

openLocalMicrophone

void openLocalMicrophone (TUIAudioQuality quality

TUICallback* callback)

打开本地麦克风

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

closeLocalMicrophone

closeLocalMicrophone

关闭本地麦克风

muteLocalAudio

muteLocalAudio

暂停发布本地的音频流

unmuteLocalAudio

unmuteLocalAudio

```
void
unmuteLocalAudio        (TUICallback* callback)
```

恢复发布本地的音频流

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

setRemoteVideoView

setRemoteVideoView

```
void
setRemoteVideoView      (const char* userId
                         TUIVideoStreamType streamType
                         const TUIVideoView& view)
```

设置远端用户视频渲染的视图控件

参数	描述
streamType	视频流的类型，定义可参考 TUIVideoStreamType 的定义
userId	远端用户ID
view	视频渲染视图

startPlayRemoteVideo

startPlayRemoteVideo

```
void startPlayRemoteVideo   (const char* userId
                             TUIVideoStreamType streamType
                             TUIPlayCallback* callback)
```

开始播放远端用户视频

参数	描述
callback	调用接口的回调，用于通知播放时的状态回调
streamType	视频流的类型 详细定义可以参考 TUIVideoStreamType 的定义
userId	用户ID

stopPlayRemoteVideo

stopPlayRemoteVideo

```
void
stopPlayRemoteVideo        (const char* userId
                            TUIVideoStreamType streamType)
```

停止播放远端用户视频

参数	描述
streamType	视频流的类型 详细定义可以参考 TUIVideoStreamType 的定义
userId	用户ID

muteRemoteAudioStream

muteRemoteAudioStream

```
void
muteRemoteAudioStream      (const char * userId
                            bool isMute)
```

将远端用户禁音

参数	描述
isMute	是否禁音
userId	用户ID

getUserList

getUserList

```
void getUserList (uint64_t nextSequence
                  TUIValueCallback<TUIUserListResult>* callback)
```

获取房间内的成员列表

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败。成功回调中包含 TUIUserInfo list
nextSequence	分页拉取标志，第一次拉取填0，回调成功 如果callback返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0

getUserInfo

getUserInfo

```
void getUserInfo (const char* userId
                  TUIValueCallback<TUIUserInfo>* callback)
```

获取成员信息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userId	用户ID

changeUserRole

changeUserRole

```
void changeUserRole (const char* userId
                     TUIRole role
                     TUICallback* callback)
```

修改用户角色（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
role	用户角色 详细定义可以参考 TUIRole 的定义

userId	用户ID

kickRemoteUserOutOfRoom

kickRemoteUserOutOfRoom

```
void kickRemoteUserOutOfRoom (const char* userId
                              TUICallback* callback)
```

将远端用户踢出房间（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
userId	用户ID

disableDeviceForAllUserByAdmin

disableDeviceForAllUserByAdmin

```
void disableDeviceForAllUserByAdmin (TUIMediaDevice device
                                     bool isDisable
                                     TUICallback* callback)
```

全体用户媒体设备管理

参数	描述
callback	操作回调
device	设备。详细定义参考:TUIMediaDevice
isDisable	否禁用

openRemoteDeviceByAdmin

openRemoteDeviceByAdmin

```
TUIRequest openRemoteDeviceByAdmin (const char* userId
                                    TUIMediaDevice device
```

```
int timeout
TUIRequestCallback* callback)
```

请求远端用户打开媒体设备（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知请求状态的回调，详细定义参考：TUIRequestCallback
device	媒体设备。详细定义参考:TUIMediaDevice
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

closeRemoteDeviceByAdmin

closeRemoteDeviceByAdmin

```
void closeRemoteDeviceByAdmin (const char* userId
                               TUIMediaDevice device
                               TUICallback* callback)
```

关闭远端用户媒体设备（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
device	媒体设备。详细定义参考:TUIMediaDevice
userId	用户ID

applyToAdminToOpenLocalDevice

applyToAdminToOpenLocalDevice

```
TUIRequest applyToAdminToOpenLocalDevice (TUIMediaDevice device
                                          int timeout
                                          TUIRequestCallback* callback)
```

请求打开本地媒体设备（普通用户可用）

参数	描述
callback	调用接口的回调，用于通知请求的回调状态，详细定义参考：TUIRequestCallback
device	用户ID device 媒体设备。详细定义参考:TUIMediaDevice
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

返回值说明：

TUIRequest 请求体

setMaxSeatCount

setMaxSeatCount

```
void setMaxSeatCount (uint32_t maxSeatCount
                      TUICallback * callback)
```

设置最大麦位数（仅支持进房前和创建房间时设置）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
maxSeatCount	最大麦位数

getSeatList

getSeatList

```
void getSeatList (TUIListCallback<TUISeatInfo>* callback)
```

获取麦位列表

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败，成功回调中会包含 TUISeatInfo list信息

lockSeatByAdmin

lockSeatByAdmin

```
void (int seatIndex
```

lockSeatByAdmin

const TUISeatLockParams & lockParams

TUICallback * callback)

锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
lockParams	锁麦参数。详情参考:TUIRoomDefine.SeatLockParams
seatIndex	麦位编号

takeSeat

takeSeat

TUIRequest takeSeat (int seatIndex

int timeout

TUIRequestCallback * callback)

上麦（上麦发言模式下，需要申请）

参数	描述
callback	调用接口的回调，用于通知请求的回调状态，详细定义参考: TUIRequestCallback
seatIndex	麦位编号
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调

注意

开启上麦发言模式时，需要向主持人或管理员发起申请才允许上麦。

开启自由发言模式，直播场景可以自由上麦，上麦后开麦发言，会议场景无需调用该接口，即可开麦发言。

返回值说明：

TUIRequest 请求体

leaveSeat

leaveSeat

void leaveSeat (TUICallback * callback)

下麦

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败

takeUserOnSeatByAdmin

takeUserOnSeatByAdmin

```
TUIRequest takeUserOnSeatByAdmin(int seatIndex
                                 const char* userId
                                 int timeout
                                 TUIRequestCallback* callback)
```

主持人/管理员 邀请用户上麦

参数	描述
callback	调用接口的回调，用于通知请求的回调状态，详细定义参考：TUIRequestCallback
seatIndex	麦位编号。会议场景无需关心，填0即可。
timeout	超时时间，单位秒，如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
userId	用户ID

返回值说明：

TUIRequest 请求体

kickUserOffSeatByAdmin

kickUserOffSeatByAdmin

```
void kickUserOffSeatByAdmin(int seatIndex
                            const char* userId
                            TUICallback* callback)
```

主持人/管理员 将用户下麦

参数	描述

callback	调用接口的回调，用于通知接口调用的成功或者失败
seatIndex	麦位编号
userId	用户ID

sendTextMessage

sendTextMessage

void sendTextMessage (const char* message

TUICallback * callback)

发送本文消息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
message	消息内容

sendCustomMessage

sendCustomMessage

void sendCustomMessage (const char* message

TUICallback * callback)

发送自定义消息

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
message	消息内容

disableSendingMessageByAdmin

disableSendingMessageByAdmin

void disableSendingMessageByAdmin (const char* userId

bool isDisable

TUICallback* callback)

禁用远端用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isDisable	是否禁用
userId	用户ID

disableSendingMessageForAllUser

disableSendingMessageForAllUser

void disableSendingMessageForAllUser (bool isDisable
TUICallback* callback)

禁用所有用户的发送文本消息能力（只有管理员或房主能够调用）

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
isDisable	是否禁用

cancelRequest

cancelRequest

void cancelRequest (const char* requestId
TUICallback* callback)

取消请求

参数	描述
callback	调用接口的回调，用于通知接口调用的成功或者失败
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
可以使用此接口来取消已发出的请求

responseRemoteRequest

responseRemoteRequest

```
void responseRemoteRequest(const char* requestId
                           bool agree
                           TUICallback* callback)
```

回复请求

参数	描述
agree	是否同意 true: 同意请求, false: 拒绝请求
callback	调用接口的回调，用于通知接口调用的成功或者失败
requestId	请求ID(发送请求的接口返回或者OnRequestReceived事件通知)

注意
在收到信令请求时，可以使用此接口来回复接收到的请求

getDevicesList

getDevicesList

```
ITXDeviceCollection* getDevicesList(liteav::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 (liteav::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。

setBeautyLevel

setBeautyLevel

```
void setBeautyLevel (int beautyStyle
                     uint32_t 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。
beautyStyle	美颜风格，TXBeautyStyleSmooth：光滑；TXBeautyStyleNature：自然；TXBeautyStylePitu：优图。

setWhitenessLevel

setWhitenessLevel

```
void setWhitenessLevel (uint32_t 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。

callExperimentalAPI

callExperimentalAPI

void callExperimentalAPI (const char* jsonStr)

调用实验性接口

参数	描述
jsonStr	接口信息

TUIRoomObserver

最近更新时间：2023-09-14 16:46:13

Copyright (c) 2021 Tencent. All rights reserved.
Module: TUIRoomObserver @ TUIKitEngine
Function: TUIRoomEngine的事件回调接口

TUIRoomObserver

TUIRoomObserver

函数列表	描述
onError	错误事件回调
onKickedOffLine	用户被踢下线
onUserSigExpired	用户凭证超时事件
onRoomNameChanged	房间名称更改事件
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件
onSendMessageForAllUserDisableChanged	房间内用户发送文本消息被禁用事件
onRoomDismissed	房间被解散事件
onKickedOutOfRoom	被踢出房间事件
onRoomSpeechModeChanged	房间麦控模式发生变化
onRemoteUserEnterRoom	远端用户进房事件
onRemoteUserLeaveRoom	远端用户离开房间事件
onUserRoleChanged	用户角色发生变化事件
onUserVideoStateChanged	用户视频状态发生变化事件
onUserAudioStateChanged	用户音频状态发生变化事件
onUserVoiceVolumeChanged	用户音量变化事件
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件
onUserNetworkQualityChanged	用户网络状态变化事件
onUserScreenCaptureStopped	屏幕分享结束
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）

onSeatListChanged	麦位列表发生变化事件
onKickedOffSeat	收到用户被踢下麦事件
onRequestReceived	收到请求消息事件
onRequestCancelled	收到请求被取消的事件
onReceiveTextMessage	收到普通文本消息事件
onReceiveCustomMessage	收到自定义消息事件
onDeviceChanged	本地设备添加事件

onError

onError

```
void onError (TUIError errorCode
              const char* message)
```

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。

参数	描述
errorCode	错误码，请参考：TUIError
message	错误信息

onKickedOffLine

onKickedOffLine

```
void onKickedOffLine (const char* message)
```

用户被踢下线

参数	描述
message	被踢下线的描述

onUserSigExpired

onUserSigExpired

用户凭证超时事件

onRoomNameChanged

onRoomNameChanged

```
void onRoomNameChanged (const char* roomId
                        const char* roomName)
```

房间名称更改事件

参数	描述
roomId	房间ID
roomName	房间名称

onAllUserMicrophoneDisableChanged

onAllUserMicrophoneDisableChanged

```
void onAllUserMicrophoneDisableChanged (const char* roomId
                                        bool isDisable)
```

房间内所有用户麦克风被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onAllUserCameraDisableChanged

onAllUserCameraDisableChanged

```
void onAllUserCameraDisableChanged (const char* roomId
                                    bool isDisable)
```

房间内所有用户摄像头被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onSendMessageForAllUserDisableChanged

onSendMessageForAllUserDisableChanged

```
void
onSendMessageForAllUserDisableChanged
(const char* roomId
bool isDisable)
```

房间内用户发送文本消息被禁用事件

参数	描述
isDisable	是否被禁用
roomId	房间ID

onRoomDismissed

onRoomDismissed

```
void
onRoomDismissed
(const char* roomId)
```

房间被解散事件

参数	描述
roomId	房间ID

onKickedOutOfRoom

onKickedOutOfRoom

```
void
onKickedOutOfRoom
(const char* roomId
TUIKickedOutOfRoomReason reason
```

const char* message)

被踢出房间事件

参数	描述
message	被踢出的描述
reason	被踢出原因
roomId	房间ID

onRoomSpeechModeChanged

onRoomSpeechModeChanged

void onRoomSpeechModeChanged (const char* roomId

TUISpeechMode speechMode)

房间麦控模式发生变化

参数	描述
mode	房间模式
roomId	房间ID

onRemoteUserEnterRoom

onRemoteUserEnterRoom

void onRemoteUserEnterRoom (const char* roomId

const TUIUserInfo & userInfo)

远端用户进房事件

参数	描述
roomId	房间ID
userInfo	用户信息

onRemoteUserLeaveRoom

onRemoteUserLeaveRoom

```
void
onRemoteUserLeaveRoom
(const char* roomId
const TUIUserInfo & userInfo)
```

远端用户离开房间事件

参数	描述
roomId	房间ID
userInfo	用户信息

onUserRoleChanged

onUserRoleChanged

```
void
onUserRoleChanged
(const char* userId
TUIRole role)
```

用户角色发生变化事件

参数	描述
userId	用户ID
userRole	用户角色 可参考 TUIRole 枚举定义

onUserVideoStateChanged

onUserVideoStateChanged

```
void
onUserVideoStateChanged
(const char* userId
TUIVideoStreamType streamType
bool hasVideo
TUIChangeReason reason)
```

用户视频状态发生变化事件

参数	描述

hasVideo	是否有视频流
reason	视频流发生变化原因 kChangedBySelf: 自己切换 kChangedByAdmin: 被管理员切换
streamType	视频流类型
userId	用户ID

onUserAudioStateChanged

onUserAudioStateChanged

void onUserAudioStateChanged (const char* userId

bool hasVideo

TUIChangeReason reason)

用户音频状态发生变化事件

参数	描述
hasAudio	是否有音频流
reason	视频流发生变化原因 kChangedBySelf: 自己切换 kChangedByAdmin: 被管理员切换
userId	用户ID

onUserVoiceVolumeChanged

onUserVoiceVolumeChanged

void onUserVoiceVolumeChanged (TUIMap<const char*, int>* volumeMap)

用户音量变化事件

参数	描述
volumeMap	用户音量字典 key: userId, value: 用于承载所有正在说话的用户的音量大小，取值范围 0 – 100。

onSendMessageForUserDisableChanged

onSendMessageForUserDisableChanged

void onSendMessageForUserDisableChanged (const char* roomId

const char* userId

bool isDisable)

用户文本消息发送能力发生变化事件

参数	描述
isDisable	是否被禁止发送文本消息 true: 用户被禁止发送消息 false: 用户被解除禁止，可以发送消息
userId	用户ID

onUserNetworkQualityChanged

onUserNetworkQualityChanged

void onUserNetworkQualityChanged (TUIList<TUINetwork>* networkList)

用户网络状态变化事件

参数	描述
networkList	用户网络状态数组，可参考 TUINetwork 对象

onUserScreenCaptureStopped

onUserScreenCaptureStopped

void onUserScreenCaptureStopped (int reason)

屏幕分享结束

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）

onRoomMaxSeatCountChanged

onRoomMaxSeatCountChanged

void onRoomMaxSeatCountChanged (const char *roomId

int maxSeatCount)

房间内最大麦位数发生变化事件（仅在会议类型房间生效）

参数	描述
maxSeatCount	房间内最大麦位数量
roomId	房间ID

onSeatListChanged

onSeatListChanged

void onSeatListChanged (TUIList<TUISeatInfo>* seatList
TUIList<TUISeatInfo>* seatedList
TUIList<TUISeatInfo>* leftList)

麦位列表发生变化事件

参数	描述
leftList	新下麦的用户列表
seatList	目前麦上最新的用户列表，包含新上麦的用户
seatedList	新上麦的用户列表

onKickedOffSeat

onKickedOffSeat

void onKickedOffSeat (const char* userId)

收到用户被踢下麦事件

参数	描述
userId	操作踢人的（主持人/管理员）用户id

onRequestReceived

onRequestReceived

void onRequestReceived	(const TUIRequest* request)

收到请求消息事件

参数	描述
request	请求内容，可参考 TUIRequest 对象

onRequestCancelled

onRequestCancelled

void onRequestCancelled	(const char* requestId const char* userId)

收到请求被取消的事件

参数	描述
requestId	请求ID
userId	取消信令的用户ID

onReceiveTextMessage

onReceiveTextMessage

void onReceiveTextMessage	(const char* roomId const TUIMessage& message)

收到普通文本消息事件

参数	描述
message	消息内容, 请参考: TUIMessage 定义
roomId	房间ID

onReceiveCustomMessage

onReceiveCustomMessage

void onReceiveCustomMessage	(const char* roomId const TUIMessage & message)

收到自定义消息事件

参数	描述
message	消息内容, 请参考: TUIMessage 定义
roomId	房间ID

onDeviceChanged

onDeviceChanged

void onDeviceChanged	(const char* deviceId liteav::TXMediaDeviceType type liteav::TXMediaDeviceState state)

本地设备添加事件

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。

注意
当本地设备（包括摄像头、麦克风以及扬声器）添加时，SDK 便会抛出此事件回调

TUICommonDefine

最近更新时间：2023-09-14 16:46:13

TUICommonDefine

结构体类型

函数列表	描述
TUINetwork	网络质量信息
TUIMessage	消息
TUIImageBuffer	TUIImageBuffer定义

TUINetworkInfo

TUINetworkInfo

网络质量信息

枚举类型	描述
delay	网络延迟，单位 ms
downLoss	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
quality	网络质量
upLoss	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
userId	用户ID

TUIMessage

TUIMessage

消息

枚举类型	描述
avatarUrl	消息发送者头像

message	消息文本
messageId	消息 ID
timestamp	消息时间
userId	消息发送者
userName	消息发送者昵称

TUIMessage

TUIMessage

TUIImageBuffer定义

枚举类型	描述
buffer	图片数据缓存地址
height	高度
length	长度
width	宽度

TUIRoomDefine

最近更新时间：2023-09-28 15:21:02

TUIRoomDefine

TUICallback

函数列表	描述
onSuccess	接口调用成功
onError	接口调用失败

TUIPlayCallback

函数列表	描述
onPlaying	视频播放成功
onLoading	视频正在加载。
onError	接口调用失败

TUIRequestCallback

函数列表	描述
onAccepted	请求被接受
onRejected	请求被拒绝
onCancelled	请求被取消
onTimeout	请求超时
onError	请求错误。

结构体类型

函数列表	描述
TUIRoomInfo	房间信息
TUILoginUserInfo	用户登录信息
TUIUserInfo	房间内用户信息
TUIRoomVideoEncoderParams	视频编码参数

TUISeatInfo	房间内座位信息
TUISeatLockParams	锁定麦位操作参数
TUIUserVoiceVolume	房间内用户音量
TUIRequest	信令请求
TUIUserListResult	TUIUserListResult 定义

onSuccess

onSuccess

接口调用成功

onError

onError

```
void onError (const TUIError code
              const char* message)
```

接口调用失败

参数	描述
code	错误码
message	错误信息

onPlaying

onPlaying

```
void onPlaying (const char* userId)
```

视频播放成功

参数	描述
userId	用户ID

onLoading

onLoading

```
void onLoading (const char* userId)
```

视频正在加载。

参数	描述
userId	用户ID

onError

onError

```
void onError (const char* userId
              const TUIError code
              const char* message)
```

接口调用失败

参数	描述
code	错误码
message	错误信息
userId	用户ID

onAccepted

onAccepted

```
void
onAccepted (const char* requestId
            const char* userId)
```

请求被接受

参数	描述
requestId	请求ID
userId	用户ID

onRejected

onRejected

```
void onRejected (const char* requestId
                 const char* userId
                 const char* message)
```

请求被拒绝

参数	描述
message	错误信息
requestId	请求ID
userId	用户ID

onCancelled

onCancelled

```
void onCancelled (const char* requestId
                  const char* userId)
```

请求被取消

参数	描述
requestId	请求ID
userId	用户ID

onTimeout

onTimeout

```
void
onTimeout (const char* requestId
           const char* userId)
```

请求超时

参数	描述

requestId	请求ID
userId	用户ID

onError

onError

```
void onError (const char* requestId
              const char* userId
              const TUIError code
              const char* message)
```

请求错误。

参数	描述
code	错误码
message	错误信息
requestId	请求ID
userId	用户ID

TUIRoomInfo

TUIRoomInfo

房间信息

TUIRoomEngine 只支持字符串房间ID

枚举类型	描述
createTime	房间创建时间（只读）
isCameraDisableForAllUser	是否禁止打开摄像头（创建房间可选参数），默认值：false。
isMessageDisableForAllUser	是否禁止发送消息（创建房间可选参数），默认值：false。
isMicrophoneDisableForAllUser	是否禁止打开麦克风（创建房间可选参数），默认值：false。
maxSeatCount	最大麦位数
memberCount	房间内成员数量（只读）
name	房间名称（创建房间可选参数，默认房间ID）

ownerId	主持人ID: 默认为房间创建者（只读）
roomId	房间ID(创建房间必填参数)
roomType	房间类型（创建房间可选参数，默认Group类型），请参考：TUIRoomType。
speechMode	房间麦控模式

TUIUserInfo

TUIUserInfo

用户登录信息

枚举类型	描述
avatarUrl	用户头像URL
userId	用户ID
userName	用户名称

TUIUserInfo

TUIUserInfo

房间内用户信息

枚举类型	描述
avatarUrl	用户头像URL
hasAudioStream	是否有音频流，默认值：false。
hasScreenStream	是否有屏幕分享流，默认值：false。
hasVideoStream	是否有视频流，默认值：false。
userId	用户ID
userName	用户名称
userRole	用户角色类型, 请参考：TUIRole。

TUIRoomVideoEncoderParams

TUIRoomVideoEncoderParams

视频编码参数

枚举类型	描述
bitrate	目标视频码率
fps	视频采集帧率
resolutionMode	分辨率模式, 请参考：TUIResolutionMode。
videoResolution	视频质量, 请参考：TUIVideoQuality。

TUISeatInfo

TUISeatInfo

房间内座位信息

枚举类型	描述
index	麦位序号
isAudioLocked	麦位是否被禁止打开麦克风，默认值：false。
isLocked	麦位是否被锁定，默认值：false。
isVideoLocked	麦位是否被禁止打开摄像头，默认值：false。
userId	用户ID

TUISeatLockParams

TUISeatLockParams

锁定麦位操作参数

枚举类型	描述
lockAudio	锁定麦位麦克风，默认值：false。
lockSeat	锁定麦位，默认值：false。
lockVideo	锁定麦位摄像头，默认值：false。

TUIUserVoiceVolume

TUIUserVoiceVolume

版权所有：腾讯云计算（北京）有限责任公司

房间内用户音量

枚举类型	描述
userId	用户ID
volume	音量 用于承载所有正在说话的用户的音量大小，取值范围 0 – 100

TUIRequest

TUIRequest

信令请求

枚举类型	描述
content	信令内容
requestAction	请求类型
requestId	请求ID
timestamp	时间戳
userId	用户ID

TUIUserListResult

TUIUserListResult

TUIUserListResult定义

枚举类型	描述
nextSequence	获取下一批成员列表的序号
userInfoList	成员信息列表

Web
API 概览

最近更新时间：2024-03-04 14:32:51

TUIRoomKit（含 UI 组件）

TUIRoomKit 接口列表

API	描述
init	初始化 TUIRoomKit 组件
createRoom	主持人创建房间
enterRoom	普通成员加入房间

TUIRoomKit 事件类型

事件	描述
onCreateRoom	创建房间回调
onEnterRoom	进入房间回调
onDestroyRoom	销毁房间回调
onExitRoom	普通成员退出房间回调
onKickedOutOfRoom	普通成员被主持人踢出房间通知
onKickedOffLine	用户账号在其他地方登录，被踢下线
onUserSigExpired	用户 userSig 过期，请重新获取 userSig

TUIRoomEngine (无 UI 接口)

TUIRoomEngine 接口列表

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.init 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.init 方法执行成功后执行。**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

版权所有：腾讯云计算（北京）有限责任公司

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用），该接口自 v2.0.0 版本废弃
updateRoomSeatModeByAdmin	更新房间上麦模式（仅群主或者管理员可以调用）
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集
closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateVideoQuality	设置本地视频的参数
updateAudioQuality	设置本地音频的参数
startScreenSharing	开始屏幕共享
stopScreenSharing	停止屏幕共享
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
startPushLocalAudio	开始向远端推本地音频流
muteLocalAudio	停止向远端推本地音频流
unmuteLocalAudio	开始向远端推本地音频流
stopPushLocalAudio	停止向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域

版权所有：腾讯云计算（北京）有限责任公司

startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流
muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备
cancelRequest	取消已经发出的请求
responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户踢出房间
disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 麦位管理 API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人踢下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送 API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

版权所有：腾讯云计算（北京）有限责任公司

roomEngine 设备管理 API

API	描述
getCameraDevicesList	获取摄像头设备列表
getMicDevicesList	获取麦克风设备列表
getSpeakerDevicesList	获取扬声器设备列表
setCurrentCameraDevice	设置要使用的摄像头设备
setCurrentMicDevice	设置要使用的麦克风设备
setCurrentSpeakerDevice	设置要使用的扬声器设备
getCurrentCameraDevice	获取当前正在使用的摄像头设备
getCurrentMicDevice	获取当前正在使用的麦克风设备
getCurrentSpeakerDevice	获取当前正在使用的扬声器设备
startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止摄像头测试

roomEngine 事件监听 API

API	描述
on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

TUIRoomEngine 事件类型

TUIRoomEvent 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	踢出房间事件
TUIRoomEvents.onKickedOffLine	当前用户被踢下线事件
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomNameChanged	房间名称修改事件

版权所有：腾讯云计算（北京）有限责任公司

~~TUIRoomEvents.onRoomSpeechModeChanged~~	房间发言模式修改事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onAllUserCameraDisableChanged	所有成员摄像头使用权限改变事件
TUIRoomEvents.onAllUserMicrophoneDisableChanged	所有成员麦克风使用权限改变事件
TUIRoomEvents.onSendMessageForAllUserDisableChanged	所有成员发送消息状态改变事件
TUIRoomEvents.onRoomMaxSeatCountChanged	房间最大麦位数修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChanged	用户音频状态改变事件
TUIRoomEvents.onSendMessageForUserDisableChanged	用户发送消息状态事件
TUIRoomEvents.onUserVoiceVolumeChanged	用户音量改变事件
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件
TUIRoomEvents.onSeatListChanged	麦位列表改变事件
TUIRoomEvents.onKickedOffSeat	用户被踢下麦事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
~~TUIRoomEvents.onReceiveCustomMessage~~	接收自定义消息事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onDeviceChange	设备变更事件
TUIRoomEvents.onUserScreenCaptureStopped	屏幕分享停止事件 当用户使用浏览器自带的**停止分享**按钮结束屏幕共享时，该用户会收到 'onUserScreenCaptureStopped' 事件用来修改屏幕共享状态。

TUIRoomKit

最近更新时间：2024-02-20 17:41:31

API 简介

TUIRoomKit API 是多人视频会议组件的**含 UI 接口**，使用 TUIRoomKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，如果您想要体验并调试视频会议效果，请阅读 Demo 快速跑通，如果您想把我们的功能直接嵌入到您的项目中，请阅读 快速接入(TUIRoomKit)。

API 概览

在页面中引用 TUIRoomKit 组件。例如：在 App.vue 组件中引入 TUIRoomKit 组件。

- TUIRoomKit 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
- 主持人及普通成员可通过 init 方法向 TUIRoomKit 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
  <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
const TUIRoomRef = ref();

onMounted(async () => {
  // 初始化 TUIRoom 组件
  // 主持人在创建房间前需要先初始化 TUIRoom 组件
  // 普通成员在进入房间前需要先初始化 TUIRoom 组件
  await TUIRoomRef.value.init({
    // 获取 sdkAppId 请您参考 步骤一
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig, 注意 userSig 与
userId 为一一对应关系
    userSig: '',
    // 用户在您业务中使用的昵称
    userName: '',
    // 用户在您业务中使用的头像链接
    avatarUrl: '',
    // 用户在您业务中需要的皮肤主题颜色及是否支持切换皮肤主题
    theme: {
      defaultTheme:'black',
      isSupportSwitchTheme: true
    }
  })
  // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
  await handleCreateRoom();
})
```

```
// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认为'FreeToSpeak'，注意目前仅支持自由发言模式
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
  const roomMode = 'FreeToSpeak';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.createRoom({ roomId, roomName: roomId, roomMode, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.createRoom error: ' + error.message);
  }
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.enterRoom({ roomId, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.enterRoom error: ' + error.message);
  }
}
</script>

<style lang="scss">
html, body {
    width: 100%;
    height: 100%;
    margin: 0;
}

#app {
  width: 100%;
  height: 100%;
  * {
    box-sizing: border-box;
  }
}
</style>
```

API 详情

TUIRoomkit 接口

init

初始化 TUIRoomKit 数据，任何使用 TUIRoomKIt 的用户都需要调用 init 方法。

```
TUIRoomRef.value.init(roomData);
```

参数如下表所示：

参数	类型	含义
roomData	object	房间初始化参数
roomData.sdkAppId	number	客户的 SDKAppId
roomData.userId	string	用户的唯一 ID
roomData.userSig	string	用户的 UserSig
roomData.userName	string	用户的昵称
roomData.avatarUrl	string	用户的头像地址
roomData.theme	object	界面的主题风格
roomData.theme.defaultTheme	string	默认主题皮肤颜色
roomData.theme.isSupportSwitchTheme	boolean	是否支持切换主题

createRoom

主持人创建房间。

```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomMode	string	房间模式，FreeToSpeak（自由发言模式）和 SpeakAfterTakingSeat（上台发言模式），默认为 FreeToSpeak
roomParam	object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID

roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	string	非必填，默认扬声器设备 ID

enterRoom

普通成员加入房间。

```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomParam	object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	string	非必填，默认扬声器设备 ID

TUIRoomkit 事件

onCreateRoom

创建房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-create-room="onCreateRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onCreateRoom(info) {
    if (info.code === 0) {
      console.log('创建房间成功')
    }
  }
</script>
```

onEnterRoom

进入房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-enter-room="onEnterRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onEnterRoom(info) {
    if (info.code === 0) {
      console.log('进入房间成功')
    }
  }
</script>
```

onDestroyRoom

主持人销毁房间的回调。

```
<template>
  <room ref="TUIRoomRef" @on-destroy-room="onDestroyRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onDestroyRoom(info) {
    if (info.code === 0) {
      console.log('主持人销毁成功')
    }
  }
</script>
```

onExitRoom

普通成员退出房间的回调。

```
<template>
  <room ref="TUIRoomRef" @on-exit-room="onExitRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onExitRoom(info) {
    if (info.code === 0) {
      console.log('普通成员退出房间成功')
    }
  }
</script>
```

onKickedOutOfRoom

普通成员被主持人踢出房间通知。

```
<template>
  <room ref="TUIRoomRef" @on-kicked-out-of-room="onKickedOutOfRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOutOfRoom({ roomId, message }) {
    console.log('普通成员被主持人踢出房间', roomId, message);
  }
</script>
```

onKickedOffLine

用户账号在其他地方登录，被踢下线。

```
<template>
  <room ref="TUIRoomRef" @on-kick-off-line="onKickedOffLine"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOffLine({ message }) {
    console.log('成员被踢下线', message);
  }
</script>
```

onUserSigExpired

用户 userSig 过期，请重新获取 userSig。

```
<template>
  <room ref="TUIRoomRef" @on-user-sig-expired="onUserSigExpired"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onUserSigExpired() {
    console.log('用户 userSig 过期，请重新生成 userSig.');
  }
</script>
```

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEngine

最近更新时间：2024-03-04 14:32:51

TUIRoomEngine 简介

TUIRoomEngine SDK 提供了房间管理、多人实时音视频互动、屏幕分享、成员管理、即时聊天等功能。
安装方式：

```
// 使用 npm
npm i @tencentcloud/tuiroom-engine-js --save

// 使用 pnpm
pnpm i @tencentcloud/tuiroom-engine-js --save

// 使用 yarn
yarn add @tencentcloud/tuiroom-engine-js
```

TUIRoomEngine API

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.login 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.login 方法执行成功后执行。**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用），该接口自 v2.0.0

	版本废弃
updateRoomSeatModeByAdmin	更新房间上麦模式（仅群主或者管理员可以调用）
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集
closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateVideoQuality	设置本地视频的分辨率
setVideoResolutionMode	设置本地视频流的分辨率模式
updateVideoQualityEx	设置本地视频的编码参数
updateAudioQuality	设置本地音频的参数
startScreenSharing	开始屏幕共享
stopScreenSharing	停止屏幕共享
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
muteLocalAudio	停止向远端推本地音频流
unmuteLocalAudio	开始向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域
startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流
muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备

版权所有：腾讯云计算（北京）有限责任公司

cancelRequest	取消已经发出的请求
responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户踢出房间
disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 麦位管理 API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人踢下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送 API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

roomEngine 设备管理 API

API	描述
getCameraDevicesList	获取摄像头设备列表
getMicDevicesList	获取麦克风设备列表
getSpeakerDevicesList	获取扬声器设备列表
setCurrentCameraDevice	设置要使用的摄像头设备
setCurrentMicDevice	设置要使用的麦克风设备
setCurrentSpeakerDevice	设置要使用的扬声器设备

版权所有：腾讯云计算（北京）有限责任公司

getCurrentCameraDevice	获取当前正在使用的摄像头设备
getCurrentMicDevice	获取当前正在使用的麦克风设备
getCurrentSpeakerDevice	获取当前正在使用的扬声器设备
startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止摄像头测试

roomEngine 事件监听 API

API	描述
on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

API 详情

once

监听 TUIRoomEngine 'ready' 事件。

```
TUIRoomEngine.once('ready', () => {
  const roomEngine = new TUIRoomEngine();

  await TUIRoomEngine.login({
    sdkAppId: 0,   // 填写您申请的 sdkAppId
    userId: '',    // 填写您业务对应的 userId
    userSig: '',   // 填写服务器或本地计算的 userSig
  });

  await roomEngine.createRoom({
    roomId: '12345',   // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
    name: 'Test Room',     // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
    roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
  });
});
```

login

> **说明：**
> v1.0.0 版本该接口命名为 TUIRoomEngine.init，v1.0.1 及其以上版本请使用 TUIRoomEngine.login 登录 TUIRoomEngine。

登录 TUIRoomEngine， 您必须在登录 TUIRoomEngine 之后才可以调用 TUIRoomEngine 及其实例的其他方法。

```
// 登录 TUIRoomEngine
await TUIRoomEngine.login({
 sdkAppId: 0,  // 填写您申请的 sdkAppId
 userId: '',  // 填写您业务对应的 userId
 userSig: '',  // 填写服务器或本地计算的 userSig
});
```

参数：

参数	类型	说明	默认值	含义
sdkAppId	number	必填	-	在 实时音视频控制台 单击 **应用管理** > **创建应用** 创建新应用之后，即可在 应用信息 中获取 sdkAppId 信息。
userId	string	必填	-	用户ID建议限制长度为32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。
userSig	string	必填	-	userSig 签名 计算 userSig 的方式请参考 UserSig 相关 。
tim	TIM	非必填	-	若您在接入 roomEngine 的同时想使用即时通信 SDK 的更多能力，您可以将创建的 tim 实例传入 TUIRoomEngine。tim 实例的创建方式请参考 TIM.create 。

Returns *Promise<void>*

setSelfInfo

设置当前用户基本信息（用户名、用户头像）。

```
// 设置当前用户用户名及用户头像
await TUIRoomEngine.setSelfInfo({
 userName: '',  // 填写您的新用户名
 avatarUrl: '',  // 填写您的新头像地址
});

// 设置当前用户用户名，用户头像及用户自定义消息
await TUIRoomEngine.setSelfInfo({
 userName: '',  // 填写您的新用户名
 avatarUrl: '',  // 填写您的新头像地址
 customInfo: {  // 填写用户自定义信息
  Tag_Profile_Custom_Test: 'value1',
  Tag_Profile_Custom_profile: 'value2',
 },
})
```

参数：

参数	类型	说明	默认值	含义
userName	string	必填	-	用户名

avatarUrl	string	必填	-	用户头像
customInfo	object	选填	-	用户自定义资料，该字段自 v1.3.0 版本支持

说明：
App 管理员可以在 即时通信 IM 控制台 > 用户自定义字段 新增自定义资料字段，用户自定义字段生效以后，用户可以调用 setSelfInfo 接口设置自定义资料。

Returns *Promise<void>*

getSelfInfo

获取当前用户基本信息（用户名、用户头像）。

```
// 获取当前用户用户名及用户头像
const loginUserInfo = await TUIRoomEngine.getSelfInfo();
```

Returns *Promise<TUILoginUserInfo > loginUserInfo*

logout

说明：
该接口自 v1.0.1 版本支持。

登出 TUIRoomEngine。

```
// 登出 TUIRoomEngine
await TUIRoomEngine.logout();
```

Returns *Promise<void>*

createRoom

主持人创建房间，调用 createRoom 的用户为房间的所有者。创建房间时可设置房间 ID、房间名称、房间类型、是否开启麦位控制、是否允许加入用户开启音视频、发送消息等功能。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({
  roomId: '12345',  // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
  roomName: 'Test Room',    // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
  roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
  isSeatEnabled: false, // 设置是否开启麦位控制
  isMicrophoneDisableForAllUser: false,  // 设置是否允许加入用户打开麦克风
  isCameraDisableForAllUser: false,  // 设置是否允许加入用户打开摄像头
  isMessageDisableForAllUser: false,  // 设置是否允许加入用户发送消息
});
```

参数：

参数	类型	说明	默认值	含义

roomId	string	必填	-	房间 ID，roomId 限制长度为64字节，且仅支持以下范围的字符集： • 大小写英文字母（a-zA-Z） • 数字（0-9） • 空格 ! # $ % & ( ) + - : ; < = . > ? @ [ ] ^ _ { } | ~ ,
roomName	string	选填	roomId	房间名称，默认值为 roomId, 不可传入空字符串
roomType	TUIRoomType	选填	TUIRoomType.kConference	房间类型 • 办公协同、医疗问诊、远程会议、教育场景，roomType 设置为 TUIRoomType.kConference • 电商直播、语聊房场景，roomType 设置为 TUIRoomType.kLivingRoom
isSeatEnabled	boolean	选填	false	是否开启麦位控制，默认不开启
seatMode	TUISeatMode	选填	TUISeatMode.kFreeToTake	上麦模式(开启麦位控制后生效)，默认值为 TUISeatMode.kFreeToTake 自由上麦模式，台下观众可以自由上麦，无需申请，seatMode 设置为 TUISeatMode.kFreeToTake 申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦，seatMode 设置为 TUISeatMode.kApplyToTake
isMicrophoneDisableForAllUser	boolean	选填	false	是否开启全员禁麦，默认不开启全员禁麦
isCameraDisableForAllUser	boolean	选填	false	是否开启全员禁画，默认不开启全员禁画
isMessageDisableForAllUser	boolean	选填	false	是否允许成员发送消息，默认不禁止
maxSeatCount	number	选填	-	最大麦位数量 • roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制 • roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16
enableCDNStreaming	boolean	选填	false	是否开启 CDN 直播流
cdnStreamDomain	string	选填	''	直播推流域名

Returns *Promise<void>*

enterRoom

进入房间接口。

```
const roomEngine = new TUIRoomEngine();
```

```
const roomInfo = await roomEngine.enterRoom({
  roomId: '12345',
});
```

参数：

参数	类型	说明	默认值	含义
roomId	string	必填	-	房间号

Returns Promise<TUIRoomInfo> roomInfo
该接口返回当前房间信息

destroyRoom

解散房间接口，解散房间必须由房间所有者发起，解散房间之后房间不可进入。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.destroyRoom();
```

Returns *Promise<void>*

exitRoom

离开房间接口，用户在执行 enterRoom 之后可通过 exitRoom 离开房间。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.exitRoom();
```

Returns *Promise<void>*

fetchRoomInfo

获取房间信息。

```
const roomEngine = new TUIRoomEngine();
const roomInfo = roomEngine.fetchRoomInfo();
```

Returns：*Promise<TUIRoomInfo> roomInfo*

updateRoomNameByAdmin

更新当前房间的名字（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomNameByAdmin({ roomName: '新的名字' });
```

参数：

参数	类型	说明	默认值	含义
roomName	string	必填	-	更新房间的名字，要求 roomName 不为空字符串

updateRoomSpeechModeByAdmin

更新房间的发言模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSpeechModeByAdmin({
  speechMode: TUISpeechMode.kSpeakAfterTakingSeat  // 更新为上麦后发言模式
});
```

参数：

参数	类型	说明	默认值	含义
speechMode	TUISpeechMode	必填	-	更新房间的发言模式

> ⚠ 注意：
> 该接口自 v2.0.0 版本废弃

updateRoomSeatModeByAdmin

更新房间上麦模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSeatModeByAdmin({
  seatMode: TUISeatMode.kApplyToTake,  // 更新为申请上麦模式
});
```

参数：

参数	类型	说明	默认值	含义
seatMode	TUISeatMode	必填	-	发言类型

getUserList

获取当前房间用户列表，注意该接口一次拉取的用户列表量最大为 50 个。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
let result;
let nextSequence = 0;
do {
  result = await roomEngine.getUserList({ nextSequence });
  userList.push(...result.userInfoList);
  nextSequence = result.nextSequence;
} while (result.nextSequence !== 0)
```

参数：

参数	类型	说明	默认值	含义
nextSequence	number	选填	0	偏移量，默认从 0 开始拉取用户

Returns：*Promise<Array> result*
result.nextSequence 下一次拉取群组用户的偏移量，如果 result.nextSequence 为 0 则代表 userList 全部拉取完毕。
result.userInfoList 本次拉取的 userList

getUserInfo

获取用户的详细信息。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
const userInfo = await roomEngine.getUserInfo({
   userId: 'user_12345',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	根据 userId 获取该用户的详细信息

Returns：*Promise<Array<TUIUserInfo >> userInfoList*
该接口返回指定用户的用户信息

setLocalVideoView

设置本地视频流的渲染位置。

```
const roomEngine = new TUIRoomEngine();

// 设置本地摄像头流的播放区域为 id 是 'preview-camera' 的 div 元素
await roomEngine.setLocalVideoView({
  view: 'preview-camera',
});
```

参数：

参数	类型	说明	默认值	含义
view	string	必填	-	streamType 对应的流渲染的 div 元素的 id

Returns：*Promise<void>*

> **说明：**
> 自 v1.6.1 版本起，setLocalVideoView 只需传入 view 参数，设置本地视频流的播放容器。

openLocalCamera

打开本地摄像头，开始视频流采集。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});

// case 1: 桌面端打开摄像头
await roomEngine.openLocalCamera();

// case 2: 移动端使用前置摄像头打开视频预览
await roomEngine.openLocalCamera({ isFrontCamera: true });
// case 3: 移动端使用后置摄像头打开视频预览
await roomEngine.openLocalCamera({ isFrontCamera: false });
```

参数：

参数	类型	说明	默认值	含义
isFrontCamera	boolean	选填	-	设置移动端打开本地预览视频的默认摄像头，该参数仅对移动端有效 该参数自 v1.3.1 版本开始支持

Returns：*Promise<void>*

closeLocalCamera

关闭本地摄像头。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalCamera();
```

Returns：*Promise<void>*

openLocalMicrophone

打开本地麦克风，开始采集音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.openLocalMicrophone();
```

Returns：*Promise<void>*

closeLocalMicrophone

关闭本地麦克风。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalMicrophone();
```

Returns：*Promise<void>*

版权所有：腾讯云计算（北京）有限责任公司

updateVideoQuality

设置本地视频流的编码参数，默认为 TUIVideoProfile.kVideoQuality_720P。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQuality({
  quality: TUIVideoQuality.kVideoQuality_540p,
});
```

参数：

参数	类型	说明	默认值	含义
quality	TUIVideoQuality	必填	-	• 清晰 TUIVideoProfile.kVideoQuality_360P • 标清 TUIVideoProfile.kVideoQuality_540P • 高清 TUIVideoProfile.kVideoQuality_720P • 超清 TUIVideoProfile.kVideoQuality_1080P

Returns：*Promise<void>*

setVideoResolutionMode

设置本地视频流的分辨率模式，默认为 TUIResolutionMode.kResolutionMode_Landscape。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setVideoResolutionMode({
  type: TUIVideoStreamType.kCameraStream,
  resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
resolutionMode	TUIResolutionMode	必填	TUIResolutionMode.kResolutionMode_Landscape	分辨率模式

Returns：*Promise<void>*

> **说明：**
> 该接口自 v1.5.0 版本开始支持。

updateVideoQualityEx

设置本地视频流的编码参数

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQualityEx({
  streamType: TUIVideoStreamType.kCameraStream,
```

```
  encoderParams: {
    videoResolution: TUIVideoQuality.kVideoQuality_720p,
    fps: 15,
    bitrate: 2000,
    resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
  }
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
encoderParams	TUIVideoEncoderParams	必填	-	详细编码参数

> **说明：**
> 该接口自 v1.5.0 版本开始支持。

updateAudioQuality

设置本地音频的参数。

> **注意：**
> 该方法需要在 openLocalMicrophone 之前进行设置，否则不会生效。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateAudioQuality({
  quality: TUIAudioQuality.kAudioProfileMusic,
});
```

参数：

参数	类型	说明	默认值	含义
audioProfile	TUIAudioQuality	必填	-	• TUIAudioQuality.kAudioProfileSpeech：语言模式；采样率：16k • TUIAudioQuality.kAudioProfileDefault：标准模式（或者默认模式）；采样率：48k • TUIAudioQuality.kAudioProfileMusic：音乐模式；采样率：48k

Returns: *Promise<void>*

startScreenSharing

开始屏幕共享。

```
const roomEngine = new TUIRoomEngine();
// example 1: 开始屏幕分享
```

```
await roomEngine.startScreenSharing();
// example 2: 开始屏幕分享(可采集系统音频)
await roomEngine.startScreenSharing({ screenAudio: true });
// example 3: 开始屏幕分享并在本地预览
await roomEngine.startScreenSharing({ view: 'screen-preview' });
```

参数：

参数	类型	说明	默认值	含义
screenAudio	boolean	选填	false	web 端屏幕共享是否可选共享系统声音, screenAudio 默认值为 false
view	string	选填	-	本地预览屏幕分享的 dom 元素的 id，该参数自 v1.6.1 版本支持

Returns：*Promise<void>*

stopScreenSharing

停止屏幕共享。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopScreenSharing();
```

Returns：*Promise<void>*

startPushLocalVideo

进房后会默认向远端推本地视频流，该接口用于停止推流后重新向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalVideo();
```

Returns：*Promise<void>*

stopPushLocalVideo

停止向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalVideo();
```

Returns：*Promise<void>*

startPushLocalAudio

进房后会默认向远端推本地音频流，该接口用于停止推流后重新向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalAudio();
```

Returns：*Promise<void>*

版权所有：腾讯云计算（北京）有限责任公司

> ⚠ **注意：**
> 该接口自 v1.5.0 版本废弃，请使用 muteLocalAudio 方法代替。

stopPushLocalAudio

停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalAudio();
```

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v1.5.0 版本废弃，请使用 muteLocalAudio 方法代替。

muteLocalAudio

停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteLocalAudio();
```

Returns：*Promise<void>*

> ⓘ **说明：**
> 该接口自 v1.5.1 版本开始支持。

unmuteLocalAudio

开始向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.unmuteLocalAudio();
```

Returns：*Promise<void>*

> ⓘ **说明：**
> 该接口自 v1.5.1 版本开始支持。

setRemoteVideoView

设置远端流渲染的区域。

```
const roomEngine = new TUIRoomEngine();

// 设置远端用户视频流在 id 为 'remote_preview_camera' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
```

版权所有：腾讯云计算（北京）有限责任公司

```
  view: 'remote_preview_camera',
});
// 设置远端用户屏幕分享流在 id 为 'remote_preview_screen' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'remote_preview_screen',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型
view	string	必填	-	播放远端用户流的 div 元素的 id

Returns：*Promise<void>*

startPlayRemoteVideo

开始播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流 • TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns：*Promise<void>*

stopPlayRemoteVideo

停止播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
```

```
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流 • TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns：*Promise<void>*

muteRemoteAudioStream

停止远端用户的音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteRemoteAudioStream({
  userId: 'user_1234',
  isMute: true,
});
```

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isMute	boolean	必填	-	是否停止远端用户的音频

openRemoteDeviceByAdmin

请求远端用户打开媒体设备。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.openRemoteDeviceByAdmin({
    userId: 'user_1234',
    device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
    timeout: 0,
    requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
     switch (requestCallbackType) {
       case TUIRequestCallbackType.kRequestAccepted:
         // 请求被接受
         break;
       case TUIRequestCallbackType.kRequestRejected:
         // 请求被拒绝
         break;
       case TUIRequestCallbackType.kRequestCancelled:
         // 请求已取消
         break;
       case TUIRequestCallbackType.kRequestTimeout:
         // 请求超时
```

```
      break;
    case TUIRequestCallbackType.kRequestError:
      // 请求错误
      break;
    default:
      break;
  }
 },
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

applyToAdminToOpenLocalDevice

参会者向主持人申请打开设备。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.applyToAdminToOpenLocalDevice({
  device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
   switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // 请求被接受
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // 请求被拒绝
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // 请求已取消
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // 请求超时
      break;
    case TUIRequestCallbackType.kRequestError:
```

```
      // 请求错误
      break;
    default:
      break;
  }
 },
});
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*

该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

closeRemoteDeviceByAdmin

关闭远端用户媒体设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone, //关闭麦克风
 });
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）

Returns：*Promise<void>*

cancelRequest

取消已经发出的请求。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.cancelRequest({
  requestId: '',    // 请使用实际 requestId
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID

Returns：*Promise<void>*

> **说明：**
> v1.0.2 及其以上版本该接口接收的 requestId 参数类型为 string；v1.0.0 及 v1.0.1 版本该接口接收的 requestId 参数类型为 number。

responseRemoteRequest

回复远端用户的请求。

```
const roomEngine = new TUIRoomEngine();
// 同意远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: true,
});
// 拒绝远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: false,
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID
agree	boolean	必填	-	是否同意

Returns：*Promise<void>*

> **说明：**
> v1.0.2 及其以上版本该接口接收的 requestId 参数类型为 string；v1.0.0 及 v1.0.1 版本该接口接收的 requestId 参数类型为 number。

disableDeviceForAllUserByAdmin

禁止/允许所有用户打开媒体设备（此接口对于房主和管理员无效）。

```
// Example 1: 禁止所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: true,
})
// Example 2: 允许所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: false,
```

```
})
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	被禁用的媒体设备类型（摄像头/麦克风/屏幕共享）
isDisable	boolean	必填	-	是否被禁止

Returns：*Promise<void>*

disableSendingMessageForAllUser

所有用户的是否允许发消息（此接口对于房主和管理员无效）。

```
await roomEngine.disableSendingMessageForAllUser({
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

disableSendingMessageByAdmin

特定用户是否被允许发消息。

```
await roomEngine.disableSendingMessageByAdmin({
  userId: 'user_1234',
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

changeUserRole

改变用户的角色（仅主持人可调用该接口）。

```
const roomEngine = new TUIRoomEngine();
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 将房间移交给用户 user_1234
await roomEngine.changeUserRole({
  userId: 'user_1234',
  role: TUIRole.kRoomOwner,
});

// 将用户 user_1234 设置为房间管理员
await roomEngine.changeUserRole({
 userId: 'user_1234',
 userRole: TUIRole.kAdministrator,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
userRole	TUIRole	必填	-	用户角色 • 主持人 TUIRole.kRoomOwner • 管理员 TUIRole.kAdministrator • 普通成员 TUIRole.kGeneralUser

Returns：*Promise<void>*

kickRemoteUserOutOfRoom

将用户踢出房间（仅主持人和管理员可调用该接口）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.kickRemoteUserOutOfRoom({
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID

Returns：*Promise<void>*

setMaxSeatCount

设置房间麦位最大值
roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制；
roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16；

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.setMaxSeatCount({ maxSeatCount: 16 })
```

参数：

参数	类型	说明	默认值	含义
maxSeatCount	number	必填	-	设置房间麦位最大值

Returns：Promise<void>

getSeatList

获取麦位列表。

```
const roomEngine = new TUIRoomEngine();
const seatList = await roomEngine.getSeatList();
```

Returns：*Promise*<TUISeatInfo *[]*> seatList
seatList 为当前房间所有麦位列表

takeSeat

麦下用户可调用 takeSeat 成为麦上用户，仅麦上用户可发布本地音视频流。

- 当 roomInfo.roomType 为 TUIRoomType.kConference 且 roomInfo.seatMode 为 TUISeatMode.kApplyToTake 时，普通用户调用 takeSeat 方法需要等待主持人/管理员的同意后成为麦上用户。
- 当 roomInfo.roomType 为 TUIRoomType.kLivingRoom 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 时，普通用户调用 takeSeat 方法成功后即为麦上用户。
- 主持人&管理员调用 takeSeat 成功后即为麦上用户。
- 麦上用户的变更通过 TUIRoomEvents.onSeatListChanged 通知所有用户。

```
const roomEngine = new TUIRoomEngine();
// 情景一：主持人/管理员上麦
// 情景二：当 roomInfo.roomType 为 TUIRoomType.kConference
// 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 模式时，普通用户上麦
await roomEngine.takeSeat({
 seatIndex: -1,
 timeout: 0,
});

// 情景三：roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户上麦
const requestId = await roomEngine.instance?.takeSeat({
  seatIndex: -1,
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
    switch (requestCallbackType) {
     case TUIRequestCallbackType.kRequestAccepted:
       // 请求被接受
       break;
     case TUIRequestCallbackType.kRequestRejected:
       // 请求被拒绝
       break;
     case TUIRequestCallbackType.kRequestCancelled:
       // 请求已取消
       break;
     case TUIRequestCallbackType.kRequestTimeout:
       // 请求超时
       break;
```

```
      case TUIRequestCallbackType.kRequestError:
        // 请求错误
        break;
      default:
        break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index, 无麦位序号时设置为 -1
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：Promise<string> requestId
roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户调用该接口时返回 requestId，普通用户可使用该 requestId 调用 cancelRequest 接口取消上麦请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

leaveSeat

释放麦位。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.leaveSeat();
```

Returns：*Promise<void>*

takeUserOnSeatByAdmin

邀请其他人上麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.takeUserOnSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
   switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // 请求被接受
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // 请求被拒绝
```

```
        break;
      case TUIRequestCallbackType.kRequestCancelled:
        // 请求已取消
        break;
      case TUIRequestCallbackType.kRequestTimeout:
        // 请求超时
        break;
      case TUIRequestCallbackType.kRequestError:
        // 请求错误
        break;
      default:
        break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID
timeout	number	必填	-	超时时间，设置为 0 时，无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string>* requestId
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

kickUserOffSeatByAdmin

要求其他人下麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = await roomEngine.kickUserOffSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID

Returns：*Promise<void>*

lockSeatByAdmin

锁定某个麦位状态（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.lockSeatByAdmin({
  seatIndex: 0,
  lockParams: {
    lockSeat: true,
    lockVideo: true,
    lockAudio: true,
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位索引
lockParams	TUISeatLockParams	必填	-	锁麦参数

Returns：*Promise<void>*

getSeatApplicationList

获取房间内申请上麦用户的请求列表, 该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
const applicationList = await roomEngine.getSeatApplicationList();
```

Returns：*Promise*<TUIRequest[]>

sendTextMessage

发送文本消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendTextMessage({
 messageText: 'hello, everyone',
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	文本消息内容

Returns：*Promise<void>*

> **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

sendCustomMessage

发送自定义消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendCustomMessage({
 messageText: '{ data:'', description: ''}',
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	自定义消息内容

Returns：*Promise<void>*

> **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

switchCamera

切换前后置摄像头，该接口适用于移动端。

> **说明：**
> 该接口自 v1.3.1 版本开始支持

```
// 移动端切换为前置摄像头
await roomEngine.switchCamera({ isFrontCamera: true });

// 移动端切换为后置摄像头
await roomEngine.switchCamera({ isFrontCamera: false });
```

参数：

参数	类型	说明	默认值	含义
isFrontCamera	boolean	必填	-	是否切换为前置摄像头

Returns：*Promise<void>*

getCameraDevicesList

获取摄像头设备列表。

```
const roomEngine = new TUIRoomEngine();
const cameraList = await roomEngine.getCameraDevicesList();
for (i = 0; i < cameraList.length; i++) {
```

版权所有：腾讯云计算（北京）有限责任公司

```
  var camera = cameraList[i];
  console.info("camera deviceName: " + camera.deviceName + " deviceId:" + camera.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> cameraList

getMicDevicesList

获取麦克风设备列表。

```
const roomEngine = new TUIRoomEngine();
const micList = await roomEngine.getMicDevicesList();
for (i = 0; i < micList.length; i++) {
  var mic = micList[i];
  console.info("mic deviceName: " + mic.deviceName + " deviceId:" + mic.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> micList

getSpeakerDevicesList

获取扬声器设备列表。

```
const roomEngine = new TUIRoomEngine();
const speakerList = await roomEngine.getSpeakerDevicesList();
for (i = 0; i < speakerList.length; i++) {
  var speaker = speakerList[i];
  console.info("speaker deviceName: " + speaker.deviceName + " deviceId:" + speaker.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> speakerList

setCurrentCameraDevice

设置要使用的摄像头设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentCameraDevice({ deviceId: '' });
```

参数：

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getCameraDevicesList 中得到的设备 ID

Returns: *void*

setCurrentMicDevice

设置要使用的麦克风设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentMicDevice({ deviceId: '' });
```

版权所有：腾讯云计算（北京）有限责任公司

参数：

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getMicDevicesList 中得到的设备 ID

Returns：*void*

setCurrentSpeakerDevice

设置要使用的扬声器设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentSpeakerDevice({ deviceId: '' });
```

参数：

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getSpeakerDevicesList 中得到的设备 ID

Returns：*void*

getCurrentCameraDevice

获取当前正在使用的摄像头设备。

```
const roomEngine = new TUIRoomEngine();
const currentCameraDevice = roomEngine.getCurrentCameraDevice();
```

Returns：TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

getCurrentMicDevice

获取当前正在使用的麦克风设备。

```
const roomEngine = new TUIRoomEngine();
const currentMicDevice = roomEngine.getCurrentMicDevice();
```

Returns：TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

getCurrentSpeakerDevice

获取当前正在使用的扬声器设备。

```
const roomEngine = new TUIRoomEngine();
const currentSpeakerDevice = roomEngine.getCurrentSpeakerDevice();
```

Returns：TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

版权所有：腾讯云计算（北京）有限责任公司

startCameraDeviceTest

开始进行摄像头测试。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startCameraDeviceTest({ view: 'test-preview' });
```

参数：

参数	类型	说明	默认值	含义
view	string	必填	-	显示摄像头测试的视频区域，传入的 view 为承载预览画面 div 元素的 Id

Returns：*Promise<void>*

stopCameraDeviceTest

停止摄像头测试。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopCameraDeviceTest();
```

Returns：*Promise<void>*

on

监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(event, func);
```

参数：

参数	类型	说明	默认值	含义
event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns：*void*

off

取消监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.off(event, func);
```

参数：

参数	类型	说明	默认值	含义

event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns：*void*

getTRTCCloud

获取 trtcCloud 实例, web 端 trtcCloud 能力请查看：TRTCCloud API 文档 。

```
const roomEngine = new TUIRoomEngine();
const trtcCloud = roomEngine.getTRTCCloud();
```

Returns：TRTCCloud

getTIM

获取 tim 实例, web 端 tim 能力请查看： IM API 文档 。

```
const roomEngine = new TUIRoomEngine();
const tim = roomEngine.getTIM();
```

Returns：tim

> ⚠ **注意：**
> 自 v1.6.0 版本后，IM SDK 升级至 v3 版本，请查看： IM API 文档 。

TUIRoomEvents

最近更新时间：2024-03-04 14:32:51

TUIRoomEvent API 简介

TUIRoomEvent API 是音视频通话组件的**事件接口**。

事件列表

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	踢出房间事件
TUIRoomEvents.onKickedOffLine	当前用户被踢下线事件
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomNameChanged	房间名称修改事件
~~TUIRoomEvents.onRoomSpeechModeChanged~~	房间发言模式修改事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onAllUserCameraDisableChanged	所有成员摄像头使用权限改变事件
TUIRoomEvents.onAllUserMicrophoneDisableChanged	所有成员麦克风使用权限改变事件
TUIRoomEvents.onSendMessageForAllUserDisableChanged	所有成员发送消息状态改变事件
TUIRoomEvents.onRoomMaxSeatCountChanged	房间最大麦位数修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChanged	用户音频状态改变事件
TUIRoomEvents.onSendMessageForUserDisableChanged	用户发送消息状态事件
TUIRoomEvents.onUserVoiceVolumeChanged	用户音量改变事件
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEvents.onSeatListChanged	麦位列表改变事件
TUIRoomEvents.onKickedOffSeat	用户被踢下麦事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
~~TUIRoomEvents.onReceiveCustomMessage~~	接收自定义消息事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onDeviceChange	设备变更事件
TUIRoomEvents.onUserScreenCaptureStopped	屏幕分享停止事件 当用户使用浏览器自带的**停止分享**按钮结束屏幕共享时，该用户会收到 'onUserScreenCaptureStopped' 事件用来修改屏幕共享状态。

onError

错误事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onError, (error) => {
 console.log('TUIRoomError error', error);
})
```

参数如下表所示：

参数	类型	含义
code	number	错误代码
message	string	错误信息

onKickedOutOfRoom

踢出房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOutOfRoom, ({ roomId, reason, message }) => {
  console.log('roomEngine.onKickedOutOfRoom', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
reason	TUIKickedOutOfRoomReason	用户被踢出房间原因
message	string	踢出房间信息

onKickedOffLine

当前用户被踢下线事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffLine, ({ message }) => {
  console.log('roomEngine.onKickedOffLine', message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
message	string	用户在其他端登录信息

onUserSigExpired

userSig 过期事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserSigExpired, () => {
  console.log('roomEngine.onUserSigExpired');
});
```

onRoomDismissed

主持人销毁房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomDismissed, ({ roomId }) => {
  console.log('roomEngine.onRoomDismissed', roomId);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号

onRoomNameChanged

房间名称修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomNameChanged, ({ roomId, roomName }) => {
  console.log('roomEngine.onRoomNameChanged', roomId, roomName);
});
```

参数如下表所示：

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
roomId	string	房间号
roomName	string	房间名称

onRoomSpeechModeChanged

房间模式修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSpeechModeChanged, ({ roomId, speechMode }) => {
  console.log('roomEngine.onRoomSpeechModeChanged', roomId, speechMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
speechMode	TUISpeechMode	发言模式

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。

onRoomSeatModeChanged

上麦模式修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSeatModeChanged, ({ roomId, seatMode }) => {
  console.log('roomEngine.onRoomSeatModeChanged', roomId, seatMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
seatMode	TUISeatMode	房间上麦模式

onAllUserCameraDisableChanged

所有成员摄像头使用权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserCameraDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserCameraDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许使用摄像头

onAllUserMicrophoneDisableChanged

所有成员麦克风使用权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserMicrophoneDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserMicrophoneDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许使用麦克风

onSendMessageForAllUserDisableChanged

所有成员发送消息权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onRoomMaxSeatCountChanged

房间最大麦位数修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomMaxSeatCountChanged, ({ maxSeatNumber }) => {
  console.log('roomEngine.onRoomMaxSeatCountChanged', maxSeatNumber);
});
```

参数如下表所示：

参数	类型	含义
maxSeatNumber	number	最大麦位数量

onRemoteUserEnterRoom

远端用户进入房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserEnterRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserEnterRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onRemoteUserLeaveRoom

远端用户离开房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserLeaveRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserLeaveRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onUserRoleChanged

用户角色改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserRoleChanged, ({ userId, userRole }) => {
  console.log('roomEngine.onUserRoleChanged', userId, userRole);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
userRole	TUIRole	用户变更后的角色

onUserVideoStateChanged

用户视频状态改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVideoStateChanged, ({ userId, streamType, hasVideo, reason }) => {
  console.log('roomEngine.onUserVideoStateChanged', userId, streamType, hasVideo, reason);
```

```
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
streamType	TUIVideoStreamType	用户流类型
hasVideo	boolean	是否有视频流
reason	TUIChangeReason	变更原因, 自己操作/主持人操作

onUserAudioStateChanged

用户音频状态改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserAudioStateChanged, ({ userId, hasAudio, reason }) => {
  console.log('roomEngine.onUserAudioStateChanged', userId, hasAudio, reason);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
hasVideo	boolean	是否有音频流
reason	TUIChangeReason	变更原因，自己操作/主持人操作

onSendMessageForUserDisableChanged

有成员摄像头使用权限修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onUserVoiceVolumeChanged

用户音量改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVoiceVolumeChanged, ({ userVolumeList }) => {
```

```
  userVolumeList.forEach(userVolume => {
    console.log('roomEngine.onUserVoiceVolumeChanged', userVolume.userId, userVolume.volume);
  })
});
```

参数如下表所示：

参数	类型	含义
userVolumeList	Array.<object>	房间内所有用户的音量，包含 userId 及 volume 信息，volume 区间为 1~100

onUserNetworkQualityChanged

用户网络质量改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserNetworkQualityChanged, ({ userNetworkList }) => {
  userNetworkList.forEach(userNetwork => {
    console.log('roomEngine.onUserNetworkQualityChanged', userNetwork.userId, userNetwork.quality,
userNetwork.upLoss, userNetwork.downLoss, userNetwork.delay);
  })
});
```

参数如下表所示：

参数	类型	含义
userNetworkList	Array.<TUINetwork>	网络质量信息列表

onSeatListChanged

麦位列表改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatListChanged, ({ seatList, seatedList, leftList }) => {
  console.log('roomEngine.onSeatListChanged',seatList, seatedList, leftList);
});
```

参数如下表所示：

参数	类型	含义
seatList	Array.<TUISeatInfo>	麦位列表
seatedList	Array.<TUISeatInfo>	新增麦位信息
leftList	Array.<TUISeatInfo>	离开的麦位信息

onKickedOffSeat

麦位列表改变事件

```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onKickedOffSeat, ({ userId }) => {
  console.log('roomEngine.onKickedOffSeat', userId);
});
```

参数如下表所示：

参数	类型	含义
userId	String	被踢下麦的用户 ID

onRequestReceived

请求接收事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestReceived, ({ request }) => {
  console.log('roomEngine.onRequestReceived', request);
});
```

参数如下表所示：

参数	类型	含义
request	TUIRequest	请求接收

onRequestProcessed

请求被处理事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestProcessed, ({ request }) => {
  console.log('roomEngine.onRequestProcessed', request);
});
```

参数如下表所示：

参数	类型	含义
request	TUIRequest	请求接收

onRequestCancelled

请求取消事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestCancelled, ({ requestId, userId }) => {
  console.log('roomEngine.onRequestCancelled', requestId, userId);
});
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

requestId	string	请求 ID
userId	string	取消请求的用户 ID

onReceiveTextMessage

接收文本消息事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveTextMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveTextMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收文本消息

> **注意：**
> 该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

onReceiveCustomMessage

接收自定义消息事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveCustomMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveCustomMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收自定义消息

> **注意：**
> 该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

onDeviceChange

设备变更事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onDeviceChange, ({ deviceId, type, state }) => {
  console.log('roomEngine.onDeviceChange', deviceId, type, state);
});
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
deviceId	string	设备 ID
type	TRTCDeviceType	设备类型
state	TRTCDeviceState	设备变更状态

onUserScreenCaptureStopped

屏幕分享停止事件，当用户使用浏览器自带的**停止分享**按钮结束屏幕共享时，该用户会收到 'onUserScreenCaptureStopped' 事件用来修改屏幕共享状态。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserScreenCaptureStopped, () => {
  console.log('roomEngine.onUserScreenCaptureStopped', deviceId, type, state);
});
```

TUIRoomEngine Defines

最近更新时间：2024-02-20 17:41:31

本文介绍 TUIRoomEnigne web 端关键类型定义。

枚举值

TUIRole

用户角色，TUIRoomEngine 共提供三种用户角色，分别是主持人，管理员，普通用户。

字段	类型	描述
kRoomOwner	number	主持人角色
kAdministrator	number	管理员角色
kGeneralUser	number	普通用户角色

TUIVideoQuality

视频分辨率

字段	类型	描述
kVideoQuality_360p	number	低清，分辨率为 640 * 360
kVideoQuality_540p	number	标清，分辨率为 960 * 540
kVideoQuality_720p	number	高清，分辨率为 1280 * 720
kVideoQuality_1080p	number	超清，分辨率为 1920 * 1080

TUIResolutionMode

分辨率模式（横屏分辨率 | 竖屏分辨率）

字段	类型	描述
kResolutionMode_Landscape	number	横屏模式
kResolutionMode_Portrait	number	竖屏模式

TUIAudioQuality

音频分辨率

字段	类型	描述
kAudioProfileSpeech	number	语音模式
kAudioProfileDefault	number	标准模式（默认模式）
kAudioProfileMusic	number	音乐模式

版权所有：腾讯云计算（北京）有限责任公司

TUIVideoStreamType

视频流类型

字段	类型	描述
kCameraStream	number	摄像头视频流
kScreenStream	number	屏幕分享视频流
kCameraStreamLow	number	低清摄像头视频流

TUINetworkQuality

网络状态

字段	类型	描述
kQualityUnknown	number	网络状况未知
kQualityExcellent	number	网络状况极佳
kQualityGood	number	网络状况较好
kQualityPoor	number	网络状况一般
kQualityBad	number	网络状况差
kQualityVeryBad	number	网络状况极差
kQualityDown	number	网络连接已断开

TUIRoomType

房间类型

字段	类型	描述
kConference	number	kConference 类型房间，适用于会议、教育场景
kLivingRoom	number	kLivingRoom 类型房间，适用于直播场景

TUISpeechMode

发言类型

字段	类型	描述
kFreeToSpeak	number	自由发言模式
kApplyToSpeak	number	举手发言模式
kSpeakAfterTakingSeat	number	就坐后发言（抢麦位）

注意：
该类型自 v2.0.0 版本废弃

版权所有：腾讯云计算（北京）有限责任公司

TUISeatMode

上麦模式

字段	类型	描述
kFreeToTake	number	自由上麦模式，台下观众可以自由上麦，无需申请
kApplyToTake	number	申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦

TUIChangeReason

更改原因（用户音视频状态变更操作原因: 自己主动修改 或者 被房主、管理员修改）

字段	类型	描述
kChangedBySelf	number	自己操作
kChangedByAdmin	number	房主或管理员操作

TUIMediaDevice

字段	类型	描述
kMicrophone	number	麦克风
kCamera	number	摄像头
kScreen	number	屏幕分享

TUIRequestAction

房间类型

字段	类型	描述
kInvalidAction	number	无效操作
kRequestToOpenRemoteCamera	number	请求远端打开摄像头
kRequestToOpenRemoteMicrophone	number	请求远端打开麦克风
kRequestToConnectOtherRoom	number	请求远端跨房连麦，web 端暂不支持
kRequestToTakeSeat	number	请求上麦
kRequestRemoteUserOnSeat	number	请求远端上麦
kApplyToAdminToOpenLocalCamera	number	向管理员请求打开本地摄像头
kApplyToAdminToOpenLocalMicrophone	number	向管理员请求打开本地麦克风

TUIRequestCallbackType

版权所有：腾讯云计算（北京）有限责任公司

请求类型

字段	类型	描述
kRequestAccepted	number	对端已接受
kRequestRejected	number	对端已拒绝
kRequestCancelled	number	请求已取消
kRequestTimeout	number	请求超时
kRequestError	number	请求错误

TUIKickedOutOfRoomReason

用户被踢出房间原因

字段	类型	描述
kKickedByAdmin	number	被主持人或管理员踢出
kKickedByLoggedOnOtherDevice	number	相同 userId 用户进入同一房间被踢出
kKickedByServer	number	被服务端踢出

类型定义

TUILoginUserInfo

当前登录用户信息

名称	类型	描述
userId	string	登录用户的 ID
userName	string	登录用户的名称
avatarUrl	string	登录用户的头像
customInfo	object	登录用户的自定义资料，自 v1.3.0 版本支持

TUIRoomInfo

房间信息，用户可使用 roomEngine.fetchRoomInfo 获取房间信息。

名称	类型	描述
roomId	string	房间号，字符串类型房间号
roomName	string	房间名称
roomType	TUIRoomType	房间类型
isSeatEnabled	boolean	是否开启麦位控制（创建房间可选参数，默认值为 false）
seatMode	TUISeatMode	房间内上麦模式（开启麦位控制后生效，默认值为

		TUISeatMode.kFreeToTake)
isMicrophoneDisableForAllUser	boolean	是否开启全员禁音（创建房间可选参数，默认值为 false ）
isCameraDisableForAllUser	boolean	是否开启全员禁画（创建房间可选参数， 默认值为 false ）
isMessageDisableForAllUser	boolean	是否允许所有用户发消息（创建房间可选参数，默认值为 false ）
maxSeatCount	number	最大麦位数量，默认为 6
roomOwner	string	房间主持人，只读
createTime	number	房间创建时间，只读，精确到秒
roomMemberCount	number	房间成员数量，只读

TUIUserInfo

用户信息

名称	类型	描述
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址
userRole	TUIRole	用户角色
hasAudioStream	boolean	是否有音频流
hasVideoStream	boolean	是否有视频流
hasScreenStream	boolean	是否有屏幕分享流
roomCustomInfo	object	用户房间自定义字段，自 v1.3.0 版本支持

TUIMessage

消息信息

名称	类型	描述
messageId	string	消息 ID
message	string	消息内容
timestamp	number	时间戳信息, 精确到秒
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址

TUIRequest

请求信息

名称	类型	描述
requestAction	TUIRequestAction	请求类型
timestamp	number	请求发起时间
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	发起请求的用户 ID
content	string	其他内容

TUIRequestCallback

请求回调信息

名称	类型	描述
requestCallbackType	TUIRequestCallbackType	请求回调类型，接受/拒绝/取消/超时/错误
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	用户 ID
code	number	请求响应码
message	string	请求状态补充说明

TUISeatInfo

麦位信息

名称	类型	描述
index	number	麦位序号
userId	string	麦位对应的用户 ID
locked	boolean	当前麦位是否被锁
videoMuted	boolean	当前麦位是否禁止视频
audioMuted	boolean	当前麦位是否禁止音频

TUISeatLockParams

麦位锁定状态

名称	类型	描述

版权所有：腾讯云计算（北京）有限责任公司

lockSeat	boolean	锁定麦位
lockVideo	boolean	锁定麦位视频
lockAudio	boolean	锁定麦位音频

TUINetwork

网络信息

名称	类型	描述
userId	string	用户 ID
quality	TUINetworkQuality	网络质量
upLoss	number	上行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
downLoss	number	下行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
delay	number	网络延迟，单位 ms，目前仅本地用户有该信息

TUIVideoEncoderParams

视频编码参数

名称	类型	描述
videoResolution	TUIVideoQuality	视频分辨率, 该字段在 v1.6.0 版本由 quality 修改为 videoResolution
fps	number	视频帧率
bitrate	number	视频码率
resolutionMode	TUIResolutionMode	横竖屏模式

Electron
API 概览

最近更新时间：2024-03-04 14:32:51

TUIRoomEngine (无 UI 接口)

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.init 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.init 方法执行成功后执行**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用）该接口自 v2.0.0 版本废弃
updateRoomSeatModeByAdmin	更新房间上麦模式（仅群主或者管理员可以调用）
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集

closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
muteLocalAudio	停止向远端推本地音频流
unmuteLocalAudio	开始向远端推本地音频流
updateVideoQuality	设置本地视频的参数
updateAudioQuality	设置本地音频的参数
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
startPushLocalAudio	开始向远端推本地音频流
stopPushLocalAudio	停止向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域
startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流
muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备
cancelRequest	取消已经发出的请求
responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户踢出房间
disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 屏幕分享 API

API	描述
startScreenSharingElectron	开始屏幕共享
stopScreenSharingElectron	停止屏幕共享
getScreenSharingTarget	获取屏幕分享列表
selectScreenSharingTarget	切换屏幕分享窗口

roomEngine 麦位管理API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人踢下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送 API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

roomEngine 设备管理 API

API	描述
getCameraDevicesList	获取摄像头设备列表
getMicDevicesList	获取麦克风设备列表
getSpeakerDevicesList	获取扬声器设备列表
setCurrentCameraDevice	设置要使用的摄像头设备
setCurrentMicDevice	设置要使用的麦克风设备
setCurrentSpeakerDevice	设置要使用的扬声器设备
getCurrentCameraDevice	获取当前正在使用的摄像头设备
getCurrentMicDevice	获取当前正在使用的麦克风设备
getCurrentSpeakerDevice	获取当前正在使用的扬声器设备

版权所有：腾讯云计算（北京）有限责任公司

startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止摄像头测试

roomEngine 事件监听 API

API	描述
on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

事件类型定义

TUIRoomEvent 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	踢出房间事件
TUIRoomEvents.onKickedOffSeat	用户被踢下麦事件
TUIRoomEvents. onKickedOffLine	当前用户被踢下线
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomInfoChanged	房间信息修改事件
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserMuteStateChanged	用户禁止发文字消息状态改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChan	用户音频状态改变事件

版权所有：腾讯云计算（北京）有限责任公司

ged	
TUIRoomEvents.onUserVoiceVolumeChanged	用户音量改变事件
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件
TUIRoomEvents.onSeatControlEnabled	麦位控制启用事件
TUIRoomEvents.onSeatListChanged	麦位列表改变事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
~~TUIRoomEvents.onReceiveCustomMessage~~	接收自定义消息事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onDeviceChange	设备变更事件
TUIRoomEvents.onUserScreenCaptureStopped	屏幕分享停止事件，当用户使用浏览器自带的**停止分享**按钮结束屏幕共享时，该用户会收到 'onUserScreenCaptureStopped' 事件用来修改屏幕共享状态

TUIRoomKit

最近更新时间：2024-02-20 17:41:31

API 简介

TUIRoomkit API 是多人视频会议组件的**含 UI 接口**，使用 TUIRoomKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景，如果您想要体验并调试视频会议效果，请阅读 Demo 快速跑通，如果您想把我们的功能直接嵌入到您的项目中，请阅读 快速接入(TUIRoomKit)。

API 概览

在页面中引用 TUIRoom 组件。例如：在 App.vue 组件中引入 TUIRoom 组件。

- TUIRoom 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init、createRoom、enterRoom 方法。
- 主持人及普通成员可通过 init 方法向 TUIRoom 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
  <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
  import { ref, onMounted } from 'vue';
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';
  // 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
  const TUIRoomRef = ref();

   onMounted(async () => {
    // 初始化 TUIRoom 组件
    // 主持人在创建房间前需要先初始化 TUIRoom 组件
    // 普通成员在进入房间前需要先初始化 TUIRoom 组件
    await TUIRoomRef.value.init({
      // 获取 sdkAppId 请您参考 步骤一
      sdkAppId: 0,
      // 用户在您业务中的唯一标识 Id
      userId: '',
      // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig
      // 注意 userSig 与 userId 为一一对应关系
      userSig: '',
      // 用户在您业务中使用的昵称
      userName: '',
      // 用户在您业务中使用的头像链接
      avatarUrl: '',
      // 用户在您业务中需要的皮肤主题颜色及是否支持切换皮肤主题
      theme: {
        defaultTheme:'black',
        isSupportSwitchTheme: true
      }
    })
     // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
    await handleCreateRoom();
  })
```

```
    // 主持人创建房间，该方法只在创建房间时调用
    async function handleCreateRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 number
      // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认为'FreeToSpeak'，注意目前仅支持自由发言模式
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
      const roomId = '123456';
      const roomMode = 'FreeToSpeak';
      const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
      };
      const createRoomInfo = {
        roomId,
        roomName: 'User defined room name' || roomId,
        roomMode,
        roomParam
      };
      await TUIRoomRef.value.createRoom(createRoomInfo);
    }

    // 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
    async function handleEnterRoom() {
      // roomId 为用户进入的房间号, 要求 roomId 类型为 number
      // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
      const roomId = '123456';
      const roomParam = {
        isOpenCamera: true,
        isOpenMicrophone: true,
      };
      const enterRoomInfo = {
        roomId,
        roomParam
      };
      await TUIRoomRef.value.enterRoom(enterRoomInfo);
    }
</script>

<style>
html, body {
    width: 100%;
    height: 100%;
    margin: 0;
}

#app {
    width: 100%;
    height: 100%;
}
</style>
```

API 详情

TUIRoomkit 接口

init

初始化 TUIRoom 数据，任何使用 TUIRoom 的用户都需要调用 init 方法。

```
TUIRoomRef.value.init(roomData);
```

参数如下表所示：

参数	类型	含义
roomData	object	房间初始化参数
roomData.sdkAppId	number	客户的 SDKAppId
roomData.userId	string	用户的唯一 ID
roomData.userSig	string	用户的 UserSig
roomData.userName	string	用户的昵称
roomData.avatarUrl	string	用户的头像地址
roomData.theme	object	界面的主题风格
roomData.theme.defaultTheme	string	默认主题皮肤颜色
roomData.theme.isSupportSwitchTheme	boolean	是否支持切换主题

createRoom

主持人创建房间。

```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomMode	string	房间模式，FreeToSpeak（自由发言模式）和 SpeakAfterTakingSeat（上台发言模式），默认为 FreeToSpeak
roomParam	object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID

版权所有：腾讯云计算（北京）有限责任公司

roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	string	非必填，默认扬声器设备 ID

enterRoom

普通成员加入房间。

```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomParam	object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭
roomParam.defaultCameraId	string	非必填，默认摄像头设备 ID
roomParam.defaultMicrophoneId	string	非必填，默认麦克风设备 ID
roomParam.defaultSpeakerId	string	非必填，默认扬声器设备 ID

TUIRoomkit 事件

onCreateRoom

创建房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-create-room="onCreateRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onCreateRoom(info) {
    if (info.code === 0) {
      console.log('创建房间成功')
    }
  }
</script>
```

onEnterRoom

进入房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-enter-room="onEnterRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onEnterRoom(info) {
    if (info.code === 0) {
      console.log('进入房间成功')
    }
  }
</script>
```

onDestroyRoom

主持人销毁房间通知。

```
<template>
  <room ref="TUIRoomRef" @on-destroy-room="onDestroyRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onDestroyRoom(info) {
    if (info.code === 0) {
      console.log('主持人销毁成功')
    }
  }
</script>
```

onExitRoom

普通成员退出房间通知。

```
<template>
  <room ref="TUIRoomRef" @on-exit-room="onExitRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onExitRoom(info) {
    if (info.code === 0) {
      console.log('普通成员退出房间成功')
    }
```

版权所有：腾讯云计算（北京）有限责任公司

```
  }
</script>
```

onKickedOutOfRoom

普通成员被主持人踢出房间通知。

```
<template>
  <room ref="TUIRoomRef" @on-kicked-out-of-room="onKickedOutOfRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOutOfRoom({ roomId, message }) {
    console.log('普通成员被主持人踢出房间', roomId, message);
  }
</script>
```

onKickedOffLine

用户账号在其他地方登录，被踢下线。

```
<template>
  <room ref="TUIRoomRef" @on-kick-off-line="onKickedOffLine"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOffLine({ message }) {
    console.log('成员被踢下线', message);
  }
</script>
```

onUserSigExpired

用户 userSig 过期，请重新获取 userSig。

```
<template>
  <room ref="TUIRoomRef" @on-user-sig-expired="onUserSigExpired"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onUserSigExpired() {
    console.log('用户 userSig 过期，请重新生成 userSig.');
  }
</script>
```

版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEngine

最近更新时间：2024-03-04 14:32:51

TUIRoomEngine 简介

TUIRoomEngine SDK 提供了房间管理，多人实时音视频互动，屏幕分享，成员管理，即时聊天等功能。
安装方式：

```
// 使用 npm
npm i @tencentcloud/tuiroom-engine-electron --save

// 使用 pnpm
pnpm i @tencentcloud/tuiroom-engine-electron --save

// 使用 yarn
yarn add @tencentcloud/tuiroom-engine-electron
```

TUIRoomEngine API

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.login 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.login 方法执行成功后执行。**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用），该接口自 v2.0.0 版本废弃
updateRoomSeatModeByAd	更新房间上麦模式（仅群主或者管理员可以调用）

min	
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集
closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateVideoQuality	设置本地视频的参数
setVideoResolutionMode	设置本地视频流的分辨率模式
updateVideoQualityEx	设置本地视频的编码参数
updateAudioQuality	设置本地音频的参数
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
muteLocalAudio	停止向远端推本地音频流
unmuteLocalAudio	开始向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域
startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流
muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备
cancelRequest	取消已经发出的请求

版权所有：腾讯云计算（北京）有限责任公司

responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户踢出房间
disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 屏幕分享 API

API	描述
startScreenSharingElectron	开始屏幕共享
stopScreenSharingElectron	停止屏幕共享
getScreenSharingTarget	获取屏幕分享列表
selectScreenSharingTarget	切换屏幕分享窗口

roomEngine 麦位管理API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人踢下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

版权所有：腾讯云计算（北京）有限责任公司

roomEngine 设备管理 API

API	描述
getCameraDevicesList	获取摄像头设备列表
getMicDevicesList	获取麦克风设备列表
getSpeakerDevicesList	获取扬声器设备列表
setCurrentCameraDevice	设置要使用的摄像头设备
setCurrentMicDevice	设置要使用的麦克风设备
setCurrentSpeakerDevice	设置要使用的扬声器设备
getCurrentCameraDevice	获取当前正在使用的摄像头设备
getCurrentMicDevice	获取当前正在使用的麦克风设备
getCurrentSpeakerDevice	获取当前正在使用的扬声器设备
startCameraDeviceTest	开始进行摄像头测试
stopCameraDeviceTest	停止摄像头测试

roomEngine 事件监听 API

API	描述
on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

API 详情

once

监听 TUIRoomEngine 'ready' 事件

```
TUIRoomEngine.once('ready', () => {
  const roomEngine = new TUIRoomEngine();

  await TUIRoomEngine.init({
    sdkAppId: 0,   // 填写您申请的 sdkAppId
    userId: '',    // 填写您业务对应的 userId
    userSig: '',   // 填写服务器或本地计算的 userSig
  });
```

```
  await roomEngine.createRoom({
    roomId: '12345',  // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
    name: 'Test Room',    // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
    roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
  });
});
```

login

> **说明：**
> v1.0.0 版本该接口命名为 TUIRoomEngine.init 接口，v1.0.1 及其以上版本请使用 TUIRoomEngine.login 登录 TUIRoomEngine。

登录 TUIRoomEngine， 您必须在登录 TUIRoomEngine 之后才可以调用 TUIRoomEngine 及其实例的其他方法。

```
// 登录 TUIRoomEngine
await TUIRoomEngine.login({
 sdkAppId: 0,  // 填写您申请的 sdkAppId
 userId: '',  // 填写您业务对应的 userId
 userSig: '',  // 填写服务器或本地计算的 userSig
});
```

参数：

参数	类型	说明	默认值	含义
sdkAppId	number	必填	-	sdkAppId 在 实时音视频控制台 单击 应用管理 > 创建应用 创建新应用之后，即可在 应用信息 中获取 sdkAppId 信息。
userId	string	必填	-	用户ID建议限制长度为32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。
userSig	string	必填	-	userSig 签名 计算 userSig 的方式请参考 UserSig 相关 。
tim	TIM	非必填	-	若您在接入 roomEngine 的同时想使用即时通信 SDK 的更多能力，您可以将创建的 tim 实例传入 TUIRoomEngine。tim 实例的创建方式请参考 TIM.create 。

Returns *Promise<void>*

setSelfInfo

设置当前用户基本信息（用户名、用户头像）。

```
// 设置当前用户用户名及用户头像
await TUIRoomEngine.setSelfInfo({
 userName: '',  // 填写您的新用户名
 avatarUrl: '',  // 填写您的新头像地址
});
```

参数：

参数	类型	说明	默认值	含义
userName	string	必填	-	用户名
avatarUrl	string	必填	-	用户头像

Returns *Promise<void>*

getSelfInfo

获取当前用户基本信息（用户名、用户头像）。

```
// 获取当前用户用户名及用户头像
const loginUserInfo = await TUIRoomEngine.getSelfInfo();
```

Returns *Promise<*TUILoginUserInfo *> loginUserInfo*

logout

> **说明：**
> 该接口自 v1.0.1 版本支持。

登出 TUIRoomEngine。

```
// 登出 TUIRoomEngine
await TUIRoomEngine.logout();
```

Returns *Promise<void>*

createRoom

主持人创建房间，调用 createRoom 的用户为房间的所有者。创建房间时可设置房间 ID、房间名称、房间类型、是否开启麦位控制、是否允许加入用户开启音视频、发送消息等功能。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({
  roomId: '12345',  // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
  roomName: 'Test Room',    // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
  roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
  isSeatEnabled: false, // 设置是否开启麦位控制
  isMicrophoneDisableForAllUser: false,  // 设置是否允许加入用户打开麦克风
  isCameraDisableForAllUser: false,  // 设置是否允许加入用户打开摄像头
  isMessageDisableForAllUser: false,  // 设置是否允许加入用户发送消息
});
```

参数：

参数	类型	说明	默认值	含义
roomId	string	必	-	房间 ID，roomId 限制长度为64字

		填		节，且仅支持以下范围的字符集： • 大小写英文字母（a-zA-Z） • 数字（0-9） • 空格！# $ % & ( ) + - : ; < = . > ? @ [ ] ^ _ { } | ~ ,
roomName	string	选填	roomId	房间名称，默认值为 roomId，不可传入空字符串
roomType	TUIRoomType	选填	TUIRoomType.kConference	房间类型 • 办公协同、医疗问诊、远程会议、教育场景，roomType 设置为 TUIRoomType.kConference • 电商直播、语聊房场景，roomType 设置为 TUIRoomType.kLivingRoom
isSeatEnabled	boolean	选填	false	是否开启麦位控制，默认不开启
seatMode	TUISeatMode	选填	TUISeatMode.kFreeToTake	上麦模式(开启麦位控制后生效)，默认值为 TUISeatMode.kFreeToTake 自由上麦模式，台下观众可以自由上麦，无需申请，seatMode 设置为 TUISeatMode.kFreeToTake 申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦，seatMode 设置为 TUISeatMode.kApplyToTake
isMicrophoneDisableForAllUser	boolean	选填	false	是否开启全员禁麦，默认不开启全员禁麦
isCameraDisableForAllUser	boolean	选填	false	是否开启全员禁画，默认不开启全员禁画
isMessageDisableForAllUser	boolean	选填	false	是否允许成员发送消息，默认不禁止
maxSeatCount	number	选填	-	最大麦位数量 • roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制 • roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16
enableCDNStreaming	boolean	选填	false	是否开启 CDN 直播流

cdnStreamDomain	string	选填	''	直播推流域名

Returns *Promise<void>*

enterRoom

进入房间接口。

```
const roomEngine = new TUIRoomEngine();
const roomInfo = await roomEngine.enterRoom({
  roomId: '12345',
});
```

参数:

参数	类型	说明	默认值	含义
roomId	string	必填	-	房间号

Returns *Promise<TUIRoomInfo > roomInfo*
该接口返回当前房间信息

destroyRoom

解散房间接口，解散房间必须由房间所有者发起，解散房间之后房间不可进入。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.destroyRoom();
```

Returns *Promise<void>*

exitRoom

离开房间接口，用户在执行 enterRoom 之后可通过 exitRoom 离开房间。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.exitRoom();
```

Returns *Promise<void>*

fetchRoomInfo

获取房间信息。

```
const roomEngine = new TUIRoomEngine();
const roomInfo = roomEngine.fetchRoomInfo();
```

Returns: *Promise<TUIRoomInfo > roomInfo*

updateRoomNameByAdmin

更新当前房间的名字（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomNameByAdmin({ roomName: '新的名字' });
```

参数：

参数	类型	说明	默认值	含义
roomName	string	必填	-	更新房间的名字，要求 roomName 不为空字符串

Returns *Promise<void>*

updateRoomSpeechModeByAdmin

更新房间的发言模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSpeechModeByAdmin({
  speechMode: TUISpeechMode.kSpeakAfterTakingSeat  // 更新为上麦后发言模式
});
```

参数：

参数	类型	说明	默认值	含义
speechMode	TUISpeechMode	必填	-	更新房间的发言模式

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。

updateRoomSeatModeByAdmin

更新房间上麦模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSeatModeByAdmin({
  seatMode: TUISeatMode.kApplyToTake,  // 更新为申请上麦模式
});
```

参数：

参数	类型	说明	默认值	含义
seatMode	TUISeatMode	必填	-	发言类型

getUserList

获取当前房间用户列表，注意该接口一次拉取的用户列表量最大为 50 个。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
let result;
let nextSequence = 0;
do {
  result = await roomEngine.getUserList({ nextSequence });
  userList.push(...result.userInfoList);
  nextSequence = result.nextSequence;
} while (result.nextSequence !== 0)
```

参数：

参数	类型	说明	默认值	含义
nextSequence	number	选填	0	偏移量，默认从 0 开始拉取用户

Returns： *Promise<object> result*
result.nextSequence 下一次拉取群组用户的偏移量，如果 result.nextSequence 为 0 则代表 userList 全部拉取完毕。
result.userInfoList 本次拉取的 userList

getUserInfo

获取用户的详细信息。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
const userInfo = await roomEngine.getUserInfo({
   userId: 'user_12345',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	根据 userId 获取该用户的详细信息

Returns： *Promise<TUIUserInfo > userInfo*
该接口返回指定用户的用户信息

setLocalVideoView

设置本地流的渲染位置。

```
const roomEngine = new TUIRoomEngine();

// 设置本地摄像头流的播放区域为 id 是 'preview-camera' 的 div 元素
await roomEngine.setLocalVideoView({
  view: 'preview-camera',
});
```

参数：

参数	类型	说明	默认值	含义

view	string	必填	-	streamType 对应的流渲染的 div 元素的 id

Returns：*Promise<void>*

> **说明：**
> 自 v1.6.1 版本起，setLocalVideoView 只需传入 view 参数，设置本地视频流的播放容器。

openLocalCamera

打开本地摄像头，开始视频流采集。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});
await roomEngine.openLocalCamera();
```

Returns：*Promise<void>*

closeLocalCamera

关闭本地摄像头。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalCamera();
```

Returns：*Promise<void>*

openLocalMicrophone

打开本地麦克风, 开始采集音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.openLocalMicrophone();
```

Returns：*Promise<void>*

closeLocalMicrophone

关闭本地麦克风。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalMicrophone();
```

Returns：*Promise<void>*

updateVideoQuality

设置本地视频流的编码参数，默认为 TUIVideoProfile.kVideoQuality_720P。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQuality({
  quality: TUIVideoQuality.kVideoQuality_540p,
});
```

参数：

参数	类型	说明	默认值	含义
quality	TUIVideoQuality	必填	-	• 清晰 TUIVideoProfile.kVideoQuality_360P • 标清 TUIVideoProfile.kVideoQuality_540P • 高清 TUIVideoProfile.kVideoQuality_720P • 超清 TUIVideoProfile.kVideoQuality_1080P

Returns：*Promise<void>*

setVideoResolutionMode

设置本地视频流的分辨率模式，默认为 TUIResolutionMode.kResolutionMode_Landscape。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setVideoResolutionMode({
 type: TUIVideoStreamType.kCameraStream,
 resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
resolutionMode	TUIResolutionMode	必填	TUIResolutionMode.kResolutionMode_Landscape	分辨率模式

Returns：*Promise<void>*

> **说明：**
> 该接口自 v1.5.0 版本开始支持。

updateVideoQualityEx

设置本地视频流的编码参数，默认为 TUIVideoProfile.kVideoQuality_720P。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQualityEx({
 streamType: TUIVideoStreamType.kCameraStream,
 encoderParams: {
  videoResolution: TUIVideoQuality.kVideoQuality_720p,
  fps: 15,
  bitrate: 2000,
```

版权所有：腾讯云计算（北京）有限责任公司

```
    resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
  }
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
encoderParams	TUIVideoEncoderParams	必填	-	详细编码参数

说明：
该接口自 v1.5.0 版本开始支持。

updateAudioQuality

设置本地音频的参数。

注意：
该方法需要在 openLocalMicrophone 之前进行设置，否则不会生效。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateAudioQuality({
  quality: TUIAudioQuality.kAudioProfileMusic,
});
```

参数：

参数	类型	说明	默认值	含义
audioProfile	TUIAudioQuality	必填	-	• TUIAudioQuality.kAudioProfileSpeech：语言模式；采样率：16k • TUIAudioQuality.kAudioProfileDefault：标准模式（或者默认模式）；采样率：48k • TUIAudioQuality.kAudioProfileMusic：音乐模式；采样率：48k

Returns：*Promise<void>*

startPushLocalVideo

进房后会默认向远端推本地视频流，该接口用于停止推流后重新向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalVideo();
```

Returns：*Promise<void>*

stopPushLocalVideo

版权所有：腾讯云计算（北京）有限责任公司

停止向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalVideo();
```

Returns：*Promise<void>*

startPushLocalAudio

进房后会默认向远端推本地音频流，该接口用于停止推流后重新向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalAudio();
```

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v1.5.0 版本废弃，请使用 muteLocalAudio 方法代替。

stopPushLocalAudio

停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalAudio();
```

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v1.5.0 版本废弃，请使用 muteLocalAudio 方法代替。

muteLocalAudio

停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteLocalAudio();
```

Returns：*Promise<void>*

> ⓘ **说明：**
> 该接口自 v1.5.1 版本开始支持。

unmuteLocalAudio

开始向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.unmuteLocalAudio();
```

Returns：*Promise<void>*

> **说明：**
> 该接口自 v1.5.1 版本开始支持

setRemoteVideoView

设置远端流渲染的区域。

```
const roomEngine = new TUIRoomEngine();

// 设置远端用户视频流在 id 为 'remote_preview_camera' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'remote_preview_camera',
});
// 设置远端用户屏幕分享流在 id 为 'remote_preview_screen' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'remote_preview_screen',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型
view	string	必填	-	播放远端用户流的 div 元素的 id

Returns：*Promise<void>*

startPlayRemoteVideo

开始播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID

streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流 • TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns: *Promise<void>*

stopPlayRemoteVideo

停止播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

参数:

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流 • TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns: *Promise<void>*

muteRemoteAudioStream

停止远端用户的音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteRemoteAudioStream({
 userId: 'user_1234',
 isMute: true,
});
```

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isMute	boolean	必填	-	是否停止远端用户的音频

openRemoteDeviceByAdmin

请求远端用户打开媒体设备。

```
const roomEngine = new TUIRoomEngine();
```

```
const requestId = roomEngine.openRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
   switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
     // 请求被接受
     break;
    case TUIRequestCallbackType.kRequestRejected:
     // 请求被拒绝
     break;
    case TUIRequestCallbackType.kRequestCancelled:
     // 请求已取消
     break;
    case TUIRequestCallbackType.kRequestTimeout:
     // 请求超时
     break;
    case TUIRequestCallbackType.kRequestError:
     // 请求错误
     break;
    default:
     break;
   }
  },
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number；

applyToAdminToOpenLocalDevice

参会者向主持人申请打开设备。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.applyToAdminToOpenLocalDevice({
```

```
    device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
    timeout: 0,
    requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
     switch (requestCallbackType) {
       case TUIRequestCallbackType.kRequestAccepted:
         // 请求被接受
         break;
       case TUIRequestCallbackType.kRequestRejected:
         // 请求被拒绝
         break;
       case TUIRequestCallbackType.kRequestCancelled:
         // 请求已取消
         break;
       case TUIRequestCallbackType.kRequestTimeout:
         // 请求超时
         break;
       case TUIRequestCallbackType.kRequestError:
         // 请求错误
         break;
       default:
         break;
     }
    },
});
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

closeRemoteDeviceByAdmin

关闭远端用户媒体设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone, //关闭麦克风
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID

device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）

cancelRequest

取消已经发出的请求。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.cancelRequest({
  requestId: '',    // 请使用实际 requestId
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID

Returns： *Promise<string>* requestId
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

responseRemoteRequest

回复远端用户的请求。

```
const roomEngine = new TUIRoomEngine();
// 同意远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: true,
});
// 拒绝远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: false,
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID
agree	boolean	必填	-	是否同意

Returns： *Promise<void>*

> **说明：**

> v1.0.2 及其以上版本该接口接收的 requestId 参数类型为 string；v1.0.0 及 v1.0.1 版本该接口接收的 requestId 参数类型为 number。

disableDeviceForAllUserByAdmin

禁止/允许所有用户打开媒体设备（此接口对于房主和管理员无效）。

```
// Example 1: 禁止所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: true,
})
// Example 2: 允许所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: false,
})
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	被禁用的媒体设备类型（摄像头/麦克风/屏幕共享）
isDisable	boolean	必填	-	是否被禁止

Returns：*Promise<void>*

disableSendingMessageForAllUser

所有用户的是否允许发消息（此接口对于房主和管理员无效）。

```
await roomEngine.disableSendingMessageForAllUser({
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

disableSendingMessageByAdmin

特定用户是否被允许发消息。

```
await roomEngine.disableSendingMessageByAdmin({
  userId: 'user_1234',
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

changeUserRole

改变用户的角色（仅主持人可调用该接口）。

```
const roomEngine = new TUIRoomEngine();

// 将房间移交给用户 user_1234
await roomEngine.changeUserRole({
  userId: 'user_1234',
  role: TUIRole.kRoomOwner,
});

// 将用户 user_1234 设置为房间管理员
await roomEngine.changeUserRole({
 userId: 'user_1234',
 userRole: TUIRole.kAdministrator,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
userRole	TUIRole	必填	-	用户角色 • 主持人 TUIRole.kRoomOwner • 管理员 TUIRole.kAdministrator • 普通成员 TUIRole.kGeneralUser

Returns：*Promise<void>*

kickRemoteUserOutOfRoom

将用户踢出房间（仅主持人和管理员可调用该接口）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.kickRemoteUserOutOfRoom({
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID

Returns：*Promise<void>*

setMaxSeatCount

设置房间麦位最大值
roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制；
roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16；

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.setMaxSeatCount({ maxSeatCount: 16 })
```

参数：

参数	类型	说明	默认值	含义
maxSeatCount	number	必填	-	设置房间麦位最大值

Returns：Promise<void>

getSeatList

获取麦位列表

```
const roomEngine = new TUIRoomEngine();
const seatList = await roomEngine.getSeatList();
```

Returns：*Promise<*TUISeatInfo *[]>* seatList
seatList 为当前房间所有麦位列表

takeSeat

麦下用户可调用 takeSeat 成为麦上用户，仅麦上用户可发布本地音视频流。

- **当 roomInfo.roomType 为 TUIRoomType.kConference 且 roomInfo.seatMode 为 TUISeatMode.kApplyToTake 时，普通用户调用 takeSeat 方法需要等待主持人/管理员的同意后成为麦上用户。**
- **当 roomInfo.roomType 为 TUIRoomType.kLivingRoom 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 时，普通用户调用 takeSeat 方法成功后即为麦上用户。**
- **主持人&管理员调用 takeSeat 成功后即为麦上用户。**
- **麦上用户的变更通过 TUIRoomEvents.onSeatListChanged 通知所有用户。**

```
const roomEngine = new TUIRoomEngine();
// 情景一：主持人/管理员上麦
// 情景二：当 roomInfo.roomType 为 TUIRoomType.kConference
// 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 模式时，普通用户上麦
await roomEngine.takeSeat({
 seatIndex: -1,
 timeout: 0,
});

// 情景三：当 roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户上麦
const requestId = await roomEngine.instance?.takeSeat({
  seatIndex: -1,
```

```
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
    switch (requestCallbackType) {
     case TUIRequestCallbackType.kRequestAccepted:
       // 请求被接受
       break;
     case TUIRequestCallbackType.kRequestRejected:
       // 请求被拒绝
       break;
     case TUIRequestCallbackType.kRequestCancelled:
       // 请求已取消
       break;
     case TUIRequestCallbackType.kRequestTimeout:
       // 请求超时
       break;
     case TUIRequestCallbackType.kRequestError:
       // 请求错误
       break;
     default:
       break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index，无麦位序号时设置为 -1
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：Promise<string> requestId
roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户调用该接口时返回 requestId，普通用户可使用该 requestId 调用 cancelRequest 接口取消上麦请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

leaveSeat

释放麦位。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.leaveSeat();
```

Returns：*Promise<void>*

takeUserOnSeatByAdmin

邀请其他人上麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.takeUserOnSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
   switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // 请求被接受
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // 请求被拒绝
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // 请求已取消
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // 请求超时
      break;
    case TUIRequestCallbackType.kRequestError:
      // 请求错误
      break;
    default:
      break;
   }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID
timeout	number	必填	-	超时时间。若 timeout 设置为 0 ，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

> **说明：**
> v1.0.2 及其以上版本该接口返回的 requestId 类型为 string；v1.0.0 及 v1.0.1 版本该接口返回的 requestId 类型为 number。

kickUserOffSeatByAdmin

要求其他人下麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = await roomEngine.kickUserOffSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID

Returns： *Promise<void>*

lockSeatByAdmin

锁定某个麦位状态（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.lockSeatByAdmin({
  seatIndex: 0,
  lockParams: {
    lockSeat: true,
    lockVideo: true,
    lockAudio: true,
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位索引
lockParams	TUISeatLockParams	必填	-	锁麦参数

Returns：*Promise<void>*

getSeatApplicationList

获取房间内申请上麦用户的请求列表, 该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
const applicationList = await roomEngine.getSeatApplicationList();
```

Returns：*Promise*<TUIRequest[]>

startScreenSharingElectron

开始屏幕共享 Electron。

```
const roomEngine = new TUIRoomEngine();
// 情景一：开始屏幕共享
await roomEngine.startScreenSharingElectron({
  targetId: 'targetId'
});

// 情景二：开始屏幕共享并在本地预览
await roomEngine.startScreenSharingElectron({
  targetId: 'targetId',
  view: 'screen-preview'
});
```

参数：

参数	类型	说明	默认值	含义
targetId	string	选填	-	分享窗口 ID，可从 getScreenSharingTarget 获取
view	string	选填	-	本地预览屏幕分享的 dom 元素的 id

Returns： *Promise<void>*

stopScreenSharingElectron

停止屏幕共享 Electron。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopScreenSharingElectron();
```

Returns： *Promise<void>*

getScreenSharingTarget

获取屏幕分享列表 Electron。

```
const roomEngine = new TUIRoomEngine();
const screenList = await roomEngine.getScreenSharingTarget();
```

Returns： *Promise<void>*

selectScreenSharingTarget

切换屏幕分享窗口 Electron。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.selectScreenSharingTarget({
  targetId: 'targetId'
});
```

参数：

参数	类型	说明	默认值	含义

targetId	string	必填	-	分享窗口 ID，可从 getScreenSharingTarget 获取

Returns：*Promise<void>*

sendTextMessage

发送文本消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendTextMessage({
 messageText: 'hello, everyone',
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	文本消息内容

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

sendCustomMessage

发送自定义消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendCustomMessage({
 messageText: '{ data:'', description: ''}',
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	自定义消息内容

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

getCameraDevicesList

获取摄像头设备列表。

```
const roomEngine = new TUIRoomEngine();
const cameraList = await roomEngine.getCameraDevicesList();
for (i = 0; i < cameraList.length; i++) {
  var camera = cameraList[i];
```

```
  console.info("camera deviceName: " + camera.deviceName + " deviceId:" + camera.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> cameraList

getMicDevicesList

获取麦克风设备列表。

```
const roomEngine = new TUIRoomEngine();
const micList = await roomEngine.getMicDevicesList();
for (i = 0; i < micList.length; i++) {
  var mic = micList[i];
  console.info("mic deviceName: " + mic.deviceName + " deviceId:" + mic.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> micList

getSpeakerDevicesList

获取扬声器设备列表。

```
const roomEngine = new TUIRoomEngine();
const speakerList = await roomEngine.getSpeakerDevicesList();
for (i = 0; i < speakerList.length; i++) {
  var speaker = speakerList[i];
  console.info("speaker deviceName: " + speaker.deviceName + " deviceId:" + speaker.deviceId);
}
```

Returns: *Promise*<TRTCDeviceInfo *[]*> speakerList

setCurrentCameraDevice

设置要使用的摄像头设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentCameraDevice({ deviceId: '' });
```

参数:

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getCameraDevicesList 中得到的设备 ID

Returns: *void*

setCurrentMicDevice

设置要使用的麦克风设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentMicDevice({ deviceId: '' });
```

参数：

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getMicDevicesList 中得到的设备 ID

Returns: *void*

setCurrentSpeakerDevice

设置要使用的扬声器设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setCurrentSpeakerDevice({ deviceId: '' });
```

参数：

参数	类型	说明	默认值	含义
deviceId	string	必填	-	从 getSpeakerDevicesList 中得到的设备 ID

Returns: *void*

getCurrentCameraDevice

获取当前正在使用的摄像头设备。

```
const roomEngine = new TUIRoomEngine();
const currentCameraDevice = roomEngine.getCurrentCameraDevice();
```

Returns: TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

getCurrentMicDevice

获取当前正在使用的麦克风设备。

```
const roomEngine = new TUIRoomEngine();
const currentMicDevice = roomEngine.getCurrentMicDevice();
```

Returns: TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

getCurrentSpeakerDevice

获取当前正在使用的扬声器设备。

```
const roomEngine = new TUIRoomEngine();
const currentSpeakerDevice = roomEngine.getCurrentSpeakerDevice();
```

Returns: TRTCDeviceInfo
设备信息，能获取设备 ID 和设备名称。

版权所有：腾讯云计算（北京）有限责任公司

startCameraDeviceTest

开始进行摄像头测试。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startCameraDeviceTest({ view: 'test-preview' });
```

参数：

参数	类型	说明	默认值	含义
view	string	必填	-	显示摄像头测试的视频区域，传入的 view 为承载预览画面 div 元素的 Id

Returns： *Promise<void>*

stopCameraDeviceTest

停止摄像头测试。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopCameraDeviceTest();
```

Returns： *Promise<void>*

on

监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(event, func);
```

参数：

参数	类型	说明	默认值	含义
event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns： *void*

off

取消监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.off(event, func);
```

参数：

参数	类型	说明	默认值	含义

event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns: *void*

getTRTCCloud

获取 trtcCloud 实例，web 端 trtcCloud 能力请查看： TRTCCloud API 文档 。

```
const roomEngine = new TUIRoomEngine();
const trtcCloud = roomEngine.getTRTCCloud();
```

Returns: TRTCCloud

getTIM

获取 tim 实例，web 端 tim 能力请查看： IM API 文档 。

```
const roomEngine = new TUIRoomEngine();
const tim = roomEngine.getTIM();
```

Returns: tim

> ⚠ **注意：**
> 自 v1.6.0 版本后，IM SDK 升级至 v3 版本，请查看： IM API 文档 。

TUIRoomEvent

最近更新时间：2024-03-04 14:32:51

TUIRoomEvent API 简介

TUIRoomEvent API 是多人组件的**事件接口**。

事件列表

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	踢出房间事件
TUIRoomEvents. onKickedOffLine	当前用户被踢下线
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomNameChanged	房间名称修改事件
~~TUIRoomEvents.onRoomSpeechModeChanged~~	房间发言模式修改事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onAllUserMicrophoneDisableChanged	所有成员麦克风使用权限改变事件
TUIRoomEvents.onSendMessageForAllUserDisableChanged	所有成员发送消息状态改变事件
TUIRoomEvents.onRoomMaxSeatCountChanged	房间最大麦位数修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChanged	用户音频状态改变事件
TUIRoomEvents.onSendMessageForUserDisableChanged	用户发送消息状态事件
TUIRoomEvents.onUserVoiceVolumeChanged	用户音量改变事件
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件
TUIRoomEvents.onSeatListChanged	麦位列表改变事件

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEvents.onKickedOffSeat	用户被踢下麦事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
~~TUIRoomEvents.onReceiveCustomMessage~~	接收自定义消息事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onDeviceChange	设备变更事件
TUIRoomEvents.onUserScreenCaptureStopped	屏幕分享停止事件，当用户使用浏览器自带的**停止分享**按钮结束屏幕共享时，该用户会收到 'onUserScreenCaptureStopped' 事件用来修改屏幕共享状态。

onError

错误事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onError, (error) => {
 console.log('TUIRoomError error', error);
})
```

参数如下表所示：

参数	类型	含义
code	number	错误代码
message	string	错误信息

onKickedOutOfRoom

踢出房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOutOfRoom, ({ roomId, reason, message }) => {
  console.log('roomEngine.onKickedOutOfRoom', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
reason	TUIKickedOutOfRoomReason	用户被踢出房间原因
message	string	踢出房间信息

onKickedOffLine

当前用户被踢下线

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffLine, ({ message }) => {
  console.log('roomEngine.onKickedOffLine', message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
message	string	用户在其他端登录信息

onUserSigExpired

userSig 过期事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserSigExpired, () => {
  console.log('roomEngine.onUserSigExpired');
});
```

onRoomDismissed

主持人销毁房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomDismissed, ({ roomId }) => {
  console.log('roomEngine.onRoomDismissed', roomId);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号

onRoomInfoChanged

房间信息修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomInfoChanged, ({ roomId, roomName }) => {
  console.log('roomEngine.onRoomInfoChanged', roomId, roomName);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号

roomName	string	房间名称

onRoomSpeechModeChanged

房间名称修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSpeechModeChanged, ({ roomId, speechMode }) => {
  console.log('roomEngine.onRoomSpeechModeChanged', roomId, speechMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
speechMode	TUISpeechMode	发言模式

> ⚠ 注意：
> 该接口自 v2.0.0 版本废弃

onRoomSeatModeChanged

上麦模式修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSeatModeChanged, ({ roomId, seatMode }) => {
  console.log('roomEngine.onRoomSeatModeChanged', roomId, seatMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
seatMode	TUISeatMode	房间上麦模式

onAllUserCameraDisableChanged

所有成员摄像头使用权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserCameraDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserCameraDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

isDisable	boolean	是否允许使用摄像头

onAllUserMicrophoneDisableChanged

所有成员麦克风使用权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserMicrophoneDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserMicrophoneDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许使用麦克风

onSendMessageForAllUserDisableChanged

所有成员发送消息权限改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onRoomMaxSeatCountChanged

房间最大麦位数修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomMaxSeatCountChanged, ({ maxSeatNumber }) => {
  console.log('roomEngine.onRoomMaxSeatCountChanged', maxSeatNumber);
});
```

参数如下表所示：

参数	类型	含义
maxSeatNumber	number	最大麦位数量

onRemoteUserEnterRoom

远端用户进入房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserEnterRoom, ({ roomId, userInfo }) => {
```

```
  console.log('roomEngine.onRemoteUserEnterRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onRemoteUserLeaveRoom

远端用户离开房间事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserLeaveRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserLeaveRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onUserRoleChanged

用户角色改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserRoleChanged, ({ userId, userRole }) => {
  console.log('roomEngine.onUserRoleChanged', userId, userRole);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
userRole	TUIRole	用户变更后的角色

onUserMuteStateChanged

用户禁止发文字消息状态改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserMuteStateChanged, ({ userId, muted }) => {
  console.log('roomEngine.onUserMuteStateChanged', userId, muted);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
muted	boolean	是否被禁止发文字消息

onUserVideoStateChanged

用户视频状态改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVideoStateChanged, ({ userId, streamType, hasVideo, reason }) => {
  console.log('roomEngine.onUserVideoStateChanged', userId, streamType, hasVideo, reason);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
streamType	TUIVideoStreamType	用户流类型
hasVideo	boolean	是否有视频流
reason	TUIChangeReason	变更原因, 自己操作/主持人操作

onUserAudioStateChanged

用户音频状态改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserAudioStateChanged, ({ userId, hasAudio, reason }) => {
  console.log('roomEngine.onUserAudioStateChanged', userId, hasAudio, reason);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
hasVideo	boolean	是否有音频流
reason	TUIChangeReason	变更原因，自己操作/主持人操作

onSendMessageForUserDisableChanged

有成员摄像头使用权限修改事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
```

版权所有：腾讯云计算（北京）有限责任公司

```
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onUserVoiceVolumeChanged

用户音量改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVoiceVolumeChanged, ({ userVolumeList }) => {
  userVolumeList.forEach(userVolume => {
    console.log('roomEngine.onUserVoiceVolumeChanged', userVolume.userId, userVolume.volume);
  })
});
```

参数如下表所示：

参数	类型	含义
userVolumes	Array.<TRTCVolumeInfo>	房间内所有用户的音量, 包含 userId 及 volume 信息，volume 区间为 1~100;

onUserNetworkQualityChanged

用户网络质量改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserNetworkQualityChanged, ({ userNetworkList }) => {
  userNetworkList.forEach(userNetwork => {
    console.log('roomEngine.onUserNetworkQualityChanged', userNetwork.userId, userNetwork.quality,
userNetwork.upLoss, userNetwork.downLoss, userNetwork.delay);
  })
});
```

参数如下表所示：

参数	类型	含义
networkMap	Array.<TUINetwork>	遍历网络质量等级

onSeatControlEnabled

麦位控制启用事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatControlEnabled, ({ enabled, maxSeatNumber }) => {
  console.log('roomEngine.onSeatControlEnabled', enabled, maxSeatNumber);
});
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
enabled	boolean	是否启用
maxSeatNumber	number	最大麦位数量

onSeatListChanged

麦位列表改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatListChanged, ({ seatList, seatedList, leftList }) => {
  console.log('roomEngine.onSeatListChanged',seatList, seatedList, leftList);
});
```

参数如下表所示：

参数	类型	含义
seatList	Array.<TUISeatInfo>	麦位列表
seatedList	Array.<TUISeatInfo>	新增麦位信息
leftList	Array.<TUISeatInfo>	离开的麦位信息

onKickedOffSeat

麦位列表改变事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffSeat, ({ userId }) => {
  console.log('roomEngine.onKickedOffSeat', userId);
});
```

参数如下表所示：

参数	类型	含义
userId	String	被踢下麦的用户 ID

onRequestReceived

错误事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestReceived, ({ request }) => {
  console.log('roomEngine.onRequestReceived', request);
});
```

参数如下表所示：

参数	类型	含义

request	TUIRequest	请求接收

onRequestProcessed

请求被处理事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestProcessed, ({ request }) => {
  console.log('roomEngine.onRequestProcessed', request);
});
```

参数如下表所示：

参数	类型	含义
request	TUIRequest	请求接收

onRequestCancelled

请求取消事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestCancelled, ({ requestId, userId }) => {
  console.log('roomEngine.onRequestCancelled', requestId, userId);
});
```

参数如下表所示：

参数	类型	含义
requestId	string	请求 ID
userId	string	取消请求的用户 ID

onReceiveTextMessage

接收文本消息事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveTextMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveTextMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收文本消息

⚠ **注意：**

该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

onReceiveCustomMessage

接收自定义消息事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveCustomMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveCustomMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收自定义消息

注意：
该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

onDeviceChange

设备变更事件

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onDeviceChange, ({ deviceId, type, state }) => {
  console.log('roomEngine.onReceiveCustomMessage', deviceId, type, state);
});
```

参数如下表所示：

参数	类型	含义
deviceId	string	设备 ID
type	TRTCDeviceType	设备类型
state	TRTCDeviceState	设备变更状态

onUserScreenCaptureStopped

屏幕分享停止事件，当用户使用浏览器自带的 **停止分享** 按钮结束屏幕共享时，该用户会收到 `onUserScreenCaptureStopped` 事件用来修改屏幕共享状态。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserScreenCaptureStopped, () => {
  console.log('roomEngine.onReceiveCustomMessage', deviceId, type, state);
});
```

TUIRoomEngine Defines

最近更新时间：2024-02-21 17:54:32

本文介绍 TUIRoomEnigne Electron 端关键类型定义。

枚举值

TUIRole

用户角色，TUIRoomEngine 共提供三种用户角色，分别是主持人，管理员，普通用户。

字段	类型	描述
kRoomOwner	number	主持人角色
kAdministrator	number	管理员角色
kGeneralUser	number	普通用户角色

TUIVideoQuality

视频分辨率。

字段	类型	描述
kVideoQuality_360p	number	低清，分辨率为 640 * 360
kVideoQuality_540p	number	标清，分辨率为 960 * 540
kVideoQuality_720p	number	高清，分辨率为 1280 * 720
kVideoQuality_1080p	number	超清，分辨率为 1920 * 1080

TUIResolutionMode

分辨率模式（横屏分辨率 | 竖屏分辨率）。

字段	类型	描述
kResolutionMode_Landscape	number	横屏模式
kResolutionMode_Portrait	number	竖屏模式

TUIAudioQuality

音频分辨率。

字段	类型	描述
kAudioProfileSpeech	number	语音模式
kAudioProfileDefault	number	标准模式（默认模式）
kAudioProfileMusic	number	音乐模式

版权所有：腾讯云计算（北京）有限责任公司

TUIVideoStreamType

视频流类型。

字段	类型	描述
kCameraStream	number	摄像头视频流
kScreenStream	number	屏幕分享视频流
kCameraStreamLow	number	低清摄像头视频流

TUINetworkQuality

网络状态。

字段	类型	描述
kQualityUnknown	number	网络状况未知
kQualityExcellent	number	网络状况极佳
kQualityGood	number	网络状况较好
kQualityPoor	number	网络状况一般
kQualityBad	number	网络状况差
kQualityVeryBad	number	网络状况极差
kQualityDown	number	网络连接已断开

TUIRoomType

房间类型。

字段	类型	描述
kConference	number	kConference 类型房间，适用于会议、教育场景
kLivingRoom	number	kLivingRoom 类型房间，适用于直播场景

TUISpeechMode

发言类型。

字段	类型	描述
kFreeToSpeak	number	自由发言模式
kApplyToSpeak	number	举手发言模式
kSpeakAfterTakingSeat	number	就坐后发言（抢麦位）

> **注意：**
> 该类型自 v2.0.0 版本废弃

版权所有：腾讯云计算（北京）有限责任公司

TUISeatMode

上麦模式

字段	类型	描述
kFreeToTake	number	自由上麦模式，台下观众可以自由上麦，无需申请
kApplyToTake	number	申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦

TUICaptureSourceType

屏幕分享类型。

字段	类型	描述
kWindow	number	该分享目标是某一个 Windows 或 Mac 窗口 todo(只有 electron)
kScreen	number	该分享目标是整个 Windows 桌面或 Mac 桌面

TUIChangeReason

更改原因（用户音视频状态变更操作原因：自己主动修改 或者 被房主、管理员修改）。

字段	类型	描述
kChangedBySelf	number	自己操作
kChangedByAdmin	number	房主或管理员操作

TUIMediaDevice

字段	类型	描述
kMicrophone	number	麦克风
kCamera	number	摄像头
kScreen	number	屏幕分享

TUIRequestAction

房间类型。

字段	类型	描述
kInvalidAction	number	无效操作
kRequestToOpenRemoteCamera	number	请求远端打开摄像头
kRequestToOpenRemoteMicrophone	number	请求远端打开麦克风
kRequestToConnectOtherRoom	number	请求远端夸房连麦，web 端暂不支持

版权所有：腾讯云计算（北京）有限责任公司

kRequestToTakeSeat	number	请求上麦
kRequestRemoteUserOnSeat	number	请求远端上麦

TUIRequestCallbackType

请求类型。

字段	类型	描述
kRequestAccepted	number	对端已接受
kRequestRejected	number	对端已拒绝
kRequestCancelled	number	请求已取消
kRequestTimeout	number	请求超时
kRequestError	number	请求错误

TUIKickedOutOfRoomReason

用户被移出房间原因。

字段	类型	描述
kKickedByAdmin	number	被主持人或管理员移出
kKickedByLoggedOnOtherDevice	number	相同 userId 用户进入同一房间被移出
kKickedByServer	number	被服务端移出

类型定义

TUILoginUserInfo

当前登录用户信息。

名称	类型	描述
userId	string	登录用户的 ID
userName	string	登录用户的名称
avatarUrl	string	登录用户的头像

TUIRoomInfo

房间信息，用户可使用 roomEngine.fetchRoomInfo 获取房间信息。

名称	类型	描述
roomId	string	房间号，字符串类型房间号
roomName	string	房间名称
roomType	TUIRoomType	房间类型

版权所有：腾讯云计算（北京）有限责任公司

isSeatEnabled	boolean	是否开启麦位控制（创建房间可选参数，默认值为 false）
seatMode	TUISeatMode	房间内上麦模式（开启麦位控制后生效，默认值为 TUISeatMode.kFreeToTake）
isMicrophoneDisableForAllUser	boolean	是否开启全员禁音（创建房间可选参数，默认值为 false）
isCameraDisableForAllUser	boolean	是否开启全员禁画（创建房间可选参数， 默认值为 false）
isMessageDisableForAllUser	boolean	是否允许所有用户发消息（创建房间可选参数，默认值为 false）
maxSeatCount	number	最大麦位数量，默认为 6
roomOwner	string	房间主持人，只读
createTime	number	房间创建时间，只读，精确到秒
roomMemberCount	number	房间成员数量，只读

TUIUserInfo

用户信息。

名称	类型	描述
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址
userRole	TUIRole	用户角色
hasAudioStream	boolean	是否有音频流
hasVideoStream	boolean	是否有视频流
hasScreenStream	boolean	是否有屏幕分享流

TUIMessage

消息信息。

名称	类型	描述
messageId	string	消息 ID
message	string	消息
timestamp	number	时间戳信息, 精确到秒
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址

版权所有：腾讯云计算（北京）有限责任公司

TUIRequest

请求信息。

名称	类型	描述
requestAction	TUIRequestAction	请求类型
timestamp	number	请求发起时间
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	发起请求的用户 ID
content	string	其他内容

TUIRequestCallback

请求回调信息。

名称	类型	描述
requestCallbackType	TUIRequestCallbackType	请求回调类型，接受/拒绝/取消/超时/错误
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	用户 ID
code	number	请求响应码
message	string	请求状态补充说明

TUISeatInfo

麦位信息。

名称	类型	描述
index	number	麦位序号
userId	string	麦位对应的用户 ID
locked	boolean	当前麦位是否被锁
videoMuted	boolean	当前麦位是否禁止视频
audioMuted	boolean	当前麦位是否禁止音频

TUISeatLockParams

麦位锁定状态。

名称	类型	描述

版权所有：腾讯云计算（北京）有限责任公司

lockSeat	boolean	锁定麦位
lockVideo	boolean	锁定麦位视频
lockAudio	boolean	锁定麦位音频

TUINetwork

网络信息。

名称	类型	描述
userId	string	用户 ID
quality	TUINetworkQuality	网络质量
upLoss	number	上行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
downLoss	number	下行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
delay	number	网络延迟，单位 ms，目前仅本地用户有该信息

TUIVideoEncoderParams

视频编码参数。

名称	类型	描述
videoResolution	TUIVideoQuality	视频分辨率, 该字段在 v1.6.0 版本由 quality 修改为 videoResolution
fps	number	视频帧率
bitrate	number	视频码率
resolutionMode	TUIResolutionMode	横竖屏模式

Flutter
API 概览

最近更新时间：2023-09-28 17:12:31

TUIRoomEngine API 列表

TUIRoomEngine API 是多人音视频房间的无 UI 接口，您可以使用这套 API 根据您的业务需求自定义封装。

TUIRoomEngine

TUIRoomEngine 核心方法

API	描述
createInstance	创建 TUIRoomEngine 实例。
destroyInstance	销毁 TUIRoomEngine 实例。
login	登录接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作。
logout	退出登录接口，会有主动离开房间操作、销毁资源。
setSelfInfo	设置本地用户名称和头像。
setLoginUserInfo	设置登录用户信息。
getSelfInfo	获取本地用户登录的基本信息。
addObserver	设置事件回调。
removeObserver	移除事件回调。

房间相关主动接口

API	描述
createRoom	创建房间。
destroyRoom	解散房间。
enterRoom	进入房间。
exitRoom	离开房间。
connectOtherRoom	连接其他房间。
disconnectOtherRoom	断开与其他房间的连接。
fetchRoomInfo	获取房间信息。
updateRoomNameByAdmin	更新房间名称（只有管理员或群主能够调用）。
updateRoomSpeechModeByAdmin	设置房间管理模式（只有管理员或群主能够调用）。

版权所有：腾讯云计算（北京）有限责任公司

本地用户视图渲染、视频管理

API	描述
setLocalVideoView	设置本地用户视频渲染的控件。
openLocalCamera	打开本地摄像头。
closeLocalCamera	关闭本地摄像头。
updateVideoQuality	更新本地视频编码质量设置。
updateVideoQualityEx	设置视频编码器的编码参数。
setVideoResolutionMode	设置视频编码器分辨率模式。
enableGravitySensor	开启重力感应模式。
startPushLocalVideo	开始推送本地视频。
stopPushLocalVideo	停止推送本地视频。
startScreenSharing	开始屏幕分享。
stopScreenSharing	结束屏幕分享。

本地用户音频管理

API	描述
openLocalMicrophone	打开本地麦克风。
closeLocalMicrophone	关闭本地麦克风。
updateAudioQuality	更新本地音频编码质量设置。
muteLocalAudio	停止推送本地音频。
unMuteLocalAudio	开始推送本地音频。

远端用户视图渲染、视频管理

API	描述
setRemoteVideoView	设置远端用户视频渲染的控件。
startPlayRemoteVideo	开始播放远端用户视频。
stopPlayRemoteVideo	停止播放远端用户视频。
muteRemoteAudioStream	将远端用户禁音。

房间内用户信息

API	描述
getUserList	获取房间内的成员列表。

版权所有：腾讯云计算（北京）有限责任公司

getUserInfo	获取成员信息。

房间内用户管理

API	描述
changeUserRole	修改用户角色（只有管理员或群主能够调用）。
kickRemoteUserOutOfRoom	将远端用户移出房间（只有管理员或群主能够调用）。
addCategoryTagForUsers	为用户增加标记（只有房主能够调用）。
removeCategoryTagForUsers	为用户移除标记（只有房主能够调用）。
getUserListByTag	根据标记获取房间内用户信息。

房间内用户发言管理

API	描述
disableDeviceForAllUserByAdmin	控制当前房间内所有用户是否可打开音频流、视频流采集设备的权限状态，例如：全员禁止打开麦克风、全员禁止打开摄像头、全员禁止打开屏幕分享（目前仅会议场景下可用，并且只有管理员或群主能够调用）。
openRemoteDeviceByAdmin	请求远端用户打开媒体设备（只有管理员或群主能够调用）。
closeRemoteDeviceByAdmin	关闭远端用户媒体设备（只有管理员或群主能够调用）。
applyToAdminToOpenLocalDevice	请求打开本地媒体设备（普通用户可用）。

房间内麦位管理

API	描述
setMaxSeatCount	设置最大麦位数（仅支持进房前和创建房间时设置）。
getSeatList	获取麦位列表。
lockSeatByAdmin	锁定麦位（只有管理员或群主能够调用，包括位置锁定、音频状态锁定和视频状态锁定）。
takeSeat	申请上麦（自由发言模式下，不需要申请）。
leaveSeat	申请下麦（自由发言模式下，不需要申请）。
takeUserOnSeatByAdmin	主持人/管理员 邀请用户上麦。
kickUserOffSeatByAdmin	主持人/管理员 将用户移下麦。

信令管理

API	描述
cancelRequest	取消请求。
responseRemoteRequest	回复请求。

版权所有：腾讯云计算（北京）有限责任公司

发送消息

API	描述
sendTextMessage	发送本文消息。
sendCustomMessage	发送自定义消息。
disableSendingMessageByAdmin	禁用远端用户的发送文本消息能力（只有管理员或群主能够调用）。
disableSendingMessageForAllUser	禁用所有用户的发送文本消息能力（只有管理员或群主能够调用）。

高级功能

API	描述
switchCamera	切换前置或后置摄像头
setBeautyLevel	设置美颜级别
setWhitenessLevel	设置美白级别

调试相关

API	描述
callExperimentalAPI	调用实验性接口

TUIRoomObserver 回调事件

TUIRoomObserver 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己需要的回调事件。

TUIRoomObserver

TUIRoomObserver

错误回调

API	描述
onError	错误事件回调。

登录状态事件回调

API	描述
onKickedOffLine	其他终端登录被移下线。
onUserSigExpired	用户凭证超时事件。

房间内事件回调

API	描述
onRoomNameChanged	房间名称更改事件。
onAllUserMicrophoneDisableChanged	房间内所有用户麦克风被禁用事件。
onAllUserCameraDisableChanged	房间内所有用户摄像头被禁用事件。
onSendMessageForAllUserDisableChanged	房间内所有用户发送文本消息被禁用事件。
onKickedOutOfRoom	被移出房间事件。
onRoomDismissed	房间被解散事件。
onRoomSpeechModeChanged	房间麦控模式发生变化

房间内用户事件回调

API	描述
onRemoteUserEnterRoom	远端用户进房事件。
onRemoteUserLeaveRoom	远端用户离开房间事件。
onUserRoleChanged	用户角色发生变化事件。
onUserVideoStateChanged	用户视频状态发生变化事件。
onUserAudioStateChanged	用户音频状态发生变化事件。
onUserVoiceVolumeChanged	用户音量变化事件。
onSendMessageForUserDisableChanged	用户文本消息发送能力发生变化事件。
onUserNetworkQualityChanged	用户网络状态变化事件。
onUserScreenCaptureStopped	屏幕分享结束。

房间内麦位事件回调

API	描述
onRoomMaxSeatCountChanged	房间内最大麦位数发生变化事件（仅在会议类型房间生效）。
onSeatListChanged	麦位列表发生变化事件。
onKickedOffSeat	收到用户被移下麦事件。

请求信令事件回调

API	描述
onRequestReceived	收到请求消息事件。

版权所有：腾讯云计算（北京）有限责任公司

onRequestCancelled	收到请求被取消的事件。

房间内消息事件回调

API	描述
onReceiveTextMessage	收到普通文本消息事件。
onReceiveCustomMessage	收到自定义消息事件。

TUIRoomEngine

最近更新时间：2023-11-17 11:00:42

TUIRoomEngine API 简介

TUIRoomEngine API 是多人音视频房间的无 UI 接口，您可以使用这套 API 根据您的业务需求自定义封装。

createInstance

创建 TUIRoomEngine 实例。

```
static TUIRoomEngine createInstance()
```

return:TUIRoomEngine 实例。

destroyInstance

销毁TUIRoomEngine实例。

```
void destroyInstance()
```

login

登录 roomEngine 接口，您需要先初始化用户信息后才能进入房间，并进行一系列的操作。

```
static Future<TUIActionCallback> login(int sdkAppId,
                        String userId,
                        String userSig)
```

参数：

参数	类型	含义
sdkAppId	int	应用信息中获取 sdkAppId 信息
userId	String	用户 ID
userSig	String	userSig 签名。计算 userSig 的方式请参考 UserSig 相关

logout

退出登录接口，会有主动离开房间操作、销毁资源。

```
static Future<TUIActionCallback> logout()
```

setSelfInfo

设置本地用户名称和头像。

```
static Future<TUIActionCallback> setSelfInfo(String userName, String avatarURL)
```

版权所有：腾讯云计算（北京）有限责任公司

参数:

参数	类型	含义
userName	String	用户名
avatarUrl	String	用户头像

setLoginUserInfo

设置登录用户信息。

```
static Future<TUIActionCallback> setLoginUserInfo(TUILoginUserInfo userInfo)
```

参数:

参数	类型	含义
userInfo	TUILoginUserInfo	用户信息

getSelfInfo

获取本地用户登录的基本信息。

```
static TUILoginUserInfo getSelfInfo()
```

return:用户登录信息。

addObserver

添加 TUIRoomEngine 事件回调。

```
void addObserver(TUIRoomObserver observer)
```

参数:

参数	类型	含义
observer	TUIRoomObserver	TUIRoomEngine 事件回调

removeObserver

移除 TUIRoomEngine 事件回调。

```
void removeObserver(TUIRoomObserver observer)
```

参数	类型	含义
observer	TUIRoomObserver	TUIRoomEngine 事件回调

createRoom

主持人创建房间，调用 createRoom 的用户为房间的所有者。创建房间时可设置房间 ID、房间名称、以及房间是否允许加入用户开启音视频、发送消息等功能。

```
Future<TUIActionCallback> createRoom(TUIRoomInfo roomInfo)
```

参数：

参数	类型	含义
roomInfo	TUIRoomInfo	房间基本信息

destroyRoom

销毁房间接口，销毁房间必须由房间所有者发起，销毁房间之后房间不可进入。

```
Future<TUIActionCallback> destroyRoom()
```

enterRoom

进入房间接口。

```
Future<TUIValueCallBack<TUIRoomInfo>> enterRoom(String roomId)
```

参数：

参数	类型	含义
roomId	String	房间号，字符串类型

exitRoom

离开房间接口，用户在执行 enterRoom 之后可通过 exitRoom 离开房间。

```
Future<TUIActionCallback> exitRoom(bool syncWaiting)
```

参数：

参数	类型	含义
syncWaiting	bool	是否同步退出房间

connectOtherRoom

连接其他房间。

> **说明：**
> 用于直播场景下的申请跨房连麦。

```
TUIRequest connectOtherRoom(String roomId,
                            String userId,
                            int timeout,
```

```
                TUIRequestCallback? requestCallback)
```

参数：

参数	类型	含义
roomId	String	房间 ID
userId	String	用户 ID
timeout	int	时间
callback	TUIRequestCallback	连接其他房间请求回调

Return：请求体

disconnectOtherRoom

断开与其他房间的连接。

> **说明：**
> **用于直播场景下的断开跨房连麦。**

```
Future<TUIActionCallback> disconnectOtherRoom()
```

fetchRoomInfo

获取房间信息。

```
Future<TUIValueCallBack<TUIRoomInfo>> fetchRoomInfo()
```

updateRoomNameByAdmin

更新房间名称。

```
Future<TUIActionCallback> updateRoomNameByAdmin(String roomName)
```

参数：

参数	类型	含义
roomName	String	房间名称

updateRoomSpeechModeByAdmin

设置房间管理模式（只有管理员或群主能够调用）。

```
Future<TUIActionCallback> updateRoomSpeechModeByAdmin(TUISpeechMode mode)
```

参数	类型	含义
mode	TUISpeechMode	房间模式

setLocalVideoView

设置本地用户视频渲染的视图控件。

```
void setLocalVideoView(int viewId)
```

参数：

参数	类型	含义
viewId	int	待渲染 view 指针的 int64 类型值，通过此 viewId 可转换为对应原生平台的视图，视频画面将渲染于此视图上

openLocalCamera

打开本地摄像头，开始视频流采集。

```
Future<TUIActionCallback> openLocalCamera(bool isFront,
                          TUIVideoQuality quality)
```

参数：

参数	类型	含义
isFront	bool	是否使用前置摄像头
quality	TUIVideoQuality	视频质量

closeLocalCamera

关闭本地摄像头。

```
void closeLocalCamera()
```

updateVideoQuality

设置本地视频的参数。

```
void updateVideoQuality(TUIVideoQuality quality)
```

参数：

参数	类型	含义
quality	TUIVideoQuality	视频质量

updateVideoQualityEx

设置本地视频编码器参数。

```
void updateVideoQualityEx(
    TUIVideoStreamType streamType, TUIRoomVideoEncoderParams params);
```

参数	类型	含义
streamType	TUIVideoStreamType	视频流类型
params	TUIRoomVideoEncoderParams	视频编码器参数

setVideoResolutionMode

设置视频编码器分辨率模式（横屏分辨率 or 竖屏分辨率）。

```
void setVideoResolutionMode(
    TUIVideoStreamType streamType, TUIResolutionMode resolutionMode);
```

参数	类型	含义
streamType	TUIVideoStreamType	视频流类型
resolutionMode	TUIResolutionMode	分辨率模式

enableGravitySensor

开启重力感应模式。

```
void enableGravitySensor(bool enable);
```

参数	类型	含义
enable	bool	是否开启

startPushLocalVideo

开始向远端推本地视频流。

```
void startPushLocalVideo()
```

stopPushLocalVideo

停止向远端推本地视频流。

```
void stopPushLocalVideo()
```

startScreenSharing

开始屏幕分享

```
Future<void> startScreenSharing({String appGroup = ''})
```

stopScreenSharing

结束屏幕分享

```
Future<void> stopScreenSharing()
```

openLocalMicrophone

打开本地麦克风。

```
Future<TUIActionCallback> openLocalMicrophone(TUIAudioQuality quality)
```

参数	类型	含义
quality	TUIAudioQuality	音频质量

closeLocalMicrophone

关闭本地麦克风。

```
void closeLocalMicrophone()
```

updateAudioQuality

更新本地音频编码质量设置。

```
void updateAudioQuality(TUIAudioQuality quality)
```

参数：

参数	类型	含义
quality	TUIAudioQuality	音频质量

muteLocalAudio

停止向远端推本地音频流。

```
Future<TUIActionCallback> muteLocalAudio()
```

unMuteLocalAudio

开始向远端推本地音频流。

```
Future<TUIActionCallback> unMuteLocalAudio()
```

setRemoteVideoView

设置远端用户视频渲染的视图控件。

```
void setRemoteVideoView(String userId,
                        TUIVideoStreamType streamType,
                        int viewId)
```

参数：

参数	类型	含义
userId	String	用户 ID
streamType	TUIVideoStreamType	用户流类型
viewId	int	待渲染 view 指针的 int64 类型值，通过此 viewId 可转换为对应原生平台的视图，视频画面将渲染于此视图上

startPlayRemoteVideo

开始播放远端用户视频流。

```
void startPlayRemoteVideo(String userId,
                          TUIVideoStreamType streamType,
                          TUIPlayCallback? callback)
```

参数：

参数	类型	含义
userId	String	用户 ID
streamType	TUIVideoStreamType	用户流类型
callback	TUIPlayCallback?	播放结果回调

stopPlayRemoteVideo

停止播放远端用户视频流。

```
void stopPlayRemoteVideo(String userId,
                         TUIVideoStreamType streamType)
```

参数：

参数	类型	含义
userId	String	用户 ID
streamType	TUIVideoStreamType	用户流类型

muteRemoteAudioStream

将远端用户禁音。

```
void muteRemoteAudioStream(String userId, boolean isMute);
```

参数：

参数	类型	含义
userId	String	用户 ID
isMute	bool	是否禁音

getUserList

获取当前房间用户列表，注意该接口一次拉取的用户列表量最大为 100 个。

```
Future<TUIValueCallBack<TUIUserListResult>> getUserList(int nextSequence)
```

参数：

参数	类型	含义
nextSequence	int	分页拉取标志，第一次拉取填0。回调成功如果 nextSeq 不为零，需要分页，传入 nextSeq 再次拉取，直至 callback 中 nextSeq 为0

getUserInfo

获取用户的详细信息。

```
Future<TUIValueCallBack<TUIUserInfo>> getUserInfo(String userId)
```

参数：

参数	类型	含义
userId	String	根据 userId 获取该用户的详细信息

changeUserRole

改变用户的角色，只有管理员或群主能够调用。

```
Future<TUIActionCallback> changeUserRole(String userId,
                                         TUIRole role)
```

参数：

参数	类型	含义
userId	String	用户 ID
role	TUIRole	用户角色

kickRemoteUserOutOfRoom

将用户移出房间，只有管理员或群主能够调用。

```
Future<TUIActionCallback> kickRemoteUserOutOfRoom(String userId)
```

参数：

参数	类型	含义
userId	String	用户 ID

addCategoryTagForUsers

为用户增加标记，只有房主能够调用

```
Future<TUIActionCallback> addCategoryTagForUsers(int tag, List<String> userList);
```

参数：

参数	类型	含义
tag	int	标记类型。数字类型，大于等于 1000，您可以自定义。
userList	List<String>	用户列表

removeCategoryTagForUsers

为用户移除标记，只有房主能够调用

```
Future<TUIActionCallback> removeCategoryTagForUsers(int tag, List<String> userList);
```

参数：

参数	类型	含义
tag	int	类型。数字类型，大于等于 1000，您可以自定义。
userList	List<String>	用户列表

getUserListByTag

根据标记获取房间内用户信息
为用户移除标记，只有房主能够调用

```
Future<TUIValueCallBack<TUIUserListResult>> getUserListByTag(int tag, int nextSequence);
```

参数：

参数	类型	含义
tag	int	类型。数字类型，大于等于 1000，您可以自定义。
nextSequence	int	分页拉取标志，第一次拉取填0，回调成功 如果 callback 返回的数据中 nextSequence 不为零，需要分页，传入再次拉取，直至为0

disableDeviceForAllUserByAdmin

全体用户媒体设备管理，只有管理员或群主能够调用。

版权所有：腾讯云计算（北京）有限责任公司

```
Future<TUIActionCallback> disableDeviceForAllUserByAdmin(TUIMediaDevice device,
                                                         bool isDisable)
```

参数	类型	含义
device	TUIMediaDevice	设备
isDisable	bool	是否禁用

openRemoteDeviceByAdmin

请求远端用户打开媒体设备，只有管理员或群主能够调用。

```
TUIRequest openRemoteDeviceByAdmin(String userId,
                                   TUIMediaDevice device,
                                   int timeout,
                                   TUIRequestCallback? requestCallback)
```

参数	类型	含义
userId	String	用户 ID
device	TUIMediaDevice	设备
timeout	int	超时时间，单位秒。如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
requestCallback	TUIRequestCallback?	操作结果回调

closeRemoteDeviceByAdmin

关闭远端用户媒体设备，只有管理员或群主能够调用。

```
Future<TUIActionCallback> closeRemoteDeviceByAdmin(String userId,
                                                   TUIMediaDevice device)
```

参数	类型	含义
userId	String	用户ID
device	TUIMediaDevice	设备

applyToAdminToOpenLocalDevice

全体用户媒体设备管理 lock。

```
TUIRequest applyToAdminToOpenLocalDevice(TUIMediaDevice device,
                                         int timeout,
                                         TUIRequestCallback? requestCallback)
```

参数	类型	含义

device	TUIMediaDevice	设备
timeout	int	超时时间，单位秒。如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
callback	TUIRequestCallback?	操作结果回调

setMaxSeatCount

设置最大麦位数，仅支持进房前和创建房间时设置。

```
Future<TUIActionCallback> setMaxSeatCount(int maxSeatCount)
```

参数	类型	含义
maxSeatCount	int	最大麦位数

lockSeatByAdmin

锁定麦位（包括位置锁定，音频状态锁定，视频状态锁定）。

```
Future<TUIActionCallback> lockSeatByAdmin(int seatIndex,
                          TUISeatLockParams lockParams)
```

参数	类型	含义
seatIndex	int	麦位编号
lockParams	TUISeatLockParams	锁麦参数

getSeatList

获取麦位列表。

```
Future<TUIValueCallBack<List<TUISeatInfo>>> getSeatList()
```

takeSeat

本地上麦。

> **说明：**
> - 会议场景：applyToSpeak 模式需要向主持人或管理员发起申请才允许上麦，其他模式不支持上麦。
> - 直播场景：freeToSpeak 模式可以自由上麦，上麦后开麦发言；applySpeakAfterTakingSeat 模式需要向主持人或管理员发起申请才允许上麦；其他模式不支持上麦。

```
TUIRequest takeSeat(int seatIndex,
            int timeout,
            TUIRequestCallback? requestCallback)
```

参数：

参数	类型	含义
seatIndex	int	麦位编号
timeout	int	超时时间，单位秒。如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
requestCallback	TUIRequestCallback?	调用接口的回调，用于通知请求的回调状态

Return: 请求体

leaveSeat

本地下麦。

```
Future<TUIActionCallback> leaveSeat()
```

takeUserOnSeatByAdmin

主持人/管理员 邀请用户上麦。

```
TUIRequest takeUserOnSeatByAdmin(int seatIndex,
                                 String userId,
                                 int timeout,
                                 TUIRequestCallback? requestCallback)
```

参数	类型	含义
seatIndex	int	麦位编号
userId	String	用户 ID
timeout	int	超时时间，单位秒。如果设置为 0，SDK 不会做超时检测，也不会触发超时回调
requestCallback	TUIRequestCallback?	调用接口的回调，用于通知请求的回调状态

Return: 请求体

kickUserOffSeatByAdmin

主持人/管理员 将用户下麦。

```
Future<TUIActionCallback> kickUserOffSeatByAdmin(int seatIndex,
                                                 String userId)
```

参数	类型	含义
seatIndex	int	麦位编号
userId	String	用户 ID

sendTextMessage

发送文本消息。

```
Future<TUIActionCallback> sendTextMessage(String message)
```

参数：

参数	类型	含义
message	String	文本消息内容

sendCustomMessage

发送自定义消息。

```
Future<TUIActionCallback> sendCustomMessage(String message)
```

参数：

参数	类型	含义
message	String	自定义消息内容

disableSendingMessageByAdmin

禁用远端用户的发送文本消息能力（只有管理员或群主能够调用）。

```
Future<TUIActionCallback> disableSendingMessageByAdmin(String userId,
                                                       bool isDisable)
```

参数	类型	含义
userId	String	用户 ID
isDisable	bool	是否禁用

disableSendingMessageForAllUser

禁用所有用户的发送文本消息能力（只有管理员或群主能够调用）。

```
Future<TUIActionCallback> disableSendingMessageForAllUser(bool isDisable)
```

参数	类型	含义
isDisable	bool	是否禁用

cancelRequest

取消已经发出的请求。

```
Future<TUIActionCallback> cancelRequest(String requestId)
```

参数：

参数	类型	含义
requestId	String	请求 ID

responseRemoteRequest

回复远端用户的请求。

```
Future<TUIActionCallback> responseRemoteRequest(String requestId,
                                                bool agree)
```

参数：

参数	类型	含义
requestId	String	请求 ID
agree	bool	是否同意

callExperimentalAPI

调用实验性接口。

```
void callExperimentalAPI(String jsonStr);
```

参数	类型	含义
jsonStr	String	接口信息

switchCamera

切换前置或后置摄像头（仅适用于移动端）。

```
Future<int?> switchCamera(bool isFrontCamera);
```

参数	类型	含义
isFrontCamera	bool	是否前置

setBeautyLevel

设置美颜级别。

```
void setBeautyLevel(int beautyStyle, int beautyLevel);
```

参数	类型	含义
beautyStyle	int	美颜风格
beautyLevel	int	美颜级别

setWhitenessLevel

设置美白级别。

```
void setWhitenessLevel(int whitenessLevel);
```

参数	类型	含义
whitenessLevel	int	美白级别

TUIRoomObserver

最近更新时间：2023-07-06 11:30:32

TUIRoomEngine 事件回调

onError

错误事件。

```
OnError onError = (TUIError errorCode, String message) {}
```

参数	类型	描述
errorCode	TUIError	错误码
message	String	错误信息

onKickedOffLine

用户被移下线事件。

```
OnKickedOffLine onKickedOffLine = (String message) {}
```

参数	类型	描述
message	String	被移下线的描述

onUserSigExpired

`userSig` **过期事件。**

```
OnUserSigExpired onUserSigExpired = () {}
```

onRoomNameChanged

房间名称更改事件。

```
OnRoomNameChanged onRoomNameChanged = (String roomId, String roomName) {}
```

参数	类型	描述
roomId	String	房间 ID
roomName	String	房间名称

onAllUserMicrophoneDisableChanged

房间内所有用户麦克风被禁用事件。

版权所有：腾讯云计算（北京）有限责任公司

```
OnAllUserMicrophoneDisableChanged onAllUserMicrophoneDisableChanged = (String roomId, bool isDisable)
{}
```

参数	类型	描述
roomId	String	房间 ID
isDisable	bool	是否被禁用

onAllUserCameraDisableChanged

房间内所有用户摄像头被禁用事件。

```
OnAllUserCameraDisableChanged onAllUserCameraDisableChanged = (String roomId, bool isDisable) {}
```

参数	类型	描述
roomId	String	房间 ID
isDisable	bool	是否被禁用

onSendMessageForAllUserDisableChanged

房间内所有用户发送文本消息被禁用事件。

```
OnSendMessageForAllUserDisableChanged onSendMessageForAllUserDisableChanged = (String roomId, bool
isDisable) {}
```

参数	类型	描述
roomId	String	房间 ID
isDisable	bool	是否被禁用

onRoomDismissed

房间被解散事件。

```
OnRoomDismissed onRoomDismissed = (String roomId) {}
```

参数	类型	描述
roomId	String	房间 ID

onKickedOutOfRoom

移出房间事件。

```
OnKickedOutOfRoom onKickedOutOfRoom = (String roomId, String message) {}
```

参数	类型	描述

roomId	String	房间 ID
message	String	被移出的描述

onRoomSpeechModeChanged

房间麦控模式发生变化。

```
OnRoomSpeechModeChanged onRoomSpeechModeChanged = (String roomId, TUISpeechMode speechMode)
{}
```

参数	类型	描述
roomId	String	房间 ID
speechMode	TUISpeechMode	房间麦控模式

onRemoteUserEnterRoom

远端用户进入房间事件。

```
OnRemoteUserEnterRoom onRemoteUserEnterRoom = (String roomId, TUIUserInfo userInfo) {}
```

参数	类型	描述
roomId	String	房间 ID
userInfo	TUIUserInfo	用户信息

onRemoteUserLeaveRoom

远端用户离开房间事件。

```
OnRemoteUserLeaveRoom onRemoteUserLeaveRoom = (String roomId, TUIUserInfo userInfo) {}
```

参数	类型	描述
roomId	String	房间 ID
userInfo	TUIUserInfo	用户信息

onUserRoleChanged

用户角色改变事件。

```
OnUserRoleChanged onUserRoleChanged = (String userId, TUIRole role) {}
```

参数	类型	描述
userId	String	用户 ID
role	TUIRole	用户角色

版权所有：腾讯云计算（北京）有限责任公司

onUserVideoStateChanged

用户视频状态改变事件。

```
OnUserVideoStateChanged onUserVideoStateChanged = (String userId, TUIVideoStreamType streamType,
bool hasVideo, TUIChangeReason reason) {}
```

参数	类型	描述
userId	String	用户 ID
streamType	TUIVideoStreamType	视频流类型
hasVideo	bool	是否有视频流
reason	TUIChangeReason	视频流发生变化原因

onUserAudioStateChanged

用户音频状态改变事件。

```
OnUserAudioStateChanged onUserAudioStateChanged = (String userId, bool hasAudio, TUIChangeReason
reason) {}
```

参数	类型	描述
userId	String	用户 ID
hasAudio	bool	是否有音频流
reason	TUIChangeReason	视频流发生变化原因

onUserVoiceVolumeChanged

用户音量改变事件。

```
OnUserVoiceVolumeChanged onUserVoiceVolumeChanged = (Map<String, int> volumeMap) {}
```

参数	类型	描述
volumeMap	Map	用户音量 Map • key：userId • value：用于承载所有正在说话的用户的音量大小。取值范围 0 - 100

onSendMessageForUserDisableChanged

用户文本消息发送能力发生变化事件。

```
OnSendMessageForUserDisableChanged onSendMessageForUserDisableChanged = (String roomId, String
userId, bool isDisable) {}
```

参数	类型	描述
roomId	String	房间 ID
userId	String	用户 ID
isDisable	bool	是否被禁止发送文本消息

onUserNetworkQualityChanged

用户网络质量改变事件。

```
OnUserNetworkQualityChanged onUserNetworkQualityChanged = (Map<String, TUINetwork> networkMap)
{}
```

参数	类型	描述
networkMap	Map	用户网络状态 Map • key：userId • value：网络状况

onUserScreenCaptureStopped

屏幕分享结束事件。

```
OnUserScreenCaptureStopped onUserScreenCaptureStopped = (int reason) {}
```

参数	类型	描述
reason	int	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）

onRoomMaxSeatCountChanged

房间内最大麦位数发生变化事件（仅在会议类型房间生效）。

```
OnRoomMaxSeatCountChanged onRoomMaxSeatCountChanged = (String roomId, int maxSeatCount) {}
```

参数	类型	描述
roomId	String	房间 ID
maxSeatCount	int	房间内最大麦位数量

onSeatListChanged

麦位列表改变事件。

```
OnSeatListChanged onSeatListChanged = (List<TUISeatInfo> seatList, List<TUISeatInfo> seatedList,
List<TUISeatInfo> leftList) {}
```

参数	类型	描述
seatList	List<TUISeatInfo>	目前麦上最新的用户列表，包含新上麦的用户
seatedList	List<TUISeatInfo>	新上麦的用户列表
leftList	List<TUISeatInfo>	新下麦的用户列表

onKickedOffSeat

收到用户被移下麦事件。

```
OnKickedOffSeat onKickedOffSeat = (String userId) {}
```

参数	类型	描述
userId	String	操作移人的（主持人/管理员）用户 ID

onRequestReceived

请求接收事件。

```
OnRequestReceived onRequestReceived = (TUIRequest request) {}
```

参数	类型	描述
request	TUIRequest	请求内容

onRequestCancelled

请求取消事件。

```
OnRequestCancelled onRequestCancelled = (String requestId, String userId) {}
```

参数	类型	描述
requestId	String	请求 ID
userId	String	取消信令的用户 ID

onReceiveTextMessage

接收文本消息事件。

```
OnReceiveTextMessage onReceiveTextMessage = (String roomId, TUIMessage message) {}
```

参数	类型	描述

版权所有：腾讯云计算（北京）有限责任公司

roomId	String	房间 ID
message	TUIMessage	消息内容

onReceiveCustomMessage

接收自定义消息事件。

```
OnReceiveCustomMessage onReceiveCustomMessage = (String roomId, TUIMessage message) {}
```

参数	类型	描述
roomId	String	房间 ID
message	TUIMessage	消息内容

类型定义

最近更新时间：2023-11-17 11:00:42

枚举定义

TUIRoomDefine

类型	说明
TUIRoomType	房间类型
TUISpeechMode	麦控模式
TUIMediaDevice	房间内媒体设备类型
TUIRole	房间内角色类型
TUIVideoQuality	视频质量
TUIAudioQuality	音频质量
TUIVideoStreamType	视频流类型
TUIChangeReason	更改原因（用户音视频状态变更操作原因：自己主动修改或者被房主、管理员修改）
TUICaptureSourceType	屏幕分享捕获源类型
TUIRequestAction	请求类型
TUIResolutionMode	分辨率模式（横屏 or 竖屏）

TUICommonDefine

类型	说明
TUIError	错误码
TUINetworkQuality	网络质量

常用结构

TUIRoomDefine

类型	说明
TUIRoomInfo	房间信息
TUILoginUserInfo	用户登录信息
TUIUserInfo	房间内用户信息
TUISeatInfo	房间内座位信息
TUISeatLockParams	锁定麦位操作参数
TUIUserVoiceVolume	房间内用户音量

版权所有：腾讯云计算（北京）有限责任公司

TUIRequest	信令请求
TUIActionCallback	用户操作回调
TUIPlayCallback	视频播放回调
TUIRequestCallback	用户请求回调
TUIUserListResult	用户列表信息
TUIUserVoiceVolume	用户音量信息
TUIValueCallBack<T>	含返回值（T）回调
TUIRoomVideoEncoderParams	视频编码器参数

TUICommonDefine

类型	说明
TUINetwork	网络质量信息
TUIMessage	消息
TUIImageBuffer	图片信息

TUIRoomType

房间类型

枚举	取值	描述
conference	1	会议类型房间，适用于会议和教育场景。在该房间中，用户可以选择自由发言模式或者申请发言、上麦发言等不同模式
livingRoom	2	直播类型房间，适用于直播场景。在该房间中，用户可以自由发言，并且房间采用麦位控制模式，每个麦位都有编号

TUISpeechMode

麦控模式

枚举	取值	描述
freeToSpeak	1	自由发言模式
applyToSpeak	2	申请发言模式（仅在会议类型房间下生效）
applySpeakAfterTakingSeat	3	上麦发言模式

说明：
房间类型、麦控模式与上麦（takeSeat）的关系

房间类型	麦控模式	

	freeToSpeak	applyToSpeak	applySpeakAfterTakingSeat
conference	不支持上麦	不支持上麦	需要向房主/管理员申请（takeSeat），待同意后方可开麦克风/摄像头发言
livingRoom	可以自由上麦	不支持上麦	需要向房主/管理员申请（takeSeat），待同意后方可开麦克风/摄像头发言

TUIMediaDevice

房间内媒体设备类型

枚举	取值	描述
microphone	1	麦克风
camera	2	摄像头
screen	3	屏幕共享

TUIRole

房间内角色类型

枚举	取值	描述
roomOwner	0	房主。一般指房间的创建者，房间内最高权限拥有者
administrator	1	房间管理员
generalUser	2	房间内普通成员

TUIVideoQuality

视频质量

枚举	取值	描述
videoQuality_360P	1	低清 360P
videoQuality_540P	2	标清 540P
videoQuality_720P	3	高清 720P
videoQuality_1080P	4	超清 1080P

TUIAudioQuality

音频质量

枚举	取值	描述
audioProfileSpeech	0	人声模式
audioProfileDefault	1	默认模式

版权所有：腾讯云计算（北京）有限责任公司

audioProfileMusic	2	音乐模式

TUIVideoStreamType

视频流类型

枚举	取值	描述
cameraStream	0	高清摄像头视频流
screenStream	1	屏幕分享视频流
cameraStreamLow	2	低清摄像头视频流

TUIChangeReason

更改原因（用户音视频状态变更操作原因: 自己主动修改 或者 被房主、管理员修改）

枚举	取值	描述
changedBySelf	0	自己操作
changedByAdmin	1	房主或管理员操作

TUICaptureSourceType

屏幕分享捕获源类型

枚举	取值	描述
unknown	-1	未定义
window	0	窗口
screen	1	屏幕

TUIRequestAction

请求类型

枚举	取值	描述
invalidAction	0	无效请求
requestToOpenRemoteCamera	1	请求远端用户打开摄像头
requestToOpenRemoteMicrophone	2	请求远端用户打开麦克风
requestToConnectOtherRoom	3	请求连接到其他房间
requestToTakeSeat	4	请求上麦
requestRemoteUserOnSeat	5	请求远端用户上麦
applyToAdminToOpenLocalC	6	向管理员请求打开本地摄像头

版权所有：腾讯云计算（北京）有限责任公司

amera		
applyToAdminToOpenLocalMicrophone	7	向管理员请求打开本地麦克风

TUIResolutionMode

枚举	取值	描述
landscape	0	横屏
portrait	1	竖屏

TUIError

错误码

枚举	取值	描述
success	0	操作成功
errFailed	-1	暂未归类的通用错误
errFreqLimit	-2	请求被限频，请稍后重试
errRepeatOperation	-3	重复操作，请检查您的接口调用是否重复
errSDKAppIDNotFound	-1000	未找到 SDKAppID，请在 腾讯云视立方 SDK 控制台 确认应用信息
errInvalidParameter	-1001	调用 API 时，传入的参数不合法
errSdkNotInitialized	-1002	SDK 未初始化
errPermissionDenied	-1003	无操作权限
errRequirePayment	-1004	该功能需要开通额外的套餐
errCameraStartFailed	-1100	打开摄像头失败
errCameraNotAuthorized	-1101	摄像头未授权
errCameraOccupy	-1102	摄像头被占用
errCameraDeviceEmpty	-1103	当前无摄像头设备
errMicrophoneStartFailed	-1104	麦克风打开失败
errMicrophoneNotAuthorized	-1105	麦克风未授权
errMicrophoneOccupy	-1106	麦克风被占用
errMicrophoneDeviceEmpty	-1107	当前无麦克风设备
errGetScreenSharingTargetFailed	-1108	获取屏幕分享对象失败
errStartScreenSharingFailed	-1109	开启屏幕分享失败
errRoomIdNotExist	-2100	进房时房间不存在，或许已被解散
errOperationInvalidBeforeEnterRoo	-2101	需要进房后才可使用此功能

m		
errExitNotSupportedForRoomOwner	-2102	房主不支持退房操作，Conference（会议）房间类型：可以先转让房主，再退房。LivingRoom（直播）房间类型：房主只能解散房间
errOperationNotSupportedInCurrentRoomType	-2103	当前房间类型下不支持该操作
errOperationNotSupportedInCurrentSpeechMode	-2104	当前发言模式下不支持该操作
errRoomIdInvalid	-2105	创建房间 ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
errRoomIdOccupied	-2106	房间 ID 已被使用，请选择别的房间 ID
errRoomNameInvalid	-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
errAlreadyInOtherRoom	-2108	当前用户已在别的房间内，需要先退房才能加入新的房间： 单个 roomEngine 实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的 roomEngine 实例
errUserNotExist	-2200	用户不存在
errUserNotEntered	-2201	用户不在当前房间内
errUserNeedOwnerPermission	-2300	需要房主权限才能操作
errUserNeedAdminPermission	-2301	需要房主或者管理员权限才能操作
errRequestNoPermission	-2310	信令请求无权限，例如取消非自己发起的邀请
errRequestIdInvalid	-2311	信令请求 ID 无效或已经被处理过
errMaxSeatCountLimit	-2340	最大麦位超出套餐包数量限制
errAlreadyInSeat	-2341	当前用户已经在麦位上
errSeatOccupied	-2342	当前麦位已经有人了
errSeatLocked	-2343	当前麦位被锁
errSeatIndexNotExist	-2344	麦位编号不存在
errUserNotInSeat	-2345	当前用户没有在麦上
errAllSeatOccupied	-2346	上麦人数已满
errOpenMicrophoneNeedSeatUnlock	-2360	当前麦位音频被锁
errOpenMicrophoneNeedPermissionFromAdmin	-2361	需要向房主或管理员申请后打开麦克风
errOpenCameraNeedSeatUnlock	-2370	当前麦位视频被锁，需要由房主解锁麦位后，才能打开摄像头
errOpenCameraNeedPermissionFromAdmin	-2371	需要向房主或管理员申请后打开摄像头

版权所有：腾讯云计算（北京）有限责任公司

errSendMessageDisabledForAll	-2380	当前房间已开启全员禁言
errSendMessageDisabledForCurrent	-2381	当前房间内，您已被禁言

TUINetworkQuality

网络质量

枚举	取值	描述
qualityUnknown	0	未定义
qualityExcellent	1	当前网络非常好
qualityGood	2	当前网络比较好
qualityPoor	3	当前网络一般
qualityBad	4	当前网络较差
qualityVeryBad	5	当前网络很差
qualityDown	6	当前网络不满足 TRTC 的最低要求

TUIRoomInfo

房间信息

字段	类型	描述
roomId	String	房间 ID
roomType	TUIRoomType	房间类型
ownerId	String	主持人 ID。默认为房间创建者（只读）
name	String	房间名称。默认为房间 ID
speechMode	TUISpeechMode	房间发言模式
createTime	int	房间创建时间（只读）
memberCount	int	房间内成员数量（只读）
maxSeatCount	int	最大麦位数量（仅支持进房前和创建房间时设置）
isCameraDisableForAllUser	bool	是否禁止打开摄像头（创建房间可选参数）。默认值：false
isMicrophoneDisableForAllUser	bool	是否禁止打开麦克风（创建房间可选参数）。默认值：false
isMessageDisableForAllUser	bool	是否禁止发送消息（创建房间可选参数）。默认值：false
enableCDNStreamin	bool	是否开启 CDN 直播（创建房间可选参数，直播房间使用）。默认值：false

版权所有：腾讯云计算（北京）有限责任公司

g		
cdnStreamDomain	String	直播推流域名（创建房间可选参数，直播房间使用）。默认值：空

TUILoginUserInfo

用户登录信息

字段	类型	含义
userId	String	用户 ID
userName	String	用户名称
avatarUrl	String	用户头像 URL
customInfo	Map<String, String>	自定义信息

TUIUserInfo

房间内用户信息

字段	类型	描述
userId	String	用户 ID
userName	String	用户名称
avatarUrl	String	用户头像 URL
userRole	TUIRole	用户角色类型
hasAudioStream	bool	是否有音频流。默认值：false
hasVideoStream	bool	是否有视频流。默认值：false
hasScreenStream	bool	是否有屏幕分享流。默认值：false

TUISeatInfo

房间内座位信息

字段	类型	描述
index	int	麦位序号
userId	String	用户 ID
isLocked	bool	麦位是否被锁定。默认 false
isVideoLocked	bool	麦位是否被禁止打开摄像头。默认 false
isAudioLocked	bool	麦位是否被禁止打开麦克风。默认 false

TUISeatLockParams

锁定麦位操作参数

字段	类型	含义

lockSeat	bool	锁定麦位。默认 false
lockVideo	bool	锁定麦位摄像头。默认 false
lockAudio	bool	锁定麦位麦克风。默认 false

TUIUserVoiceVolume

房间内用户音量

字段	类型	描述
userId	String	用户 ID
volume	int	音量。用于承载所有正在说话的用户的音量大小，取值范围 0 – 100

TUIRequest

信令请求

字段	类型	描述
requestId	String	请求 ID
requestAction	TUIRequestAction	请求类型
userId	String	用户 ID
content	String	信令内容
timestamp	int	时间戳

TUIActionCallback

字段	类型	描述
code	TUIError	错误码
message	String?	错误信息

TUIPlayCallback

字段	类型	描述
onPlaying	(String userId) {}	正在播放回调
onLoading	(String userId) {}	正在加载回调
onPlayError	(String userId, TUIError code, String message) {}	播放错误回调

TUIRequestCallback

字段	类型	描述
onAccepted	(String requestId, String userId) {}	请求被接受回调

onRejected	(String requestId, String userId, String message) {}	请求被拒绝回调
onCancelled	(String requestId, String userId) {}	请求取消回调
onTimeout	(String requestId, String userId) {}	请求超时回调
onError	(String requestId, String userId, TUIError error, String message) {}	请求错误回调

TUIUserListResult

字段	类型	描述
nextSequence	int	分页拉取标志，若返回的 nextSequence 不为零，需要使用返回的 nextSequence 再次拉取，直至返回为0
userInfoList	List<TUIUserInfo>	本次调用返回的用户列表

TUIUserVoiceVolume

字段	类型	描述
userId	String	用户 ID
volume	int	用户音量大小

TUIValueCallBack<T>

字段	类型	描述
code	TUIError	错误码
message	String?	错误信息
data	T?	返回数据，示例：若 T 为 TUIUserInfo，则 TUIValueCallBack<TUIUserInfo> 的 data 字段类型为 TUIUserInfo

TUIRoomVideoEncoderParams

字段	类型	描述
videoResolution	TUIVideoQuality	视频质量
resolutionMode	TUIResolutionMode	分辨率模式
fps	int	视频采集帧率
bitrate	int	目标视频码率

TUINetwork

网络质量信息

字段	类型	描述
userId	String	用户 ID
quality	TUINetworkQuality	网络质量
upLoss	int	上行丢包率
downLoss	int	下行丢包率
delay	int	网络延迟

TUIMessage

消息

字段	类型	描述
messageId	String	消息 ID
message	String	消息文本
timestamp	int	消息时间
userId	String	消息发送者
userName	String	消息发送者昵称
avatarUrl	String	消息发送者头像

TUIImageBuffer

图片信息

字段	类型	描述
buffer	String	图片数据缓存地址
length	int	长度
width	int	宽度
height	int	高度

uni-app（小程序）API 概览

最近更新时间：2024-03-04 14:32:51

TUIRoomKit (含 UI 组件)

TUIRoomKit 接口列表

API	描述
init	初始化 TUIRoomKit 组件
createRoom	主持人创建房间
enterRoom	普通成员加入房间

TUIRoomKit 事件类型

事件	描述
onCreateRoom	创建房间回调
onEnterRoom	进入房间回调
onDestroyRoom	销毁房间回调
onExitRoom	普通成员退出房间回调
onKickedOutOfRoom	普通成员被主持人移出房间通知
onKickedOffLine	用户账号在其他地方登录，被移下线
onUserSigExpired	用户 userSig 过期，请重新获取 userSig

TUIRoomEngine (无 UI 接口)

TUIRoomEngine 接口列表

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.init 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.init 方法执行成功后执行。**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

版权所有：腾讯云计算（北京）有限责任公司

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用），该接口自 v2.0.0 版本废弃
updateRoomSeatModeByAdmin	更新房间上麦模式（仅群主或者管理员可以调用）
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集
closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateVideoQuality	设置本地视频的参数
updateAudioQuality	设置本地音频的参数
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
startPushLocalAudio	开始向远端推本地音频流
stopPushLocalAudio	停止向远端推本地音频流
muteLocalAudio	停止向远端推本地音频流
unmuteLocalAudio	开始向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域
startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流

版权所有：腾讯云计算（北京）有限责任公司

muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备
cancelRequest	取消已经发出的请求
responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户移出房间
disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 麦位管理 API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人移下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送 API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

roomEngine 事件监听 API

API	描述

on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

TUIRoomEngine 事件类型

TUIRoomEvent 是 TUIRoomEngine 对应的回调事件类，您可以通过此回调，来监听自己感兴趣的回调事件。

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	移出房间事件
TUIRoomEvents.onKickedOffLine	当前用户被移下线事件
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomNameChanged	房间名称修改事件
~~TUIRoomEvents.onRoomSpeechModeChanged~~	房间发言模式修改事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onAllUserCameraDisableChanged	所有成员摄像头使用权限改变事件
TUIRoomEvents.onAllUserMicrophoneDisableChanged	所有成员麦克风使用权限改变事件
TUIRoomEvents.onSendMessageForAllUserDisableChanged	所有成员发送消息状态改变事件
TUIRoomEvents.onRoomMaxSeatCountChanged	房间最大麦位数修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChanged	用户音频状态改变事件

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEvents.onSendMessageForUserDisableChanged	用户发送消息状态事件
TUIRoomEvents.onUserVoiceVolumeChanged	用户音量改变事件
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件
TUIRoomEvents.onSeatListChanged	麦位列表改变事件
TUIRoomEvents.onKickedOffSeat	用户被移下麦事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onReceiveCustomMessage	接收自定义消息事件

TUIRoomKit

最近更新时间：2023-09-15 15:11:35

API 简介

TUIRoomKit API 是多人视频会议组件的**含 UI 接口**，使用 TUIRoomKit API，您可以通过简单接口快速实现一个类微信的音视频通话场景。如果您想要体验并调试视频会议效果，请阅读 Demo 快速跑通 。如果您想把我们的功能直接嵌入到您的项目中，请阅读 快速接入(TUIRoomKit) 。

API 概览

在页面中引用 TUIRoomKit 组件。例如：在 room.vue 组件中引入 TUIRoomKit 组件。

- TUIRoomKit 组件将用户分为主持人角色及普通成员角色。组件对外提供了 init 、createRoom 、enterRoom 方法。
- 主持人及普通成员可通过 init 方法向 TUIRoomKit 组件初始化应用及用户数据，主持人可通过 createRoom 方法创建并加入房间，普通成员可通过 enterRoom 方法加入主持人已经创建好的房间。

```
<template>
  <room ref="TUIRoomRef"></room>
</template>

<script setup lang="ts">
import { ref, onMounted } from 'vue';
// 引入 TUIRoom 组件，注意确认引入路径是否正确
import Room from './TUIRoom/index.vue';
// 获取 TUIRoom 组件元素，用于调用 TUIRoom 组件的方法
const TUIRoomRef = ref();

onMounted(async () => {
  // 初始化 TUIRoom 组件
  // 主持人在创建房间前需要先初始化 TUIRoom 组件
  // 普通成员在进入房间前需要先初始化 TUIRoom 组件
  await TUIRoomRef.value.init({
    // 获取 sdkAppId 请您参考 步骤一
    sdkAppId: 0,
    // 用户在您业务中的唯一标示 Id
    userId: '',
    // 本地开发调试可在 https://console.cloud.tencent.com/trtc/usersigtool 页面快速生成 userSig, 注意 userSig 与
userId 为一一对应关系
    userSig: '',
    // 用户在您业务中使用的昵称
    userName: '',
    // 用户在您业务中使用的头像链接
    avatarUrl: '',
    // 用户在您业务中需要的皮肤主题颜色及是否支持切换皮肤主题
    theme: {
      defaultTheme:'black',
      isSupportSwitchTheme: true
     }
  })
  // 默认执行创建房间，实际接入可按需求择机执行 handleCreateRoom 方法
  await handleCreateRoom();
})
```

```
// 主持人创建房间，该方法只在创建房间时调用
async function handleCreateRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomMode 包含'FreeToSpeak'(自由发言模式) 和'SpeakAfterTakingSeat'(上台发言模式) 两种模式，默认为'FreeToSpeak'，注意目前仅支持自由发言模式
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
  const roomMode = 'FreeToSpeak';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.createRoom({ roomId, roomName: roomId, roomMode, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.createRoom error: ' + error.message);
  }
}

// 普通成员进入房间，该方法在普通成员进入已创建好的房间时调用
async function handleEnterRoom() {
  // roomId 为用户进入的房间号, 要求 roomId 类型为 string
  // roomParam 指定了用户进入房间的默认行为（是否默认开启麦克风，是否默认开启摄像头，默认媒体设备Id)
  const roomId = '123456';
  const roomParam = {
    isOpenCamera: true,
    isOpenMicrophone: true,
  }
  try {
    await TUIRoomRef.value.enterRoom({ roomId, roomParam });
  } catch (error: any) {
    alert('TUIRoomKit.enterRoom error: ' + error.message);
  }
}
</script>
<style>
page {
  width: 100%;
  height: 100%;
}
</style>
```

API 详情

TUIRoomkit 接口

init

初始化 TUIRoomKit 数据，任何使用 TUIRoomKIt 的用户都需要调用 init 方法。

```
TUIRoomRef.value.init(roomData);
```

参数如下表所示：

参数	类型	含义
roomData	object	房间初始化参数
roomData.sdkAppId	number	客户的 SDKAppId
roomData.userId	string	用户的唯一 ID
roomData.userSig	string	用户的 UserSig
roomData.userName	string	用户的昵称
roomData.avatarUrl	string	用户的头像地址
roomData.theme	object	界面的主题风格
roomData.theme.defaultTheme	string	默认主题皮肤颜色
roomData.theme.isSupportSwitchTheme	boolean	是否支持切换主题

createRoom

主持人创建房间。

```
TUIRoomRef.value.createRoom(roomId, roomMode, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomMode	string	房间模式，**FreeToSpeak**（自由发言模式）和 **SpeakAfterTakingSeat**（上台发言模式），默认为 **FreeToSpeak**
roomParam	object	非必填
roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭

enterRoom

普通成员加入房间。

```
TUIRoomRef.value.enterRoom(roomId, roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
roomParam	object	非必填

版权所有：腾讯云计算（北京）有限责任公司

roomParam.isOpenCamera	string	非必填，进房是否打开摄像头，默认为关闭
roomParam.isOpenMicrophone	string	非必填，进房是否打开麦克风，默认为关闭

TUIRoomkit 事件

onCreateRoom

创建房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-create-room="onCreateRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onCreateRoom(info) {
    if (info.code === 0) {
      console.log('创建房间成功')
    }
  }
</script>
```

onEnterRoom

进入房间回调。

```
<template>
  <room ref="TUIRoomRef" @on-enter-room="onEnterRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onEnterRoom(info) {
    if (info.code === 0) {
      console.log('进入房间成功')
    }
  }
</script>
```

onDestroyRoom

主持人销毁房间的回调。

```
<template>
  <room ref="TUIRoomRef" @on-destroy-room="onDestroyRoom"></room>
</template>
```

版权所有：腾讯云计算（北京）有限责任公司

```
<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onDestroyRoom(info) {
    if (info.code === 0) {
      console.log('主持人销毁成功')
    }
  }
</script>
```

onExitRoom

普通成员退出房间的回调。

```
<template>
  <room ref="TUIRoomRef" @on-exit-room="onExitRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onExitRoom(info) {
    if (info.code === 0) {
      console.log('普通成员退出房间成功')
    }
  }
</script>
```

onKickedOutOfRoom

普通成员被主持人踢出房间通知。

```
<template>
  <room ref="TUIRoomRef" @on-kicked-out-of-room="onKickedOutOfRoom"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOutOfRoom({ roomId, message }) {
    console.log('普通成员被主持人踢出房间', roomId, message);
  }
</script>
```

onKickedOffLine

用户账号在其他地方登录，被踢下线。

```
<template>
  <room ref="TUIRoomRef" @on-kick-off-line="onKickedOffLine"></room>
```

```
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onKickedOffLine({ message }) {
    console.log('成员被踢下线', message);
  }
</script>
```

onUserSigExpired

用户 userSig 过期，请重新获取 userSig。

```
<template>
  <room ref="TUIRoomRef" @on-user-sig-expired="onUserSigExpired"></room>
</template>

<script setup lang="ts">
  // 引入 TUIRoom 组件，注意确认引入路径是否正确
  import Room from './TUIRoom/index.vue';

  function onUserSigExpired() {
    console.log('用户 userSig 过期, 请重新生成 userSig.');
  }
</script>
```

TUIRoomEngine

最近更新时间：2024-03-04 14:32:51

TUIRoomEngine 简介

TUIRoomEngine SDK 提供了房间管理、多人实时音视频互动、屏幕分享、成员管理、即时聊天等功能。
安装方式：

```
// 使用 npm
npm i @tencentcloud/tuiroom-engine-wx --save

// 使用 pnpm
pnpm i @tencentcloud/tuiroom-engine-wx --save

// 使用 yarn
yarn add @tencentcloud/tuiroom-engine-wx
```

TUIRoomEngine API

TUIRoomEngine 静态方法

API	描述
once	监听 TUIRoomEngine ready 事件。 **注意：TUIRoomEngine.login 以外的所有方法必须在监听到 TUIRoomEngine ready 事件且 TUIRoomEngine.login 方法执行成功后执行。**
login	登录 TUIRoomEngine
setSelfInfo	设置当前用户基本信息（用户名、用户头像）
getSelfInfo	获取当前用户基本信息（用户名、用户头像）
logout	登出 TUIRoomEngine

roomEngine 房间管理 API

API	描述
createRoom	创建房间
enterRoom	进入房间
destroyRoom	销毁房间
exitRoom	离开房间
fetchRoomInfo	获取房间信息
updateRoomNameByAdmin	更新房间的名字（仅群主或者管理员可以调用）
~~updateRoomSpeechModeByAdmin~~	更新房间的发言模式（仅群主或者管理员可以调用），该接口自 v2.0.0 版本废弃
updateRoomSeatModeByAd	更新房间上麦模式（仅群主或者管理员可以调用）

min	
getUserList	获取当前房间用户列表
getUserInfo	获取用户的详细信息

roomEngine 音视频 API

API	描述
setLocalVideoView	设置本地流的渲染位置
openLocalCamera	本地摄像头视频流采集
closeLocalCamera	关闭本地摄像头
openLocalMicrophone	打开本地麦克风
closeLocalMicrophone	关闭本地麦克风
updateVideoQuality	设置本地视频的分辨率
setVideoResolutionMode	设置本地视频流的分辨率模式
updateVideoQualityEx	设置本地视频的编码参数
updateAudioQuality	设置本地音频的参数
startPushLocalVideo	开始向远端推本地视频流
stopPushLocalVideo	停止向远端推本地视频流
muteLocalAudio	开始/停止向远端推本地音频流
setRemoteVideoView	设置远端流渲染的区域
startPlayRemoteVideo	开始播放远端用户视频流
stopPlayRemoteVideo	停止播放远端用户视频流
muteRemoteAudioStream	停止远端用户的音频流

roomEngine 成员管理 API

API	描述
openRemoteDeviceByAdmin	请求远端用户打开媒体设备
applyToAdminToOpenLocalDevice	参会者向主持人申请打开设备
closeRemoteDeviceByAdmin	关闭远端用户媒体设备
cancelRequest	取消已经发出的请求
responseRemoteRequest	回复远端用户的请求
changeUserRole	改变用户的角色
kickRemoteUserOutOfRoom	将用户移出房间

版权所有：腾讯云计算（北京）有限责任公司

disableDeviceForAllUserByAdmin	禁用/打开所有用户的媒体设备
disableSendingMessageForAllUser	禁止/允许所有用户发送消息
disableSendingMessageByAdmin	禁止/允许某个用户发送消息

roomEngine 麦位管理 API

API	描述
setMaxSeatCount	设置房间麦位最大值
getSeatList	获取麦位信息
takeSeat	获取麦位
leaveSeat	释放麦位
takeUserOnSeatByAdmin	邀请其他人上麦（仅房间主持人和管理员可调用此方法）
kickUserOffSeatByAdmin	将其他人移下麦（仅房间主持人和管理员可调用此方法）
lockSeatByAdmin	锁定某个麦位状态（仅房间主持人和管理员可调用此方法）
getSeatApplicationList	获取上麦请求列表，该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）

~~roomEngine 消息发送 API~~（该模块自 v2.0.0 版本废弃，聊天模块请使用 chat 实例实现）

API	描述
~~sendTextMessage~~	发送文本消息，该接口自 v2.0.0 版本废弃
~~sendCustomMessage~~	发送自定义消息，该接口自 v2.0.0 版本废弃

roomEngine 事件监听 API

API	描述
on	监听 TUIRoomEvents 事件
off	取消监听 TUIRoomEvents 事件

roomEngine 其他 API

API	描述
getTRTCCloud	获取 trtcCloud 实例
getTIM	获取 tim 实例

API 详情

once

监听 TUIRoomEngine 'ready' 事件。

```
TUIRoomEngine.once('ready', () => {
  const roomEngine = new TUIRoomEngine();

  await TUIRoomEngine.login({
    sdkAppId: 0,   // 填写您申请的 sdkAppId
    userId: '',    // 填写您业务对应的 userId
    userSig: '',   // 填写服务器或本地计算的 userSig
  });

  await roomEngine.createRoom({
    roomId: '12345',   // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
    name: 'Test Room',     // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
    roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
  });
});
```

login

登录 TUIRoomEngine， 您必须在登录 TUIRoomEngine 之后才可以调用 TUIRoomEngine 及其实例的其他方法。

```
// 登录 TUIRoomEngine
await TUIRoomEngine.login({
 sdkAppId: 0,   // 填写您申请的 sdkAppId
 userId: '',    // 填写您业务对应的 userId
 userSig: '',   // 填写服务器或本地计算的 userSig
});
```

参数：

参数	类型	说明	默认值	含义
sdkAppId	number	必填	-	在 实时音视频控制台 单击**应用管理** > **创建应用**创建新应用之后，即可在 应用信息 中获取 sdkAppId 信息。
userId	string	必填	-	用户ID建议限制长度为32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。
userSig	string	必填	-	userSig 签名 计算 userSig 的方式请参考 UserSig 相关 。
tim	TIM	非必填	-	若您在接入 roomEngine 的同时想使用即时通信 SDK 的更多能力，您可以将创建的 tim 实例传入 TUIRoomEngine。tim 实例的创建方式请参考 TIM.create 。

Returns *Promise<void>*

setSelfInfo

设置当前用户基本信息（用户名、用户头像）。

```
// 设置当前用户用户名及用户头像
```

```
await TUIRoomEngine.setSelfInfo({
 userName: '',    // 填写您的新用户名
 avatarUrl: '',   // 填写您的新头像地址
});

// 设置当前用户用户名，用户头像及用户自定义消息
await TUIRoomEngine.setSelfInfo({
 userName: '',    // 填写您的新用户名
 avatarUrl: '',   // 填写您的新头像地址
 customInfo: {    // 填写用户自定义信息
   Tag_Profile_Custom_Test: 'value1',
   Tag_Profile_Custom_profile: 'value2',
 },
})
```

参数：

参数	类型	说明	默认值	含义
userName	string	必填	-	用户名
avatarUrl	string	必填	-	用户头像
customInfo	object	选填	-	用户自定义资料

> **说明：**
> App 管理员可以在 即时通信 IM 控制台 > 用户自定义字段 新增自定义资料字段，用户自定义字段生效以后，用户可以调用 setSelfInfo 接口设置自定义资料。

Returns *Promise<void>*

getSelfInfo

获取当前用户基本信息（用户名、用户头像）。

```
// 获取当前用户用户名及用户头像
const loginUserInfo = await TUIRoomEngine.getSelfInfo();
```

Returns *Promise<TUILoginUserInfo > loginUserInfo*

logout

登出 TUIRoomEngine。

```
// 登出 TUIRoomEngine
await TUIRoomEngine.logout();
```

Returns *Promise<void>*

createRoom

主持人创建房间，调用 createRoom 的用户为房间的所有者。创建房间时可设置房间 ID、房间名称、房间类型、是否开启麦位控制、是否允许加入用户开启音视频、发送消息等功能。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({
    roomId: '12345',   // 填入您的房间 Id, 注意房间 Id 要求为字符串类型
    roomName: 'Test Room',     // 填入您的房间名称, 房间名称默认为 roomId，最长 30 字节
    roomType: TUIRoomType.kConference, // 设置房间类型为 TUIRoomType.kConference 类型
    isSeatEnabled: false, // 设置是否开启麦位控制
    isMicrophoneDisableForAllUser: false,  // 设置是否允许加入用户打开麦克风
    isCameraDisableForAllUser: false,  // 设置是否允许加入用户打开摄像头
    isMessageDisableForAllUser: false,  // 设置是否允许加入用户发送消息
});
```

参数：

参数	类型	说明	默认值	含义
roomId	string	必填	–	房间 ID，roomId 限制长度为64字节，且仅支持以下范围的字符集： • 大小写英文字母（a-zA-Z） • 数字（0-9） • 空格 ! # $ % & ( ) + - : ; < = . > ? @ [ ] ^ _ { } | ~ ,
roomName	string	选填	roomId	房间名称，默认值为 roomId, 不可传入空字符串
roomType	TUIRoomType	选填	TUIRoomType.kConference	房间类型 • 办公协同、医疗问诊、远程会议、教育场景，roomType 设置为 TUIRoomType.kConference • 电商直播、语聊房场景，roomType 设置为 TUIRoomType.kLivingRoom
isSeatEnabled	boolean	选填	false	是否开启麦位控制，默认不开启
seatMode	TUISeatMode	选填	TUISeatMode.kFreeToTake	上麦模式(开启麦位控制后生效)，默认值为 TUISeatMode.kFreeToTake 自由上麦模式，台下观众可以自由上麦，无需申请，seatMode 设置为 TUISeatMode.kFreeToTake 申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦，seatMode 设置为 TUISeatMode.kApplyToTake
isMicrophoneDisableForAllUser	boolean	选填	false	是否开启全员禁麦，默认不开启全员禁麦
isCameraDisableForAllUser	boolean	选填	false	是否开启全员禁画，默认不开启全员禁画
isMessageDisableForAllUser	boolean	选填	false	是否允许成员发送消息，默认不禁止
maxSeatCo	number	选填	–	最大麦位数量

unt				• roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制 • roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16
enableCDNStreaming	boolean	选填	false	是否开启 CDN 直播流
cdnStreamDomain	string	选填	''	直播推流域名

Returns *Promise<void>*

enterRoom

进入房间接口。

```
const roomEngine = new TUIRoomEngine();
const roomInfo = await roomEngine.enterRoom({
  roomId: '12345',
});
```

参数：

参数	类型	说明	默认值	含义
roomId	string	必填	-	房间号

Returns Promise<TUIRoomInfo> roomInfo
该接口返回当前房间信息。

destroyRoom

解散房间接口，解散房间必须由房间所有者发起，解散房间之后房间不可进入。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.destroyRoom();
```

Returns *Promise<void>*

exitRoom

离开房间接口，用户在执行 enterRoom 之后可通过 exitRoom 离开房间。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.exitRoom();
```

Returns *Promise<void>*

fetchRoomInfo

获取房间信息。

```
const roomEngine = new TUIRoomEngine();
const roomInfo = roomEngine.fetchRoomInfo();
```

Returns：*Promise<TUIRoomInfo>roomInfo*

updateRoomNameByAdmin

更新当前房间的名字（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomNameByAdmin({ roomName: '新的名字' });
```

参数：

参数	类型	说明	默认值	含义
roomName	string	必填	-	更新房间的名字，要求 roomName 不为空字符串

updateRoomSpeechModeByAdmin

更新房间的发言模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSpeechModeByAdmin({
  speechMode: TUISpeechMode.kSpeakAfterTakingSeat  // 更新为上麦后发言模式
});
```

参数：

参数	类型	说明	默认值	含义
speechMode	TUISpeechMode	必填	-	更新房间的发言模式

> **注意：**
> 该接口自 v2.0.0 版本废弃

updateRoomSeatModeByAdmin

更新房间上麦模式（仅群主或者管理员可以调用）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.updateRoomSeatModeByAdmin({
  seatMode: TUISeatMode.kApplyToTake,  // 更新为申请上麦模式
});
```

版权所有：腾讯云计算（北京）有限责任公司

参数：

参数	类型	说明	默认值	含义
seatMode	TUISeatMode	必填	-	发言类型

getUserList

获取当前房间用户列表，注意该接口一次拉取的用户列表量最大为 50 个。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
let result;
let nextSequence = 0;
do {
  result = await roomEngine.getUserList({ nextSequence });
  userList.push(...result.userInfoList);
  nextSequence = result.nextSequence;
} while (result.nextSequence !== 0)
```

参数：

参数	类型	说明	默认值	含义
nextSequence	number	选填	0	偏移量，默认从 0 开始拉取用户

Returns：*Promise<Array> result*
result.nextSequence 下一次拉取群组用户的偏移量，如果 result.nextSequence 为 0 则代表 userList 全部拉取完毕。
result.userInfoList 本次拉取的 userList。

getUserInfo

获取用户的详细信息。

```
const roomEngine = new TUIRoomEngine();
const userList = [];
const userInfo = await roomEngine.getUserInfo({
   userId: 'user_12345',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	根据 userId 获取该用户的详细信息

Returns：*Promise<Array<TUIUserInfo >> userInfoList*
该接口返回指定用户的用户信息。

setLocalVideoView

设置本地流的渲染位置。

```
const roomEngine = new TUIRoomEngine();

// 设置本地摄像头流的播放区域为 id 是 'preview-camera' 的 div 元素
await roomEngine.setLocalVideoView({
  view: 'preview-camera',
});
```

参数：

参数	类型	说明	默认值	含义
view	string	必填	-	streamType 对应的流渲染的 div 元素的 id

Returns：*Promise<void>*

> **说明：**
> 自 v1.6.1 版本起，setLocalVideoView 只需传入 view 参数，设置本地视频流的播放容器。

openLocalCamera

打开本地摄像头，开始视频流采集。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalVideoView({
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'preview-camera',
});

// case 1: 桌面端打开摄像头
await roomEngine.openLocalCamera();

// case 2: 移动端使用前置摄像头打开视频预览
await roomEngine.openLocalCamera({ isFrontCamera: true });
// case 3: 移动端使用后置摄像头打开视频预览
await roomEngine.openLocalCamera({ isFrontCamera: false });
```

参数：

参数	类型	说明	默认值	含义
isFrontCamera	boolean	选填	-	设置移动端打开 本地预览视频的默认摄像头，该参数仅对移动端有效

Returns：*Promise<void>*

closeLocalCamera

关闭本地摄像头。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalCamera();
```

Returns：*Promise<void>*

openLocalMicrophone

打开本地麦克风，开始采集音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.openLocalMicrophone();
```

Returns：*Promise<void>*

closeLocalMicrophone

关闭本地麦克风。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeLocalMicrophone();
```

Returns：*Promise<void>*

updateVideoQuality

设置本地视频流的编码参数，默认为 TUIVideoProfile.kVideoQuality_720P。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQuality({
  quality: TUIVideoQuality.kVideoQuality_540p,
});
```

参数：

参数	类型	说明	默认值	含义
quality	TUIVideoQuality	必填	–	• 清晰 TUIVideoProfile.kVideoQuality_360P • 标清 TUIVideoProfile.kVideoQuality_540P • 高清 TUIVideoProfile.kVideoQuality_720P • 超清 TUIVideoProfile.kVideoQuality_1080P

Returns：*Promise<void>*

setVideoResolutionMode

设置本地视频流的分辨率模式，默认为 TUIResolutionMode.kResolutionMode_Landscape。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setVideoResolutionMode({
  type: TUIVideoStreamType.kCameraStream,
  resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
resolutionMode	TUIResolutionMode	必填	TUIResolutionMode.kResolutionMode_Landscape	分辨率模式

Returns：*Promise<void>*

updateVideoQualityEx

设置本地视频流的编码参数。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.updateVideoQualityEx({
  streamType: TUIVideoStreamType.kCameraStream,
  encoderParams: {
    quality: TUIVideoQuality.kVideoQuality_720p,
    fps: 15,
    bitrate: 2000,
    resolutionMode: TUIResolutionMode.kResolutionMode_Landscape,
  }
});
```

参数：

参数	类型	说明	默认值	含义
streamType	TUIVideoStreamType	必填	-	本地流类型
encoderParams	TUIVideoEncoderParams	必填	-	详细编码参数

updateAudioQuality

设置本地音频的参数。

> ⚠ **注意：**
> 该方法需要在 openLocalMicrophone 之前进行设置，否则不会生效。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.setLocalAudioProfile({
   audioProfile: TUIAudioProfile.kAudioProfileSpeech,
});
```

参数：

参数	类型	说明	默认值	含义
audioProfile	TUIAudioProfile	必填	-	• TUIAudioProfile.kAudioProfileSpeech：语言模式；采样率：16k • TUIAudioProfile.kAudioProfileDefault：标准模式（或者

				默认模式）；采样率：48k • TUIAudioProfile.kAudioProfileMusic：音乐模式；采样率：48k

Returns：*Promise<void>*

startPushLocalVideo

进房后会默认向远端推本地视频流，该接口用于停止推流后重新向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalVideo();
```

Returns：*Promise<void>*

stopPushLocalVideo

停止向远端推本地视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalVideo();
```

Returns：*Promise<void>*

startPushLocalAudio

进房后会默认向远端推本地音频流，该接口用于停止推流后重新向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPushLocalAudio();
```

Returns：*Promise<void>*

stopPushLocalAudio

停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPushLocalAudio();
```

Returns：*Promise<void>*

muteLocalAudio

开始/停止向远端推本地音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteLocalAudio({ isMute: true });
```

参数：

参数	类型	说明	默认值	含义

isMute	boolean	必填	false	开始/停止向远端推本地音频流

Returns：*Promise<void>*

setRemoteVideoView

设置远端流渲染的区域。

```
const roomEngine = new TUIRoomEngine();

// 设置远端用户视频流在 id 为 'remote_preview_camera' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
  view: 'remote_preview_camera',
});
// 设置远端用户屏幕分享流在 id 为 'remote_preview_screen' 的区域播放
await roomEngine.setRemoteVideoView({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kScreenStream,
  view: 'remote_preview_screen',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型
view	string	必填	-	播放远端用户流的 div 元素的 id

Returns：*Promise<void>*

startPlayRemoteVideo

开始播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.startPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流

				• TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns：*Promise<void>*

stopPlayRemoteVideo

停止播放远端用户视频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.stopPlayRemoteVideo({
  userId: 'user_1234',
  streamType: TUIVideoStreamType.kCameraStream,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
streamType	TUIVideoStreamType	必填	-	用户流类型 • TUIVideoStreamType.kCameraStream 视频流 • TUIVideoStreamType.kScreenStream 屏幕分享流 • TUIVideoStreamType.kCameraStreamLow 低清视频流

Returns：*Promise<void>*

muteRemoteAudioStream

停止远端用户的音频流。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.muteRemoteAudioStream({
 userId: 'user_1234',
 isMute: true,
});
```

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isMute	boolean	必填	-	是否停止远端用户的音频

openRemoteDeviceByAdmin

请求远端用户打开媒体设备。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.openRemoteDeviceByAdmin({
   userId: 'user_1234',
   device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
```

```
    timeout: 0,
    requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
     switch (requestCallbackType) {
       case TUIRequestCallbackType.kRequestAccepted:
         // 请求被接受
         break;
       case TUIRequestCallbackType.kRequestRejected:
         // 请求被拒绝
         break;
       case TUIRequestCallbackType.kRequestCancelled:
         // 请求已取消
         break;
       case TUIRequestCallbackType.kRequestTimeout:
         // 请求超时
         break;
       case TUIRequestCallbackType.kRequestError:
         // 请求错误
         break;
       default:
         break;
     }
    },
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

applyToAdminToOpenLocalDevice

参会者向主持人申请打开设备。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.applyToAdminToOpenLocalDevice({
    device: TUIMediaDevice.kMicrophone  //请求的设备为麦克风
    timeout: 0,
    requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
     switch (requestCallbackType) {
       case TUIRequestCallbackType.kRequestAccepted:
         // 请求被接受
         break;
       case TUIRequestCallbackType.kRequestRejected:
```

```
        // 请求被拒绝
        break;
      case TUIRequestCallbackType.kRequestCancelled:
        // 请求已取消
        break;
      case TUIRequestCallbackType.kRequestTimeout:
        // 请求超时
        break;
      case TUIRequestCallbackType.kRequestError:
        // 请求错误
        break;
      default:
        break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string> requestId*
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

closeRemoteDeviceByAdmin

关闭远端用户媒体设备。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.closeRemoteDeviceByAdmin({
  userId: 'user_1234',
  device: TUIMediaDevice.kMicrophone, //关闭麦克风
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
device	TUIMediaDevice	必填	-	媒体设备类型（摄像头/麦克风/屏幕共享）

Returns：*Promise<void>*

cancelRequest

取消已经发出的请求。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.cancelRequest({
  requestId: '',    // 请使用实际 requestId
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID

Returns：*Promise<void>*

responseRemoteRequest

回复远端用户的请求。

```
const roomEngine = new TUIRoomEngine();
// 同意远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: true,
});
// 拒绝远端的请求
await roomEngine.responseRemoteRequest({
 requestId: '',    // 请使用实际 requestId
 agree: false,
});
```

参数：

参数	类型	说明	默认值	含义
requestId	string	必填	-	请求 ID
agree	boolean	必填	-	是否同意

Returns：*Promise<void>*

disableDeviceForAllUserByAdmin

禁止/允许所有用户打开媒体设备（此接口对于房主和管理员无效）。

```
// Example 1: 禁止所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: true,
})
// Example 2: 允许所有用户打开麦克风
await roomEngine.disableDeviceForAllUserByAdmin({
  device: TUIMediaDevice.kMicrophone,
  isDisable: false,
```

版权所有：腾讯云计算（北京）有限责任公司

```
})
```

参数：

参数	类型	说明	默认值	含义
device	TUIMediaDevice	必填	-	被禁用的媒体设备类型（摄像头/麦克风/屏幕共享）
isDisable	boolean	必填	-	是否被禁止

Returns：*Promise<void>*

disableSendingMessageForAllUser

所有用户的是否允许发消息（此接口对于房主和管理员无效）。

```
await roomEngine.disableSendingMessageForAllUser({
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

disableSendingMessageByAdmin

特定用户是否被允许发消息。

```
await roomEngine.disableSendingMessageByAdmin({
  userId: 'user_1234',
  isDisable: true,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
isDisable	boolean	必填	-	是否被禁用

Returns：*Promise<void>*

changeUserRole

改变用户的角色（仅主持人可调用该接口）。

```
const roomEngine = new TUIRoomEngine();
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 将房间移交给用户 user_1234
await roomEngine.changeUserRole({
  userId: 'user_1234',
  role: TUIRole.kRoomOwner,
});

// 将用户 user_1234 设置为房间管理员
await roomEngine.changeUserRole({
 userId: 'user_1234',
 userRole: TUIRole.kAdministrator,
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID
userRole	TUIRole	必填	-	用户角色 • 主持人 TUIRole.kRoomOwner • 管理员 TUIRole.kAdministrator • 普通成员 TUIRole.kGeneralUser

Returns：*Promise<void>*

kickRemoteUserOutOfRoom

将用户移出房间（仅主持人和管理员可调用该接口）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.kickRemoteUserOutOfRoom({
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
userId	string	必填	-	用户 ID

Returns：*Promise<void>*

setMaxSeatCount

设置房间麦位最大值
roomType 为 TUIRoomType.kConference (教育及会议场景) 时，maxSeatCount 值不做限制；
roomType 为 TUIRoomType.kLivingRoom (直播场景) 时，maxSeatCount 最大限制为 16；

```
const roomEngine = new TUIRoomEngine();
await roomEngine.createRoom({ roomId: '12345' });
await roomEngine.setMaxSeatCount({ maxSeatCount: 16 })
```

参数：

参数	类型	说明	默认值	含义
maxSeatCount	number	必填	-	设置房间麦位最大值

Returns：Promise<void>

getSeatList

获取麦位列表。

```
const roomEngine = new TUIRoomEngine();
const seatList = await roomEngine.getSeatList();
```

Returns：*Promise*<TUISeatInfo *[]*> seatList
seatList 为当前房间所有麦位列表。

takeSeat

麦下用户可调用 takeSeat 成为麦上用户，仅麦上用户可发布本地音视频流。

- 当 roomInfo.roomType 为 TUIRoomType.kConference 且 roomInfo.seatMode 为 TUISeatMode.kApplyToTake 时，普通用户调用 takeSeat 方法需要等待主持人/管理员的同意后成为麦上用户。
- 当 roomInfo.roomType 为 TUIRoomType.kLivingRoom 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 时，普通用户调用 takeSeat 方法成功后即为麦上用户。
- 主持人&管理员调用 takeSeat 成功后即为麦上用户。
- 麦上用户的变更通过 TUIRoomEvents.onSeatListChanged 通知所有用户。

```
const roomEngine = new TUIRoomEngine();
// 情景一：主持人/管理员上麦
// 情景二：当 roomInfo.roomType 为 TUIRoomType.kConference
// 且 roomInfo.seatMode 为 TUISeatMode.kFreeToTake 模式时，普通用户上麦
await roomEngine.takeSeat({
 seatIndex: -1,
 timeout: 0,
});

// 情景三：roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户上麦
const requestId = await roomEngine.instance?.takeSeat({
  seatIndex: -1,
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
    switch (requestCallbackType) {
     case TUIRequestCallbackType.kRequestAccepted:
       // 请求被接受
       break;
     case TUIRequestCallbackType.kRequestRejected:
       // 请求被拒绝
       break;
     case TUIRequestCallbackType.kRequestCancelled:
       // 请求已取消
       break;
     case TUIRequestCallbackType.kRequestTimeout:
       // 请求超时
       break;
```

```
      case TUIRequestCallbackType.kRequestError:
        // 请求错误
        break;
      default:
        break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index, 无麦位序号时设置为 -1
timeout	number	必填	-	超时时间。若 timeout 设置为 0，则无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：Promise<string> requestId
roomInfo.seatMode 为 TUISeatMode.kApplyToTake 模式时，普通用户调用该接口时返回 requestId，普通用户可使用该 requestId 调用 cancelRequest 接口取消上麦请求。

leaveSeat

释放麦位。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.leaveSeat();
```

Returns：*Promise<void>*

takeUserOnSeatByAdmin

邀请其他人上麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = roomEngine.takeUserOnSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
  timeout: 0,
  requestCallback: ({ requestCallbackType, requestId, userId, code, message }) => {
   switch (requestCallbackType) {
    case TUIRequestCallbackType.kRequestAccepted:
      // 请求被接受
      break;
    case TUIRequestCallbackType.kRequestRejected:
      // 请求被拒绝
      break;
    case TUIRequestCallbackType.kRequestCancelled:
      // 请求已取消
      break;
    case TUIRequestCallbackType.kRequestTimeout:
      // 请求超时
```

```
        break;
      case TUIRequestCallbackType.kRequestError:
        // 请求错误
        break;
      default:
        break;
    }
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID
timeout	number	必填	-	超时时间，设置为 0 时，无超时时间
requestCallback	Function	选填	空函数	请求回调，用来通知发起方请求被接受/拒绝/取消/超时/错误的回调

Returns：*Promise<string>* requestId
该接口返回 requestId，用户可使用该 requestId 调用 cancelRequest 接口取消请求。

kickUserOffSeatByAdmin

要求其他人下麦。

```
const roomEngine = new TUIRoomEngine();
const requestId = await roomEngine.kickUserOffSeatByAdmin({
  seatIndex: 0,
  userId: 'user_1234',
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位 index
userId	string	必填	-	用户 ID

Returns：*Promise<void>*

lockSeatByAdmin

锁定某个麦位状态（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.lockSeatByAdmin({
  seatIndex: 0,
  lockParams: {
```

```
    lockSeat: true,
    lockVideo: true,
    lockAudio: true,
  },
});
```

参数：

参数	类型	说明	默认值	含义
seatIndex	number	必填	-	麦位索引
lockParams	TUISeatLockParams	必填	-	锁麦参数

Returns：*Promise<void>*

getSeatApplicationList

获取房间内申请上麦用户的请求列表, 该接口自 v2.1.0 版本支持（仅房间主持人和管理员可调用此方法）。

```
const roomEngine = new TUIRoomEngine();
const applicationList = await roomEngine.getSeatApplicationList();
```

Returns：*Promise*<TUIRequest[]>

sendTextMessage

发送文本消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendTextMessage({
 messageText: 'hello, everyone',
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	文本消息内容

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

sendCustomMessage

发送自定义消息。

```
const roomEngine = new TUIRoomEngine();
await roomEngine.sendCustomMessage({
 messageText: '{ data:'', description: ''}',
```

```
});
```

参数：

参数	类型	说明	默认值	含义
messageText	string	必填	-	自定义消息内容

Returns：*Promise<void>*

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃。请使用 getTIM 方法获取 tim 实例发送消息。

switchCamera

切换前后置摄像头，该接口适用于移动端。

```
// 移动端切换为前置摄像头
await roomEngine.switchCamera({ isFrontCamera: true });

// 移动端切换为后置摄像头
await roomEngine.switchCamera({ isFrontCamera: false });
```

参数：

参数	类型	说明	默认值	含义
isFrontCamera	boolean	必填	-	是否切换为前置摄像头

Returns：*Promise<void>*

on

监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(event, func);
```

参数：

参数	类型	说明	默认值	含义
event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns：*void*

off

取消监听 roomEngine 的事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.off(event, func);
```

参数：

参数	类型	说明	默认值	含义
event	TUIRoomEvents	必填	-	TUIRoomEngine 事件列表
func	Function	必填	-	事件回调函数

Returns：*void*

getTRTCCloud

获取 trtcCloud 实例。

```
const roomEngine = new TUIRoomEngine();
const trtcCloud = roomEngine.getTRTCCloud();
```

Returns：TRTCCloud

getTIM

获取 tim 实例, web 端 tim 能力请查看： IM API 文档 。

```
const roomEngine = new TUIRoomEngine();
const tim = roomEngine.getTIM();
```

Returns： tim

> ⚠ **注意：**
> 自 v1.6.0 版本后，IM SDK 升级至 v3 版本，请查看： IM API 文档 。

TUIRoomEvents

最近更新时间：2024-03-04 14:32:51

TUIRoomEvent API 简介

TUIRoomEvent API 是音视频通话组件的**事件接口**。

事件列表

EVENT	描述
TUIRoomEvents.onError	错误事件
TUIRoomEvents.onKickedOutOfRoom	移出房间事件
TUIRoomEvents.onKickedOffLine	当前用户被移下线事件
TUIRoomEvents.onUserSigExpired	userSig 过期事件
TUIRoomEvents.onRoomDismissed	主持人销毁房间事件
TUIRoomEvents.onRoomNameChanged	房间名称修改事件
~~TUIRoomEvents.onRoomSpeechModeChanged~~	房间发言模式修改事件，该接口自 v2.0.0 版本废弃
TUIRoomEvents.onRoomSeatModeChanged	上麦模式修改事件
TUIRoomEvents.onAllUserCameraDisableChanged	所有成员摄像头使用权限改变事件
TUIRoomEvents.onAllUserMicrophoneDisableChanged	所有成员麦克风使用权限改变事件
TUIRoomEvents.onSendMessageForAllUserDisableChanged	所有成员发送消息状态改变事件
TUIRoomEvents.onRoomMaxSeatCountChanged	房间最大麦位数修改事件
TUIRoomEvents.onRemoteUserEnterRoom	远端用户进入房间事件
TUIRoomEvents.onRemoteUserLeaveRoom	远端用户离开房间事件
TUIRoomEvents.onUserRoleChanged	用户角色改变事件
TUIRoomEvents.onUserVideoStateChanged	用户视频状态改变事件
TUIRoomEvents.onUserAudioStateChanged	用户音频状态改变事件
TUIRoomEvents.onSendMessageForUserDisableChanged	用户发送消息状态事件
TUIRoomEvents.onUserVoiceVolumeChang	用户音量改变事件

版权所有：腾讯云计算（北京）有限责任公司

ed	
TUIRoomEvents.onUserNetworkQualityChanged	用户网络质量改变事件
TUIRoomEvents.onSeatListChanged	麦位列表改变事件
TUIRoomEvents.onKickedOffSeat	用户被移下麦事件
TUIRoomEvents.onRequestReceived	请求接收事件
TUIRoomEvents.onRequestProcessed	请求被处理事件
TUIRoomEvents.onRequestCancelled	请求取消事件
~~TUIRoomEvents.onReceiveTextMessage~~	接收文本消息事件，该接口自 v2.0.0 版本废弃
~~TUIRoomEvents.onReceiveCustomMessage~~	接收自定义消息事件，该接口自 v2.0.0 版本废弃

onError

错误事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onError, (error) => {
 console.log('TUIRoomError error', error);
})
```

参数如下表所示：

参数	类型	含义
code	number	错误代码
message	string	错误信息

onKickedOutOfRoom

移出房间事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOutOfRoom, ({ roomId, reason, message }) => {
  console.log('roomEngine.onKickedOutOfRoom', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
reason	TUIKickedOutOfRoomReason	用户被移出房间原因
message	string	移出房间信息

onKickedOffLine

当前用户被移下线事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onKickedOffLine, ({ message }) => {
  console.log('roomEngine.onKickedOffLine', message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
message	string	用户在其他端登录信息

onUserSigExpired

userSig 过期事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserSigExpired, () => {
  console.log('roomEngine.onUserSigExpired');
});
```

onRoomDismissed

主持人销毁房间事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomDismissed, ({ roomId }) => {
  console.log('roomEngine.onRoomDismissed', roomId);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号

onRoomNameChanged

房间名称修改事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomNameChanged, ({ roomId, roomName }) => {
  console.log('roomEngine.onRoomNameChanged', roomId, roomName);
});
```

参数如下表所示：

版权所有：腾讯云计算（北京）有限责任公司

参数	类型	含义
roomId	string	房间号
roomName	string	房间名称

onRoomSpeechModeChanged

房间模式修改事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSpeechModeChanged, ({ roomId, speechMode }) => {
  console.log('roomEngine.onRoomSpeechModeChanged', roomId, speechMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
speechMode	TUISpeechMode	发言模式

> **注意：**
> 该接口自 v2.0.0 版本废弃。

onRoomSeatModeChanged

上麦模式修改事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomSeatModeChanged, ({ roomId, seatMode }) => {
  console.log('roomEngine.onRoomSeatModeChanged', roomId, seatMode);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
seatMode	TUISeatMode	房间上麦模式

onAllUserCameraDisableChanged

所有成员摄像头使用权限改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserCameraDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserCameraDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许使用摄像头

onAllUserMicrophoneDisableChanged

所有成员麦克风使用权限改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onAllUserMicrophoneDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onAllUserMicrophoneDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许使用麦克风

onSendMessageForAllUserDisableChanged

所有成员发送消息权限改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onRoomMaxSeatCountChanged

房间最大麦位数修改事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRoomMaxSeatCountChanged, ({ maxSeatNumber }) => {
  console.log('roomEngine.onRoomMaxSeatCountChanged', maxSeatNumber);
});
```

参数如下表所示：

参数	类型	含义
maxSeatNumber	number	最大麦位数量

onRemoteUserEnterRoom

远端用户进入房间事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserEnterRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserEnterRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onRemoteUserLeaveRoom

远端用户离开房间事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRemoteUserLeaveRoom, ({ roomId, userInfo }) => {
  console.log('roomEngine.onRemoteUserLeaveRoom', roomId, userInfo);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间号
userInfo	TUIUserInfo	用户信息

onUserRoleChanged

用户角色改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserRoleChanged, ({ userId, userRole }) => {
  console.log('roomEngine.onUserRoleChanged', userId, userRole);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
userRole	TUIRole	用户变更后的角色

onUserVideoStateChanged

用户视频状态改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVideoStateChanged, ({ userId, streamType, hasVideo, reason }) => {
  console.log('roomEngine.onUserVideoStateChanged', userId, streamType, hasVideo, reason);
```

版权所有：腾讯云计算（北京）有限责任公司

```
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
streamType	TUIVideoStreamType	用户流类型
hasVideo	boolean	是否有视频流
reason	TUIChangeReason	变更原因, 自己操作/主持人操作

onUserAudioStateChanged

用户音频状态改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserAudioStateChanged, ({ userId, hasAudio, reason }) => {
  console.log('roomEngine.onUserAudioStateChanged', userId, hasAudio, reason);
});
```

参数如下表所示：

参数	类型	含义
userId	string	用户 ID
hasVideo	boolean	是否有音频流
reason	TUIChangeReason	变更原因，自己操作/主持人操作

onSendMessageForUserDisableChanged

有成员摄像头使用权限修改事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSendMessageForAllUserDisableChanged, ({ isDisable }) => {
  console.log('roomEngine.onSendMessageForAllUserDisableChanged', isDisable);
});
```

参数如下表所示：

参数	类型	含义
isDisable	boolean	是否允许发送文字消息

onUserVoiceVolumeChanged

用户音量改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserVoiceVolumeChanged, ({ userVolumeList }) => {
```

版权所有：腾讯云计算（北京）有限责任公司

```
  userVolumeList.forEach(userVolume => {
    console.log('roomEngine.onUserVoiceVolumeChanged', userVolume.userId, userVolume.volume);
  })
});
```

参数如下表所示：

参数	类型	含义
userVolumeList	Array.<object>	房间内所有用户的音量，包含 userId 及 volume 信息，volume 区间为 1~100

onUserNetworkQualityChanged

用户网络质量改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onUserNetworkQualityChanged, ({ userNetworkList }) => {
  userNetworkList.forEach(userNetwork => {
    console.log('roomEngine.onUserNetworkQualityChanged', userNetwork.userId, userNetwork.quality,
userNetwork.upLoss, userNetwork.downLoss, userNetwork.delay);
  })
});
```

参数如下表所示：

参数	类型	含义
userNetworkList	Array.<TUINetwork>	网络质量信息列表

onSeatListChanged

麦位列表改变事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onSeatListChanged, ({ seatList, seatedList, leftList }) => {
  console.log('roomEngine.onSeatListChanged',seatList, seatedList, leftList);
});
```

参数如下表所示：

参数	类型	含义
seatList	Array.<TUISeatInfo>	麦位列表
seatedList	Array.<TUISeatInfo>	新增麦位信息
leftList	Array.<TUISeatInfo>	离开的麦位信息

onKickedOffSeat

麦位列表改变事件。

```
const roomEngine = new TUIRoomEngine();
```

```
roomEngine.on(TUIRoomEvents.onKickedOffSeat, ({ userId }) => {
  console.log('roomEngine.onKickedOffSeat', userId);
});
```

参数如下表所示：

参数	类型	含义
userId	String	被移下麦的用户 ID

onRequestReceived

错误事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestReceived, ({ request }) => {
  console.log('roomEngine.onRequestReceived', request);
});
```

参数如下表所示：

参数	类型	含义
request	TUIRequest	请求接收

onRequestProcessed

请求被处理事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestProcessed, ({ request }) => {
  console.log('roomEngine.onRequestProcessed', request);
});
```

参数如下表所示：

参数	类型	含义
request	TUIRequest	请求接收

onRequestCancelled

请求取消事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onRequestCancelled, ({ requestId, userId }) => {
  console.log('roomEngine.onRequestCancelled', requestId, userId);
});
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

requestId	string	请求 ID
userId	string	取消请求的用户 ID

onReceiveTextMessage

接收文本消息事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveTextMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveTextMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收文本消息

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

onReceiveCustomMessage

接收自定义消息事件。

```
const roomEngine = new TUIRoomEngine();
roomEngine.on(TUIRoomEvents.onReceiveCustomMessage, ({ roomId, message }) => {
  console.log('roomEngine.onReceiveCustomMessage', roomId, message);
});
```

参数如下表所示：

参数	类型	含义
roomId	string	房间 ID
message	TUIMessage	接收自定义消息

> ⚠ **注意：**
> 该接口自 v2.0.0 版本废弃，请使用 tim 实例监听收消息事件。

版权所有：腾讯云计算（北京）有限责任公司

TUIRoomEngine Defines

最近更新时间：2024-02-21 15:15:22

本文介绍 TUIRoomEnigne web 端关键类型定义。

枚举值

TUIRole

用户角色，TUIRoomEngine 共提供三种用户角色，分别是主持人，管理员，普通用户。

字段	类型	描述
kRoomOwner	number	主持人角色
kAdministrator	number	管理员角色
kGeneralUser	number	普通用户角色

TUIVideoQuality

视频分辨率。

字段	类型	描述
kVideoQuality_360p	number	低清，分辨率为 640 * 360
kVideoQuality_540p	number	标清，分辨率为 960 * 540
kVideoQuality_720p	number	高清，分辨率为 1280 * 720
kVideoQuality_1080p	number	超清，分辨率为 1920 * 1080

TUIResolutionMode

分辨率模式（横屏分辨率 | 竖屏分辨率）。

字段	类型	描述
kResolutionMode_Landscape	number	横屏模式
kResolutionMode_Portrait	number	竖屏模式

TUIAudioQuality

音频分辨率。

字段	类型	描述
kAudioProfileSpeech	number	语音模式
kAudioProfileDefault	number	标准模式（默认模式）
kAudioProfileMusic	number	音乐模式

版权所有：腾讯云计算（北京）有限责任公司

TUIVideoStreamType

视频流类型。

字段	类型	描述
kCameraStream	number	摄像头视频流
kScreenStream	number	屏幕分享视频流
kCameraStreamLow	number	低清摄像头视频流

TUINetworkQuality

网络状态。

字段	类型	描述
kQualityUnknown	number	网络状况未知
kQualityExcellent	number	网络状况极佳
kQualityGood	number	网络状况较好
kQualityPoor	number	网络状况一般
kQualityBad	number	网络状况差
kQualityVeryBad	number	网络状况极差
kQualityDown	number	网络连接已断开

TUIRoomType

房间类型。

字段	类型	描述
kConference	number	kConference 类型房间，适用于会议、教育场景
kLivingRoom	number	kLivingRoom 类型房间，适用于直播场景

TUISpeechMode

发言类型。

字段	类型	描述
kFreeToSpeak	number	自由发言模式
kApplyToSpeak	number	举手发言模式
kSpeakAfterTakingSeat	number	就坐后发言（抢麦位）

> **注意：**
> 该类型自 v2.0.0 版本废弃

TUISeatMode

上麦模式。

字段	类型	描述
kFreeToTake	number	自由上麦模式，台下观众可以自由上麦，无需申请
kApplyToTake	number	申请上麦模式，台下观众上麦需要房主或者管理员同意后才能上麦

TUIChangeReason

更改原因（用户音视频状态变更操作原因: 自己主动修改 或者 被房主、管理员修改）。

字段	类型	描述
kChangedBySelf	number	自己操作
kChangedByAdmin	number	房主或管理员操作

TUIMediaDevice

字段	类型	描述
kMicrophone	number	麦克风
kCamera	number	摄像头
kScreen	number	屏幕分享

TUIRequestAction

房间类型。

字段	类型	描述
kInvalidAction	number	无效操作
kRequestToOpenRemoteCamera	number	请求远端打开摄像头
kRequestToOpenRemoteMicrophone	number	请求远端打开麦克风
kRequestToConnectOtherRoom	number	请求远端跨房连麦，web 端暂不支持
kRequestToTakeSeat	number	请求上麦
kRequestRemoteUserOnSeat	number	请求远端上麦
kApplyToAdminToOpenLocalCamera	number	向管理员请求打开本地摄像头
kApplyToAdminToOpenLocalMicrophone	number	向管理员请求打开本地麦克风

TUIRequestCallbackType

请求类型。

字段	类型	描述
kRequestAccepted	number	对端已接受
kRequestRejected	number	对端已拒绝
kRequestCancelled	number	请求已取消
kRequestTimeout	number	请求超时
kRequestError	number	请求错误

TUIKickedOutOfRoomReason

用户被移出房间原因。

字段	类型	描述
kKickedByAdmin	number	被主持人或管理员移出
kKickedByLoggedOnOtherDevice	number	相同 userId 用户进入同一房间被移出
kKickedByServer	number	被服务端移出

类型定义

TUILoginUserInfo

当前登录用户信息

名称	类型	描述
userId	string	登录用户的 ID
userName	string	登录用户的名称
avatarUrl	string	登录用户的头像
customInfo	object	登录用户的自定义资料，自 v1.3.0 版本支持

TUIRoomInfo

房间信息，用户可使用 roomEngine.getRoomInfo 获取房间信息。

名称	类型	描述
roomId	string	房间号，字符串类型房间号
roomName	string	房间名称
roomType	TUIRoomType	房间类型
speechMode	TUISpeechMode	房间内发言模式
isMicrophoneDisableFor	boolean	是否开启全员禁音（创建房间可选参数，默认值为 false）

版权所有：腾讯云计算（北京）有限责任公司

AllUser		
isCameraDisableForAllUser	boolean	是否开启全员禁画（创建房间可选参数， 默认值为 false）
isMessageDisableForAllUser	boolean	是否允许所有用户发消息（创建房间可选参数，默认值为 false）
maxSeatCount	number	最大麦位数量，默认为 6
roomOwner	string	房间主持人，只读
createTime	number	房间创建时间，只读，精确到秒
roomMemberCount	number	房间成员数量，只读

TUIUserInfo

用户信息。

名称	类型	描述
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址
userRole	TUIRole	用户角色
hasAudioStream	boolean	是否有音频流
hasVideoStream	boolean	是否有视频流
hasScreenStream	boolean	是否有屏幕分享流
roomCustomInfo	object	用户房间自定义字段，自 v1.3.0 版本支持

TUIMessage

消息信息。

名称	类型	描述
messageId	string	消息 ID
message	string	消息内容
timestamp	number	时间戳信息, 精确到秒
userId	string	用户 ID
userName	string	用户名称
avatarUrl	string	用户头像地址

TUIRequest

请求信息。

版权所有：腾讯云计算（北京）有限责任公司

名称	类型	描述
requestAction	TUIRequestAction	请求类型
timestamp	number	请求发起时间
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	发起请求的用户 ID
content	string	其他内容

TUIRequestCallback

请求回调信息。

名称	类型	描述
requestCallbackType	TUIRequestCallbackType	请求回调类型，接受/拒绝/取消/超时/错误
requestId	string	请求 ID v1.0.2 及其以上版本 requestId 类型为 string； v1.0.0 及 v1.0.1 版本 requestId 类型为 number；
userId	string	用户 ID
code	number	请求响应码
message	string	请求状态补充说明

TUISeatInfo

麦位信息。

名称	类型	描述
index	number	麦位序号
userId	string	麦位对应的用户 ID
locked	boolean	当前麦位是否被锁
videoMuted	boolean	当前麦位是否禁止视频
audioMuted	boolean	当前麦位是否禁止音频

TUISeatLockParams

麦位锁定状态。

名称	类型	描述
lockSeat	boolean	锁定麦位
lockVideo	boolean	锁定麦位视频

版权所有：腾讯云计算（北京）有限责任公司

lockAudio	boolean	锁定麦位音频

TUINetwork

网络信息。

名称	类型	描述
userId	string	用户 ID
quality	TUINetworkQuality	网络质量
upLoss	number	上行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
downLoss	number	下行丢包率，单位 (%) 该数值越小越好，目前仅本地用户有该信息
delay	number	网络延迟，单位 ms，目前仅本地用户有该信息

TUIVideoEncoderParams

视频编码参数。

名称	类型	描述
videoResolution	TUIVideoQuality	视频分辨率, 该字段在 v1.6.0 版本由 quality 修改为 videoResolution
fps	number	视频帧率
bitrate	number	视频码率
resolutionMode	TUIResolutionMode	横竖屏模式

发布日志

最近更新时间：2024-03-06 14:57:31

2024年3月

发布动态	更新日志	发布时间
多人音视频房间（TUIRoomEngine）SDK 2.1.0 iOS、Android、Windows、Web、Uniapp(小程序)、Electron 版本正式发布	RoomEngine SDK: 全平台： • 新增支持获取上麦请求列表。 • 修复偶现进房上麦后角色切换失败问题。 • 修复 room_info 的 create_time 精度丢失问题。 • 优化 OnDeviceChanged 回调事件优化 进房前/进房后均可使用。 • 优化桌面端默认横屏推流，移动端默认竖屏推流。 Electron/Web： • 优化 setLocalVideoView/setRemoteVideoView 支持传入 null。 接口变更： • 新增获取上麦请求列表接口 getSeatApplicationList。 • 新增信令被处理回调 OnRequestProcessed。 REST API变更： • 创建房间/获取房间信息 参数 RoomType: “Meeting” 修改为 “Conference” 。 RoomKit: • Android & iOS：修复上台申请被拒绝后，其他管理员上台管理列表不刷新问题。 • Android & iOS：修复被禁言用户成为房主后，房主自己无法发言的问题。 • iOS：修复大小屏切换时，偶现的黑屏问题。	2024.03.01

2024年2月

发布动态	更新日志	发布时间
多人音视频房间（TUIRoomEngine）SDK 2.0.3 iOS、Android、Windows、Web、Uniapp(小程序)、Electron 版本正式发布	全平台： • 修复大小流横竖屏分辨率不一致。 Flutter： • 修复 getUserList 接口 sequence 使用 int 存储溢出的问题。 Android： • 解决 TUIVideoView 内存泄露问题。	2024.02.04

2024年1月

发布动态	更新日志	发布时间
多人音视频房间（TUIRoomEngine）SDK 2.0.2 iOS、Android、Windows、Web、	全平台： • 邀请上麦信令被取消后对方无回调的问题修复。 • 申请上麦列表申请失效问题修复。 iOS&Mac： • TUIRoomVideoEncoderParams 符号问题修复。	2024.01.29

Uniapp(小程序)、Electron 版本正式发布	Web&Electron&小程序： • 渲染 view 偶现黑屏问题修复。	
多人音视频房间（TUIRoomEngine）SDK 2.0.1 Web、Uniapp(小程序)、Electron 版本正式发布	接口变更： Native 平台移除创建实例接口，推荐使用创建单例 sharedInstance，destroySharedInstance。 • Android 移除 createInstance、destroyInstance 创建实例接口。 • iOS 移除 init new 创建实例接口。 • Windows 移除使用 createTUIRoomEngine，destroyTUIRoomEngine 接口。 全平台： • 修复转让房主后角色异常问题。 • 复关闭麦克风后退房重新进房,麦克风还是关闭状态问题。 • 日志打印优化。 Web&Electron&小程序： • 解决 webpack5 及 vite3 项目引入 roomEngine 需要手动配置 node polyfill 的问题。 • 修复屏幕分享 targetId 失效问题。 • 修复 wasm 在小程序中报错问题，小程序环境下不支持 steady_clock 问题。	2024.01.19

多人音视频房间（TUIRoomEngine）SDK 2.0.0 Web、Uniapp(小程序)、Electron 版本正式发布	全平台升级新版房间管理，成员管理，媒体设备管理。优化麦位控制能力，聊天能力。
	• 麦位控制能力：
	  ○ 针对麦位控制模块进行接口调用和数据存储的优化，简化接口调用参数，存储麦位状态以便状态恢复。
	  ○ createRoom 接口支持传入 isSeatEnabled 及 seatMode，不再支持 speedMode 属性。
	  ○ 废弃 updateRoomSpeechModeByAdmin 接口。
	  ○ 废弃 onRoomSpeechModeChanged 事件。
	  ○ 新增 updateRoomSeatModeByAdmin 接口。
	  ○ 新增 onRoomSeatModeChanged 事件。
	  ○ 申请上台模式(即 isSeatEnabled 为 true，seatMode 为 kApplyToTake)，用户进房不再默认收到 onSeatListChanged 事件，仅当房间内用户有上下麦操作时会收到 onSeatListChanged 事件；您可以在进房后调用 getSeatList 接口获取所有麦位列表。
	  ○ 申请上台模式，onSeatListChanged 事件的 seatList 参数为当前房间所有麦位列表，没人在麦上的麦位信息里 userId 为 ''。
	• 聊天能力：
	  ○ 废弃 sendTextMessage 接口。
	  ○ 废弃 sendCustomMessage 接口。
	  ○ 废弃 onReceiveTextMessage 事件。
	  ○ 聊天模块实现需要使用 chat 实例实现。
		2024.01.19

多人音视频房间（TUIRoomEngine）SDK 2.0.0 iOS、Android、Windows 版本正式发布	全平台针对 Conference 场景，做出以下优化：
	• 房间管理：
	  ○ 独立的房间服务，不再依赖IM群组。
	  ○ 新增完整的房间列表管理 API。
	  ○ 新增预定会议相关的 REST API，完善 REST API 相关的能力。
	  ○ 底层 SDK 提供单例接口。
	  ○ 底层 SDK 更好的支持多实例。
		2024.01.05

	○ 优化进房流程，进房更稳定。 • 成员管理： ○ 独立的后台服务，更加精准的成员列表。 ○ 服务端 REST API 也支持成员管理。 ○ 与 IM 群组解绑，聊天群和会议房间不再耦合。 ○ 房间成员离线检测逻辑更新，对齐腾讯会议离线检测业务逻辑。 ○ 支持将非在线成员主动踢出房间，提供可以操作的服务端 REST API。 ○ 优化管理员&房主变更状态通知，修复1.x中关于角色反馈的若干问题。 • 媒体设备管理： ○ 独立的会控信令服务。 ○ 提供服务端操作相关的 API。 ○ 会控管理消息从 IM 群组的历史消息中剥离，不再污染 chat 中的聊天记录。 ○ 麦位（座位管理）独立的麦位服务，提供完整的麦位管理 REST API。 ○ 麦位统一业务逻辑，最多支持20个。 ○ 上麦支持排麦，上麦不在强制要求传入麦位 index。 ○ 有序麦位无序麦位统一逻辑，不在区分会议和直播。 ○ 支持随时拉取排麦（上麦请求）列表，解决1.x 版本中，中途退出后，之前的上麦申请会丢失的问题。	
多人音视频房间（TUIRoomEngine）SDK 1.7.2 全平台发布	全平台： • 修复入会音视频状态下发异常问题。 • 修复退房音视频状态偶现回调不一致问题。 • 大小流逻辑导致虚拟背景图模糊。 Electron： • setRemoteView 偶现失败问题修复。	2024.01.19
多人音视频房间（TUIRoomEngine）SDK 1.7.1 全平台发布	• UIKit 支持管理员设置交互。 • 修复若干 bug。	2024.01.05

2023年12月

发布动态	描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.7.0版本正式发布	更新日志： 全平台： • 修复转移房主后会出现状态异常的问题。 • 修复设置渲染view为空，资源不能释放的问题。 Web： • 解决 webpack5 及 vite3 项目引入 roomEngine 需要用户手动配置 node polyfill 的问题。 iOS： • 编译支持 framework 构建产物，支持 x64。 • 进一步美化 UI，对圆角、背景色、大小等进行了调整。 • 修复了一些特殊场景下的异常问题。 • 修复偶现的画面黑屏问题。 • 更改房间号的生成逻辑，修改偶现的创建房间失败问题。 Android： • 细化房间内演讲者小画面的刷新内容，将刷新拆分为 用户、视频、音频三部	2023.12.19

	分。 • 进一步美化设置面板UI样式，对面板圆角、背景色、大小等进行了统一的调整。 • 解决底部菜单栏偶现多次点击出现多个设置面板问题。 • 修复了一些特殊场景下的异常问题。 • 优化开发接入流程，适配了 Gradle 8.0。	

2023年11月

动态名称	动态描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.6.2版本正式发布	Electron： • 修复 Mac 下，开启摄像头成员超百人，订阅远端用户视频流，显示黑屏问题。 • 修复晚进房成员，拉取不到先进房成员聊天消息问题。	2023.11.24
多人音视频房间(TUIRoomEngine) SDK 1.6.1版本正式发布	全平台： • 修复视频渲染 view 缓存异常。 • 修复举手发言模式下台后本地屏幕分享状态异常。 • 修复账号切换偶现进房无回调问题。 • 废弃 `SetLocalVideoView` 带 `StreamType` 参数接口，新增不带 `StreamType` 的 `SetLocalVideoView` 接口。 Electron： • 修复Electron屏幕分享时被选中窗口不显示绿框的问题。 Web： • 登录优化。 Flutter： • RC 版本含 UI 方案 RoomKit 发布。	2023.11.10

2023年10月

发布动态	描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.6.0版本正式发布	更新日志： 全平台： • 修复大房间模式下，屏幕分享拉流问题。 • 修复大房间模式下，开启音视频设备后，退房杀进程后再次进入房间音视频状态异常问题。 • TUIError 新增 ERR_REPEAT_OPERATION，多次调用接口返回业务错误码。 Web： • 升级 chat js sdk 至 v3 版本。 iOS： • SDK 支持 Xcframework，支持模拟器运行。	2023.10.27

2023年09月

发布动态	描述	发布时间
多人音视频房间	更新日志：	2023.09.27

(TUIRoomEngine) SDK 1.5.1 版本发布	• 全平台： 修复 startPlayRemoteVideo 屏幕分享时填充模式画面异常问题。 • Web&Electron&小程序：修复本地用户网络回调（onNetworkQuality）信息不准的问题。 • Android：SDK 体积优化，体积相较上个版本缩减70%。 重点更新内容： • 会控音频管理接口优化，对齐主流方案，音频会控管理从开关麦克风切换为本地禁音。 • 新增暂停/恢复发布本地音频流接口：muteLocalAudio 、unmuteLocalAudio。 • 视频编码参数由 TUIVideoEncoderParams 修改为 TUIRoomVideoEncoderParam，解决符号冲突问题。 • 底层 SDK 支持二进制版本兼容，解除 IM SDK 版本绑定。	
多人音视频房间 (TUIRoomEngine) SDK 1.5.0 版本发布	更新日志： • 全平台：视频流策略优化，会议（Conference）场景下默认开启低清视频流。通过设置低清视频流编码参数，也可以开启低清视频流。 重点更新内容： • iOS&Android：底层 SDK 支持二进制版本兼容，解除 TRTC SDK 版本绑定。 • RoomEngine 支持小程序平台。 若干接口变动，详情见各平台文档说明： • 新增视频质量管理相关接口：UpdateVideoQualityEx、SetVideoResolutionMode、EnableGravitySensor（仅移动端）。 • 新增本地音频静音接口：muteLocalAudio。 • 新增摄像头操作接口：switchCamera（仅移动端）。 • 新增桌面端硬件设备管理接口：getDevicesList 、setCurrentDevice（仅桌面端）。 • 新增基础美颜设置接口：setBeautyLevel 、setWhitenessLevel。 • 新增回调 onDeviceChanged（桌面端）。 • 废弃若干接口，详情见各平台头文件。	2023.09.07

2023年08月

发布动态	描述	发布时间
多人音视频房间 (TUIRoomEngine) SDK 1.4.6 版本发布	• 全平台：TUIChat 组件消息过滤支持。 • 全平台：修复解析 IM 自定义消息导致的 crash。	2023.08.30
多人音视频房间 (TUIRoomEngine) SDK 1.4.5 版本发布	• 全平台：打开摄像头、麦克风 callback 回调多次问题修复。 • Android&iOS：结束屏幕分享回调优化。 • web：tsignal 升级。 • flutter：修复解析IM自定义消息导致的 crash。	2023.08.14
多人音视频房间 (TUIRoomEngine) SDK 1.4.4 版本发布	• 修复举手发言模式下麦位列表重复用户问题。 • 进房偶现失败问题修复。 • 修复直播场景下偶现拿不到用户信息问题。	2023.08.03

版权所有：腾讯云计算（北京）有限责任公司

	• 修复音视频状态回调时序异常问题。	

2023年07月

发布动态	描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.4.3 版本发布	• 修复 Conference 房间内，播放小流在移动端画面显示过大问题。 • 修复 LivingRoom 房间内，偶现拉取用户信息失败问题。 • 规范 SDK 接口注释，修复部分注释格式问题。	2023.07.28
多人音视频房间(TUIRoomEngine) SDK 1.4.2 版本发布	• 修复两个 iOS 同时屏幕共享会出现画面重叠的问题。 • 升级底层 electron sdk，解决 Mac 屏幕共享问题。 • 修复了频繁获取用户信息时，偶现Crash。 • 修复了部分场景下，关闭麦克风导致的音频音量回调异常问题。	2023.07.21
多人音视频房间(TUIRoomEngine) SDK 1.4.1 版本发布	• 支持房间人数过多时，自动开启低清视频流双路编码。 • 优化权限判断，支持房间内同级身份的用户之间的邀请开启设备、上麦等操作。 • 修复直播房间，偶现邀人上麦回调类型错乱问题。 • 修复打开麦克风偶现失败的问题。 • 修复多线程环境下，偶现调用 `RoomEngine` 出现崩溃的问题。 • 修复 `iOS&Mac` 平台下，偶现视频View释放导致程序异常问题。 • 踢人回调 `onKickedOutOfRoom` 增加 `reason` 标识，标明被移出原因。 • `RoomEngine Flutter` 版本发布。	2023.07.14

2023年5月

发布动态	描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.3.0版本正式发布	• 支持房间内管理员接收成员打开麦克风摄像头、上麦的申请。 • 统一全平台错误码，细化关键业务的错误码类型。 • `setSelfInfo` 接口支持用户设置自定义信息。 • 优化后台服务校验，增强创建房间，解散房间，房间状态的可靠性。 • 新增用户离线移出房间功能，减少僵尸用户。 • 新增服务端 REST API 支持。	2023.05.25

2023年4月

发布动态	描述	发布时间
多人音视频房间(TUIRoomEngine) SDK 1.2.0版本正式发布	多人音视频房间 SDK 是音视频终端 SDK（腾讯云视立方）的子产品之一，基于企业培训、在线会议、网络研讨会等多人音视频会话场景定制，提供房间管理、成员管理、屏幕分享等会控功能，提供含 UI 快速集成方案，仅需三步一天内可集成上线，轻松搭建专属音视频房间。 更新详情： • 更新优化了房间相关接口，提供会议模式房间类型。 • 拆分房间信息更新流程，独立房间信息字段更新相关接口。 • 优化了房间内麦控模式控制逻辑，提供丰富的麦控管理的相关接口。 • 细化成员管理相关逻辑，提供丰富的成员管理相关接口。	2023.04.18

版权所有：腾讯云计算（北京）有限责任公司

说明：

多人音视频房间 SDK（TUIRoomKit）于2023年4月8日起，正式上线。限时免费版同期结束限时免费体验，如果您在限免版发布期间接入了本产品，我们已经为您延长了一个月的限时体验时间到2023年5月8日，方便您全平台接入升级正式版，在此期间，您依旧可以使用体验本产品。

详细信息参考：产品公告 。

常见问题
iOS

最近更新时间：2023-04-18 19:41:34

我需要自己修改 UI，每次更新 pod 后就会刷新源码，导致修改丢失，该如何处理?

建议Fork TUIRoomKit 仓库 到您个人github账号下，然后采用 本地 pod 或者 git pod 路径 的方式在您的工程引用对应代码。
详情参考Pod 官网文档：

Using the files from a folder local to the machine.

If you would like to develop a Pod in tandem with its client project you can use `:path`.

```
pod 'Alamofire', :path => '~/Documents/Alamofire'
```

Using this option CocoaPods will assume the given folder to be the root of the Pod and will link the files directly from there in the Pods project. This means that your edits will persist between CocoaPods installations. The referenced folder can be a checkout of your favourite SCM or even a git submodule of the current repo.

Note that the `podspec` of the Pod file is expected to be in that the designated folder.

TUIRoomKit 和自己集成的音视频库冲突了?

腾讯云的 音视频库 不能同时集成，可能存在符号冲突，可以按照下面的场景处理。
1. 如果您使用了 TXLiteAVSDK_TRTC 库，不会发生符号冲突。可直接在 Podfile 文件中添加依赖，

```
pod 'TUIRoomKit'
```

2. 如果您使用了 TXLiteAVSDK_Professional 库，会产生符号冲突。您可在 Podfile 文件中添加依赖，

```
pod 'TUIRoomKit/Professional'
```

3. 如果您使用了 TXLiteAVSDK_Enterprise 库，会产生符号冲突。建议升级到 TXLiteAVSDK_Professional 后使用 TUIRoomKit/Professional 。

如何查看 TRTC 日志?

TRTC 的日志默认压缩加密，后缀为 .xlog。日志是否加密是可以通过 setLogCompressEnabled 来控制，生成的文件名里面含 C(compressed) 的就是加密压缩的，含 R(raw) 的就是明文的。

- iOS&Mac：sandbox的Documents/log
- Android：
  - 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/
  - 8.5之后的版本：/sdcard/Android/data/包名/files/log/liteav/
- Windows：
  - 8.8之前的版本：%appdata%/tencent/liteav/log
  - 8.8以及8.8之后的版本：%appdata%/liteav/log
- Web：打开浏览器控制台，或使用 vConsole 记录 SDK 打印信息

版权所有：腾讯云计算（北京）有限责任公司

说明：

- 查看 .xlog 文件需要下载 解密工具，在 Python 2.7环境中放到 xlog 文件同目录下直接使用 `python decode_mars_log_file.py` 运行即可。
- 查看 .clog 文件（9.6 版本以后新的日志格式）需要下载 解密工具，在 Python 2.7 环境中放到 clog 文件同目录下直接使用 `python decompress_clog.py` 运行即可。
- 更多日志相关设置参考： 日志输出配置 。

Android

最近更新时间：2023-05-29 17:06:25

TUIRoomKit 对依赖的 TRTC 和 IM SDK 有什么要求吗?

TUIRoomEngine在登录时会进行版本校验，提示您使用正确的 `TRTC SDK` 与 `IM SDK` 版本

```
W [W][05-25/20:11:34.781+8.0][7993,7993][global_service.cc:239]Tuikit: warning: im recommend to use 7.2.4146 or higher. And local use: 7.0.3768
W [W][05-25/20:11:34.781+8.0][7993,7993][global_service.cc:248]Tuikit: warning: trtc recommend to use 11.1.0.13144 or higher. And local use: 11.1.0.13141
```

TUIRoomKit 可以去掉 IM SDK 依赖吗?

不可以，多人音视频房间需要 IM 去做信令相关的互通，包括麦控，弹幕等功能，需要有 IM SDK 的依赖。

TUIRoomKit 是否支持会议预定等功能?

目前尚不支持，该功能在规划中，您可以关注我们的更新日志来第一时间获取该功能的上线情况。

集成了 TUIRoomKit 后，如何快速修改用户名和用户头像?

通过调用 `TUIRoomKit` 的 `setSelfInfo` 接口设置。

Web

最近更新时间：2023-12-20 10:37:12

环境相关问题

TUIRoomKit Web 支持的平台有哪些?

TUIRoomEngine Web 支持的平台请参考： TRTC Web SDK 对浏览器的支持情况 。
对于上述没有列出的环境，您可以在当前浏览器打开 TRTC 能力测试 测试是否完整的支持 WebRTC 的功能。

为什么本地开发测试能正常使用 TUIRoomKit, 但是部署到线上无法使用?

出于对用户安全、隐私等问题的考虑，浏览器限制网页只有在安全的环境下（例如 `https` 、 `localhost` 、 `file://` 等协议），才能采集麦克风、摄像头。HTTP 协议是不安全的，浏览器会禁止在 HTTP 协议下采集媒体设备。
若您在本地开发测试一切正常，但是页面部署后，却无法正常采集摄像头、麦克风。则请检查您的网页是否部署到了 HTTP 协议上，若是，请使用 HTTPS 部署您的网页，并确保具备合格的 HTTPS 安全证书。
更多详情请参见 URL域名及协议限制说明 。

TUIRoomKit Web 端是否支持使用 iframe 集成?

支持的。在 iframe 中集成 TUIRoom Web 需要给 iframe 标签增加属性来开启相关权限，具体参考如下。

```
// 开启麦克风、摄像头、屏幕分享、全屏权限
<iframe allow="microphone; camera; display-capture; display; fullscreen;">
```

编译相关问题

Webpack 5 引入 TUIRoomEngine sdk 报错：webpack < 5 used to include polyfills for node.js core modules by default. This is no longer the case. Verify if you need this module and configure a polyfill for it.

升级 @tencentcloud/tuiroom-engine-js npm 包到 v1.7.0 版本以上或者在 vue.config.js 中添加 configureWebpack 配置解决该问题。

```
module.exports = defineConfig({
  // ...
  configureWebpack: (config) => {
    config.resolve.fallback = {
      ...config.resolve.fallback,
      url: false,
      path: false,
      fs: false,
      crypto: false,
    };
  }
});
```

功能相关问题

项目中可以同时引入 tim-js-sdk 和 @tencentcloud/tuiroom-engine-js 吗?

可以的。您可以通过 tim-js-sdk 中的 TIM 创建 tim 实例对象，在 TUIRoomEngine 初始化时传入 init 接口。

```
await TUIRoomEngine.init({
 sdkAppId: 0,   // 填写您申请的 sdkAppId
 userId: '',    // 填写您业务对应的 userId
 userSig: '',   // 填写服务器或本地计算的 userSig
 tim,    // 传入 tim 实例
});
```

同时，您也可以通过 roomEngine.getTIM 方法获取 TUIRoomEngine 内部使用的 tim 实例。

版权所有：腾讯云计算（北京）有限责任公司

小程序

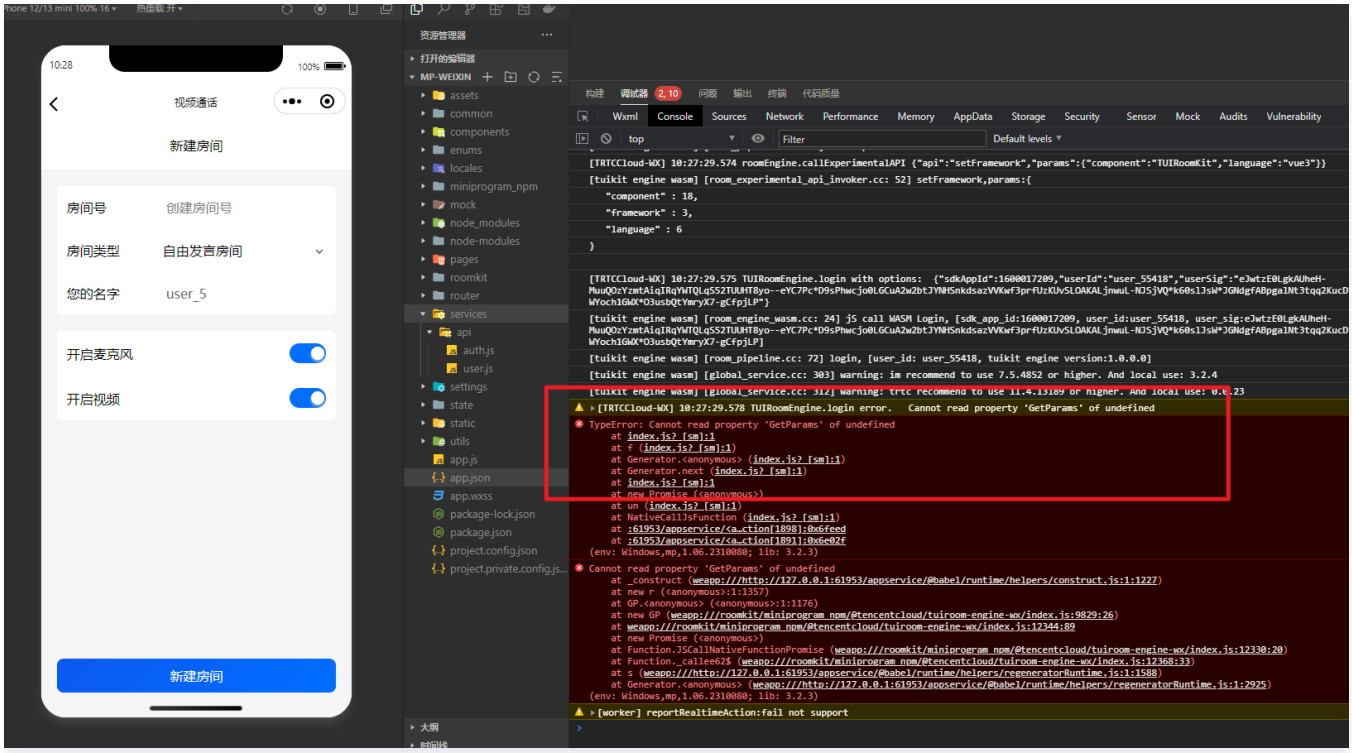
最近更新时间：2024-02-01 21:16:22

Windows 执行 ./wxmini_dev.bat 脚本报错?

报错原因可能是命令行窗口未集成 bash 脚本，可以使用 windows 的 power shell 执行该指令，或者使用 git bash 执行 ./wxmini_dev.bat。

开发者工具报错 Cannot read property 'GetParams' of undefined?

版本升级后出现 Cannot read property 'GetParams' of undefined 错误原因是 TUIRoomEngine.wasm.br 文件与当前版本不匹配的原因，需要手动到项目更目录复制 `./node_modules/@tencentcloud/tuiroom-engine-wx/npm-package/TUIRoomEngine.wasm.br` 文件对 `/static/TUIRoomEngine.wasm.br` 文件进行替换。



Flutter

最近更新时间：2023-11-17 11:00:42

iOS上运行时报错：Error (Xcode): Library not found for -ld64?

`Xcode` 15 换了编译的 `linker`， `RoomEngine Flutter` 在1.6.0做了适配，老版本的 xcode 无法编译，使用最新的 `Xcode` 即可.

工程同时集成了 tencent_calls_uikit 和 rtc_room_engine ，出现加入 room 的房间会打断 call 的通话等两个组件互相干扰的问题，如何解决?

这是由于 `RTC` 业务的独占性导致的，我们推荐您使用两个标志位 `callState` 、 `roomState` 分别用来记录当前是否在通话中，是否在会议中，通过这两个标志位来实现二者业务的隔离，来避免此问题的发生。例如，如果当前正在通话中，则禁止进入房间内。

版权所有：腾讯云计算（北京）有限责任公司

2.0版本升级指引

最近更新时间：2024-01-24 10:23:11

多人音视频（TUIRoomKit）2.0版本更新属于重大升级，全方位提升 SDK 能力，调整 SDK 服务端架构，新增全功能服务端 REST API。发布时间、1.x和2.0版本功能对比、变更影响等更多内容详见文档：多人音视频（TUIRoomKit）发布2.0版本 SDK 和推荐升级公告。

注意：

- 本文主要帮助已接入1.x版本 SDK 的客户更快的升级至2.0版本，体验更强大的服务和功能。
- 如果您已付费购买多人音视频（TUIRoomKit），为了避免您的线上业务出现问题，请尽快参照下述步骤完成升级。
- 如果您之前没有接入过1.x版本的SDK，或者您的应用目前仅在开发阶段，尚未上线，我们推荐您直接升级并使用2.0版本，功能更强大，更稳定。

升级步骤

1、关注事项

- UIKit 已自适配2.0逻辑，您无需关心 UIKit 层面的修改。
- 部分接口变更：
  - RoomInfo 废弃了 SpeechModel，使用 enableSeat 和 seatMode 来替代。
  - 废弃了 sendMessage、sendCustomMessage 接口。
- 服务端接口变更：
  - 提供的 V2 版本的 REST API，全面覆盖房间管理，用户管理，会控管理等功能。
- 更多功能变更详情参见 功能说明 。

2、获取最新版本 SDK

您只需下载升级最新的 RoomEngine SDK，即可升级到2.0版本。各开发平台的升级参考代码如下：

Android

将 RoomEngine Maven 版本依赖修改如下：

```
implement "com.tencent.liteav:LiteAVSDK_TRTC:11.5.0.13312"
implement "com.tencent.liteav.tuikit:tuiroomengine:2.0.0.100"
implement "com.tencent.imsdk:imsdk-plus:7.7.5282"
```

iOS

在 Podfile 中，将依赖修改如下：

说明：
RoomKit 和 RoomEngine 不必同时集成，您之前使用的是哪个，升级哪个即可，如果您之前修改过 Roomkit 的源码，现在仅需要升级RoomEngine，请在腾讯云实时音视频 TRTC 技术交流群平台联系我们，我们将给您提供帮助。

- 使用RoomKit：

```
pod 'TUIRoomKit', '~> 2.0.0'
```

- 使用RoomEngine：

```
pod 'TUIRoomEngine', '~> 2.0.0'
```

Web

在您的项目根目录下，命令行中执行：

```
npm install @tencentcloud/tuiroom-engine-js@2
```

Flutter

在 pubspec.yaml 文件中修改依赖的版本号，如下：

> **说明：**
> 版本号按需修改，之前依赖了哪个 package，修改对应的版本号即可。

```
dependencies:
  # RoomKit 依赖
  rtc_conference_tui_kit: ^2.0.0
  # RoomEngine 依赖
  rtc_room_engine: ^2.0.0
```

在项目的根目录下，在命令行执行：

```
flutter pub get
```

Electron

在您的项目根目录下，命令行中执行：

```
npm install @tencentcloud/tuiroom-engine-electron@2
```

小程序

在您的项目根目录下，命令行中执行：

警告：

版本升级后出现 Cannot read property 'GetParams' of undefined 错误原因是 TUIRoomEngine.wasm.br 文件与当前版本不匹配的原因，需要手动复制 `./node_modules/@tencentcloud/tuiroom-engine-wx/npm-package/TUIRoomEngine.wasm.br` 进行替换。

```
# RoomKit升级
npm install @tencentcloud/tui-roomkit-uniapp@2
# RoomEngine升级
npm install @tencentcloud/tuiroom-engine-wx@2
```

注意：

- 如果您之前没有接入过1.x版本的 SDK，可直接前往 SDK 下载 文档下载最新版本 SDK。
- 如果您参考上述代码未成功升级，请在 腾讯云实时音视频 TRTC 技术交流群 联系我们，我们将为您提供帮助。

3、线上升级

线上推送新版本，灰度100%，完成线上升级，确保您的客户都升级到已包含2.0版本 SDK 的应用。

注意：

因服务端架构重大升级，1.x和2.0版本的房间无法直接互通，请务必完成线上升级，避免业务混用新旧版本 SDK。

4、开通2.0后台服务

如果您是**2024年1月3日正式发布之前已付费的多人音视频（TUIRoomKit）客户**，在完成前3个步骤后，请加入 – 腾讯云实时音视频 TRTC 技术交流群，联系我们开通2.0后台服务，点击立即加入 。

错误码

最近更新时间：2023-08-01 17:04:31

客户端错误码

通用错误码

错误码	描述
0	操作成功
-1	暂未归类的通用错误
-2	请求被限频，请稍后重试
-1000	未找到 SDKAppID，请在 腾讯云视立方 SDK 控制台 确认应用信息
-1001	调用 API 时，传入的参数不合法，检查入参是否合法
-1002	未登录,请调用 Login 接口
-1003	获取权限失败，当前未授权音/视频权限，请查看是否开启设备权限
-1004	该功能需要开通额外的套餐，请在 腾讯云视立方 SDK 控制台 按需开通对应套餐

本地用户渲染，视频管理，音频管理 API 回调错误定义

错误码	描述
-1100	系统问题，打开摄像头失败。检查摄像头设备是否正常
-1101	摄像头没有系统授权，检查系统授权
-1102	摄像头被占用，检查是否有其他进程使用摄像头
-1103	当前无摄像头设备，请插入摄像头设备解决该问题
-1104	系统问题，打开麦克风失败。检查麦克风设备是否正常
-1105	麦克风没有系统授权，检查系统授权
-1106	麦克风被占用
-1107	当前无麦克风设备
-1108	获取屏幕分享对象失败，检查屏幕录制权限
-1109	开启屏幕分享失败，检查房间内是否有人正在屏幕分享

房间管理相关 API 回调错误定义

错误码	描述
-2100	进房时房间不存在，或许已被解散
-2101	需要进房后才可使用此功能

-2102	房主不支持退房操作，Conference（会议）房间类型: 可以先转让房主，再退房。LivingRoom（直播）房间类型: 房主只能解散房间
-2103	当前房间类型下不支持该操作
-2104	当前发言模式下不支持该操作
-2105	创建房间 ID 非法，自定义 ID 必须为可打印 ASCII 字符（0x20-0x7e），最长48个字节
-2106	房间 ID 已被使用，请选择别的房间 ID
-2107	房间名称非法，名称最长30字节，字符编码必须是 UTF-8 ，如果包含中文
-2108	当前用户已在别的房间内，单个 roomEngine 实例只支持用户进入一个房间，如果要进入不同的房间请先退房或者使用新的 roomEngine 实例

房间内用户信息 API 回调错误定义

错误码	描述
-2200	未找到该用户
-2201	房间内未找到该用户

房间内用户发言管理 API 回调错误定义&房间内麦位管理 API 回调错误定义

错误码	描述
-2300	需要房主权限才能操作
-2301	需要房主或者管理员权限才能操作
-2310	信令请求无权限，比如取消非自己发起的邀请。
-2311	信令请求 ID 无效或已经被处理过。
-2340	最大麦位超出套餐包数量限制
-2341	当前用户已经在麦位上
-2342	当前麦位已经有人了
-2343	当前麦位被锁
-2344	麦位编号不存在
-2345	当前用户没有在麦上
-2346	上麦人数已满
-2360	当前麦位音频被锁
-2361	需要向房主或管理员申请后打开麦克风
-2370	当前麦位视频被锁，需要由房主解锁麦位后，才能打开摄像头
-2371	需要向房主或管理员申请后打开摄像头
-2380	当前房间已开启全员禁言

版权所有：腾讯云计算（北京）有限责任公司

-2381	当前房间内，您已被已禁言

服务端错误码

错误码	描述
83007	请求被限频
84002	房间不存在
84003	房间内人数已超出最大套餐包限制
84004	房间不存在该用户
84005	房间已存在
84006	房主不能退房
85001	麦位已满
85002	麦位被锁定
85003	用户已上麦
85004	麦位被使用
85005	麦位号不存在
85006	麦位音频被锁定
85007	麦位视频被锁定
85008	非直播场景不能使用麦位
87001	不允许修改成员身份或房间信息
87002	权限不足，需要房主或管理员身份
87003	IM 群属性更新错误
87005	IM 群属性获取异常
87006	权限不足，需要房主才可以操作
87007	权限不足，需要管理员才可以操作
87008	不允许销毁 IM 房间
41001	创建 IM 群组错误
41002	销毁 IM 群组错误
41003	添加 IM 成员错误
41004	删除 IM 成员错误
41005	IM 改变角色错误
42001	预定会议错误

版权所有：腾讯云计算（北京）有限责任公司

42002	预定会议无效
42003	预定会议必须是会议场景
42004	预定会议未开始
42005	预定会议已经开始
42006	预定会议邀请成员太多，超出数量限制

版权所有：腾讯云计算（北京）有限责任公司

语音互动直播
集成 TUIVoiceRoom (Android)

最近更新时间：2023-05-08 10:02:03

组件介绍

TUIVoiceRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIVoiceRoom 组件，您只需要编写几行代码就可以为您的 App 添加“多人语音聊天”等场景，TUIVoiceRoom 同时支持 iOS 、小程序 等平台，基本功能如下图所示：

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 `Android/debug` 和 `Android/tuivoiceroom` 目录到您的工程中，并完成如下导入动作：

- 在 `setting.gradle` 中完成导入，参考如下：

```
include ':debug'
include ':tuivoiceroom'
```

- 在 app 的 build.gradle 文件中添加对 tuivoiceroom 的依赖：

```
api project(':tuivoiceroom')
```

- 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
  liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
  imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请麦克风权限等）：

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

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

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

步骤三：初始化并登录

```
// 1.初始化，
TRTCVoiceRoom mTRTCVoiceRoom = TRTCVoiceRoom.sharedInstance(this);
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
);
    // 2.登录，
mTRTCVoiceRoom.login(SDKAppID, userId, userSig, new TRTCVoiceRoomCallback.ActionCallback() {
   @Override
   public void onCallback(int code, String msg) {
      if (code == 0) {
      //登录成功
      }
   }
});
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 userSig，也可以参照我们的 示例工程 自行计算，更多信息见 如何计算及使用 UserSig 。

步骤四：实现语音聊天房间

1. **实现房主创建语音聊天房间** TRTCVoiceRoom#createRoom

```
// 1.房主调用创建房间
int roomId = 12345; //房间id
final TRTCVoiceRoomDef.RoomParam roomParam = new TRTCVoiceRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 7; // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.coverUrl = "房间封面图的 URL 地址";
mTRTCVoiceRoom.createRoom(roomId, roomParam, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
          //创建成功
    }
 }
});
```

2. **实现听众加入语音聊天房间** TRTCVoiceRoom#enterRoom

```
// 1.听众调用加入房间
mTRTCVoiceRoom.enterRoom(roomId, new TRTCVoiceRoomCallback.ActionCallback() {
    @Override
    public void onCallback(int code, String msg) {
       if (code == 0) {
       //进房成功
```

```
        }
    }
});
```

3. **实现听众主动上麦** TRTCVoiceRoom#enterSeat

```
// 1: 听众调用上麦
int seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
    //操作成功
    }
 }
});

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

4. **实现房主抱人上麦** TRTCVoiceRoom#pickSeat

```
// 1: 房主调用抱人麦位
int seatIndex = 2; //麦位的index
String userId = "123"; //需要上麦的用户id
mTRTCVoiceRoom.pickSeat(1, userId, new TRTCVoiceRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
        //操作成功
    }
 }
});

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
public void onSeatListChange(final List<TRTCVoiceRoomDef.SeatInfo> seatInfoList) {
}
```

5. **实现听众申请上麦** TRTCVoiceRoom#sendInvitation

```
// 听众端视角
// 1.听众调用申请上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("takeSeat", userId, seatIndex, null);

// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
```

```
    mTRTCVoiceRoom.enterSeat(index, null);
 }
}

// 房主端视角
// 1.房主收到请求
 @Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String content) {
 if (cmd.equals("takeSeat")) {
    // 2.房主同意听众请求
     mTRTCVoiceRoom.acceptInvitation(id, null);
 }
}
```

6. **实现房主邀请上麦** TRTCVoiceRoom#sendInvitation

```
// 房主端视角
// 1.房主调用请求抱人上麦
String seatIndex = "1"; //麦位的index
String userId = "123"; //用户id
String inviteId = mTRTCVoiceRoom.sendInvitation("pickSeat", userId, seatIndex, null);

// 2.收到邀请的同意请求，正式上麦
@Override
public void onInviteeAccepted(String id, String invitee) {
 if(id.equals(inviteId)) {
    mTRTCVoiceRoom.pickSeat(index, null);
 }
}

// 听众端视角
// 1.听众收到请求
 @Override
public void onReceiveNewInvitation(final String id, String inviter, String cmd, final String content) {
 if (cmd.equals("pickSeat")) {
    // 2.听众同意房主请求
     mTRTCVoiceRoom.acceptInvitation(id, null);
 }
}
```

7. **实现文字聊天** TRTCVoiceRoom#sendRoomTextMsg

```
// 发送端：发送文本消息
mTRTCVoiceRoom.sendRoomTextMsg("Hello Word!", null);
// 接收端：监听文本消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
  @Override
  public void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo) {
   Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
  }
});
```

8. **实现弹幕消息** TRTCVoiceRoom#sendRoomCustomMsg

```
// 发送端：您可以通过自定义 Cmd 来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_DANMU", "Hello world", null);
mTRTCVoiceRoom.sendRoomCustomMsg("CMD_LIKE", "", null);
// 接收端：监听自定义消息
mTRTCVoiceRoom.setDelegate(new TRTCVoiceRoomDelegate() {
 @Override
 public void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo userInfo) {
    if ("CMD_DANMU".equals(cmd)) {
        // 收到弹幕消息
        Log.d(TAG, "收到来自" + userInfo.userName + "的弹幕消息:" + message);
    } else if ("CMD_LIKE".equals(cmd)) {
        // 收到点赞消息
        Log.d(TAG, userInfo.userName + "给您点了个赞！");
    }
 }
});
```

交流与反馈

- 如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

集成 TUIVoiceRoom (iOS)

最近更新时间：2023-05-08 10:02:03

组件介绍

TUIVoiceRoom 是一个开源的音视频 UI 组件，通过在项目中集成 TUIVoiceRoom 组件，您只需要编写几行代码就可以为您的 App 添加“多人语音聊天”等场景。TUIVoiceRoom 同时支持 Android 、小程序 等平台，基本功能如下图所示：

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUIVoiceRoom 组件

在您的 xcode 工程 `Podfile` 文件同一级目录下创建 `TUIVoiceRoom` 文件夹，将 Github仓库 iOS 目录 下的 TXAppBasic 、Resources 、Source 、TUIVoiceRoom.podspec 等文件拷贝至您在自己工程创建的 `TUIVoiceRoom` 目录下。并完成如下导入动作：

- 打开工程的 Podfile 文件，引入 TUIVocieRoom.podspec，参考如下：

```
# path 为TXAppBasic.podspec相对于Podfile文件的相对路径
pod 'TXAppBasic', :path => "TUIVoiceRoom/TXAppBasic/"
# path 为TUIVoiceRoom.podspec相对于Podfile文件的相对路径
pod 'TUIVoiceRoom', :path => "TUIVoiceRoom/", :subspecs => ["TRTC"]
```

- 终端进入 Podfile 所在的目录下，执行 `pod install`，参考如下：

```
pod install
```

步骤二：配置权限及混淆规则

在 info.plist 文件中需要添加 `Privacy > Microphone Usage Description` 申请麦克风权限。

```
<key>NSMicrophoneUsageDescription</key>
<string>VoiceRoomApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```

步骤三：初始化并登录

Objective-C

```
#define SDKAppID 000000000

[[TRTCVoiceRoom sharedInstance] login:SDKAppID userId:@"userId" userSig:@"userSig" callback:^(int code,
NSString * _Nonnull message) {
  if (code == 0) {
    NSLog(@"初始化成功");
  }
}];
```

Swift

```
// 初始化
let mTRTCVoiceRoom = TRTCVoiceRoom.shared()
// TRTC 应用ID
let SDKAppID = 0
// 登录
mTRTCVoiceRoom.login(sdkAppID: SDKAppID, userId: "userId", userSig: "userSig") { code, message in
   if code == 0 {
      //登录成功
   }
}
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 userSig，也可以参照我们的 示例工程 自行计算，更多信息见 如何计算及使用 UserSig 。

步骤四：实现语音聊天房间

1. **实现房主创建语音聊天房间** TRTCVoiceRoom#createRoom

Objective-C

```
VoiceRoomParam *roomParam = [[VoiceRoomParam alloc] init];
roomParam.roomName = @"房间名称";
// 听众上麦是否需要房主同意
roomParam.needRequest = NO;
roomParam.coverUrl = @"房间封面图的 URL 地址";
// 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.seatCount = 7;

// 初始化麦位信息
NSMutableArray *seatInfoArray = [NSMutableArray array];
for (NSInteger i = 0; i < roomParam.seatCount; i++) {
    VoiceRoomSeatInfo *seatInfo = [[VoiceRoomSeatInfo alloc] init];
    [seatInfoArray addObject:seatInfo];
}
roomParam.seatInfoList = seatInfoArray;

// 房主端创建房间
[[TRTCVoiceRoom sharedInstance] createRoom:0 roomParam:roomParam callback:^(int code, NSString *
_Nonnull message) {
    if (code == 0) {
      NSLog(@"创建房间成功");
    }
```

```
}];
```

Swift

```
// 初始化语聊房参数
let roomParam = VoiceRoomParam()
roomParam.roomName = "房间名称"
roomParam.needRequest = false // 听众上麦是否需要房主同意
roomParam.coverUrl = "房间封面图的 URL 地址"
roomParam.seatCount = 7 // 房间座位数，这里一共7个座位，房主占了一个后听众剩下6个座位
roomParam.seatInfoList = []

// 初始化麦位信息
for _ in 0..< param.seatCount {
 let seatInfo = VoiceRoomSeatInfo()
 param.seatInfoList.append(seatInfo)
}

// 房主端创建房间
TRTCVoiceRoom.shared().createRoom(roomID: yourRoomID, roomParam: roomParam) { (code, message) in
 if code == 0 {
    // 创建成功
 }
}
```

2. **实现听众加入语音聊天房间** TRTCVoiceRoom#enterRoom

Objective-C

```
// 1.听众调用加入房间
[[TRTCVoiceRoom sharedInstance] enterRoom:0 callback:^(int code, NSString * _Nonnull message) {
    if (code == 0) {
      NSLog(@"进入房间成功");
    }
}];
```

Swift

```
// 1.听众调用加入房间
mTRTCVoiceRoom.enterRoom(roomID: roomID) { (code, message) in
 // 进入房间结果回调
 if code == 0 {
    // 进房成功
 }
}
```

3. 实现听众主动上麦 TRTCVoiceRoom#enterSeat

Objective-C

```
// 1: 听众调用上麦
NSInteger seatIndex = 2; // 观众要上的麦位 index
[[TRTCVoiceRoom sharedInstance] enterSeat:seatIndex callback:^(int code, NSString * _Nonnull message) {
    if (code == 0) {
      NSLog(@"上麦成功");
    }
}];

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist {
    // 刷新的麦位列表
}
```

Swift

```
// 1: 听众调用上麦
let seatIndex = 2; //麦位的index
mTRTCVoiceRoom.enterSeat(seatIndex: seatIndex) { (code, message) in
 if code == 0 {
    // 上麦成功
 }
}

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
@Override
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
 // 刷新的麦位列表
}
```

4. 实现房主抱人上麦 TRTCVoiceRoom#pickSeat

Objective-C

```
// 1: 房主调用抱人麦位
NSInteger seatIndex = 2; // 观众要上的麦位 index
NSString *userId = @"123"; //需要上麦的用户id
[[TRTCVoiceRoom sharedInstance] pickSeat:2 userId:@"user id" callback:^(int code, NSString * _Nonnull
message) {
    if (code == 0) {
      NSLog(@"抱人上麦成功");
    }
}];

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist {
    // 刷新的麦位列表
```

```
}
```

Swift

```
// 1: 房主调用抱人麦位
let seatIndex = 2; //麦位的index
let userId = "123"; //需要上麦的用户id
mTRTCVoiceRoom.pickSeat(seatIndex: seatIndex, userId: "123") { (code, message) in
 if code == 0 {
 }
}

// 2.收到 onSeatListChange 回调，刷新您的麦位列表
func onSeatListChange(seatInfoList: [VoiceRoomSeatInfo]) {
 // 刷新的麦位列表
}
```

5. 实现听众申请上麦 TRTCVoiceRoom#sendInvitation

Objective-C

```
// 听众端视角
// 1.听众调用申请上麦
NSString *seatIndex = @"1"; //麦位的index
NSString *userId = @"123"; //用户id
NSString *inviteId = [[TRTCVoiceRoom sharedInstance] sendInvitation:@"takeSeat" userId:userId
content:seatIndex callback:^(int code, NSString * _Nonnull message) {

}];

// 2.收到邀请的同意请求，正式上麦
- (void)onInviteeAccepted:(NSString *)identifier
                  invitee:(NSString *)invitee {
    if ([identifier isEqualToString:inviteId]) {
      [[TRTCVoiceRoom sharedInstance] enterSeat:1 callback:^(int code, NSString * _Nonnull message) {

      }];
    }
}

// 房主端视角
// 1.房主收到请求
- (void)onReceiveNewInvitation:(NSString *)identifier
                       inviter:(NSString *)inviter
                           cmd:(NSString *)cmd
                       content:(NSString *)content {
    if ([cmd isEqualToString:@"takeSeat"]) {
      // 2.房主同意听众请求
      [[TRTCVoiceRoom sharedInstance] acceptInvitation:identifier callback:^(int code, NSString * _Nonnull
message) {

      }];
    }
```

```
}
```

Swift

```
// 听众端视角
// 1.听众调用申请上麦
let seatIndex = "1"; //麦位的index
let userId = "123"; //用户id
let inviteId = mTRTCVoiceRoom.sendInvitation(cmd: "takeSeat", userId: ownerUserId, content: "1") { (code,
message) in
 // 发送结果回调
}

// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
    self.mTRTCVoiceRoom.enterSeat(seatIndex: 1) { (code, message) in
       // 上麦结果回调
    }
 }
}

// 房主端视角
// 1.房主收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
 if cmd == "takeSeat" {
    // 2.房主同意听众请求
    self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
 }
}
```

6. 实现房主邀请上麦 TRTCVoiceRoom#sendInvitation

Objective-C

```
// 房主端视角
// 1.房主调用 sendInvitation，请求抱听众“123”上2号麦
NSString *inviteId = [[TRTCVoiceRoom sharedInstance] sendInvitation:@"pickSeat" userId:@"123"
content:@"2" callback:^(int code, NSString * _Nonnull message) {

}];

// 2.收到邀请的同意请求，正式上麦
- (void)onInviteeAccepted:(NSString *)identifier
                  invitee:(NSString *)invitee {
   if ([identifier isEqualToString:inviteId]) {
      [[TRTCVoiceRoom sharedInstance] pickSeat:2 userId:invitee callback:^(int code, NSString * _Nonnull
message) {

      }];
   }
}
```

```
// 听众端视角
// 1.听众收到请求
- (void)onReceiveNewInvitation:(NSString *)identifier
                       inviter:(NSString *)inviter
                           cmd:(NSString *)cmd
                       content:(NSString *)content {
    if ([cmd isEqualToString:@"pickSeat"]) {
      [[TRTCVoiceRoom sharedInstance] acceptInvitation:identifier callback:^(int code, NSString * _Nonnull
message) {

      }];
    }
}
```

Swift

```
// 房主端视角
// 1.房主调用 sendInvitation，请求抱听众“123”上2号麦
let inviteId = self.mTRTCVoiceRoom.sendInvitation(cmd: "pickSeat", userId: ownerUserId, content: "2") {
(code, message) in
 // 发送结果回调
}

// 2.收到邀请的同意请求，正式上麦
func onInviteeAccepted(identifier: String, invitee: String) {
 if identifier == selfID {
    self.mTRTCVoiceRoom.pickSeat(seatIndex:2, userId:invitee) { (code, message) in
       // 上麦结果回调
    }
 }
}

// 听众端视角
// 1.听众收到请求
func onReceiveNewInvitation(identifier: String, inviter: String, cmd: String, content: String) {
 if cmd == "pickSeat" {
    // 2.听众同意房主请求
    self.mTRTCVoiceRoom.acceptInvitation(identifier: identifier, callback: nil)
 }
}
```

7. 实现文字聊天 TRTCVoiceRoom#sendRoomTextMsg

Objective-C

```
// 发送端：发送文本消息
[[TRTCVoiceRoom sharedInstance] sendRoomTextMsg:@"hello world" callback:^(int code, NSString *
_Nonnull message) {

}];

// 接收端：监听文本消息
```

```
- (void)onRecvRoomTextMsg:(NSString *)message
                 userInfo:(VoiceRoomUserInfo *)userInfo {
    // 收到的message信息处理方法
}
```

Swift

```
// 发送端：发送文本消息
self.mTRTCVoiceRoom.sendRoomTextMsg(message: message) { (code, message) in

}

// 接收端：监听文本消息
func onRecvRoomTextMsg(message: String, userInfo: VoiceRoomUserInfo) {
 //收到的message信息处理方法
}
```

8. **实现弹幕消息** TRTCVoiceRoom#sendRoomCustomMsg

Objective-C

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
[[TRTCVoiceRoom sharedInstance] sendRoomCustomMsg:@"CMD_DANMU" message:@"hello world"
callback:^(int code, NSString * _Nonnull message) {

}];
[[TRTCVoiceRoom sharedInstance] sendRoomCustomMsg:@"CMD_LIKE" message:@"" callback:^(int code,
NSString * _Nonnull message) {

}];

- (void)onRecvRoomCustomMsg:(NSString *)cmd
                    message:(NSString *)message
                   userInfo:(VoiceRoomUserInfo *)userInfo {
    if ([cmd isEqualToString:@"CMD_DANMU"]) {
        // 接收到弹幕消息
    }
    if ([cmd isEqualToString:@"CMD_LIKE"]) {
        // 接收到点赞消息
    }
}
```

Swift

```
// 例如：发送端：您可以通过自定义Cmd来区分弹幕和点赞消息
// eg:"CMD_DANMU"表示弹幕消息，"CMD_LIKE"表示点赞消息
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: “CMD_DANMU”, message: "hello world", callback: nil)
self.mTRTCVoiceRoom.sendRoomCustomMsg(cmd: "CMD_LIKE", message: "", callback: nil)

// 接收端：监听自定义消息
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: VoiceRoomUserInfo) {
```

版权所有：腾讯云计算（北京）有限责任公司

```
    if cmd == "CMD_DANMU" {
        // 收到弹幕消息
    }
    if cmd == "CMD_LIKE" {
        // 收到点赞消息
    }
}
```

交流与反馈

- 如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

版权所有：腾讯云计算（北京）有限责任公司

集成 TUIVoiceRoom (小程序)

最近更新时间：2023-05-08 10:02:03

组件介绍

TUIVoiceRoom 组件是一个开源的场景化解决方案，帮助您快速在您的小程序中集成**语音聊天室**功能，内部集成语音聊天与文字聊天功能，非常适合用于多人语音会议，娱乐聊天室，语音交友平台等场景中。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

创建房间

由于用户不需要安装 App 就可以使用该功能，因此 TUIVoiceRoom 非常适合用于快速发起多人语音聊天室的场景。

房间管理

房间内部继承**低延迟、高质量**语音通话功能，创建者可对房间麦位进行锁定解锁并对用户进行禁言、禁画以及踢出操作。

其它平台

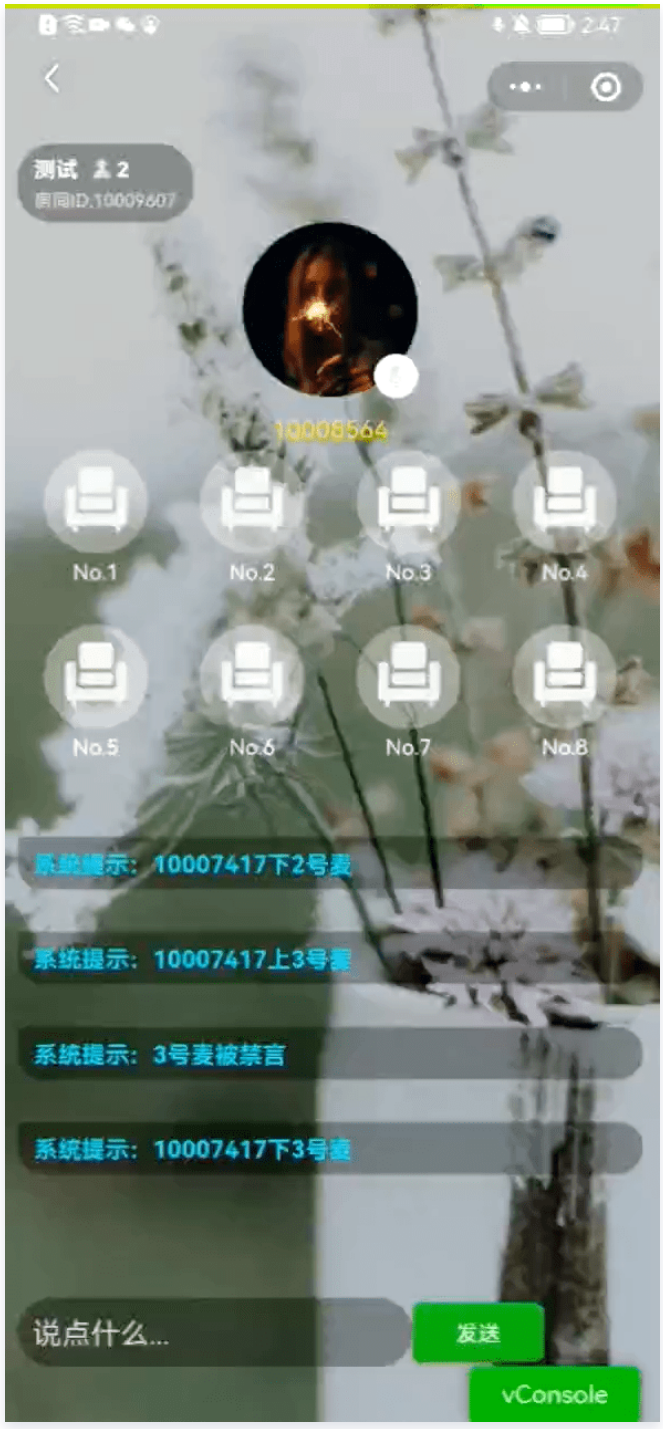
除了小程序版的 TUIVoiceRoom，我们同时也推出了 Android、iOS 等平台的源代码，用户可直接使用并对源码进行修改。

效果展示

您可以单击 Demo 体验 扫描二维码单击**语音聊天室**进行体验。语音聊天室包括麦位管理、低延时语音互动、文字聊天等 TRTC 在语音聊天场景下的相关能力。

房主麦位操作	听众麦位操作





如需快速接入语音聊天室功能，您可以直接基于我们提供的 Demo 进行修改适配。

环境要求

- 微信 App iOS 最低版本要求：7.0.9。
- 微信 App Android 最低版本要求：7.0.8。
- 小程序基础库最低版本要求：2.10.0。
- 由于微信开发者工具不支持原生组件（即 <live-pusher> 和 <live-player> 标签），需要在真机上进行运行体验。
- 由于小程序测试号不具备 <live-pusher> 和 <live-player> 的使用权限，需要申请常规小程序账号进行开发。
- 不支持 uniapp 开发环境，请使用原生小程序开发环境。

版权所有：腾讯云计算（北京）有限责任公司

前提条件

1. 您已 注册腾讯云 账号，并完成 实名认证 。
2. **开通小程序类目与推拉流标签权限（如不开通则无法正常使用）。**
   出于政策和合规的考虑，微信暂未放开所有小程序对实时音视频功能（即 <live-pusher> 和 <live-player> 标签）的支持：

- 小程序推拉流标签不支持个人小程序，只支持企业类小程序。
- 小程序推拉流标签使用权限暂时只开放给有限 类目 。
- 符合类目要求的小程序，需要在 微信公众平台 > **开发** > **开发管理** > **接口设置** 中自助开通该组件权限，如下图所示：



跑通 Demo

步骤一：开通小程序权限

由于 TUIVoiceRoom 所使用的小程序标签有更苛刻的权限要求，因此集成 TUIVoiceRoom 的第一步就是要开通小程序的类目和标签使用权限，**否则无法使用**，这包括如下步骤：

- 小程序推拉流标签不支持个人小程序，只支持企业类小程序。需要在 注册 时填写主体类型为企业，如下图所示：



- **小程序推拉流标签使用权限暂时只开放给有限 类目 。**
- **符合类目要求的小程序，需要在 微信公众平台 > 开发 > 开发管理 > 接口设置中自助开通该组件权限，如下图所示：**



步骤二：在小程序控制台配置域名

在 微信公众平台 > **开发** > **开发管理** > **开发设置** > **服务器域名**中设置 **request合法域名**和 **socket合法域名**，如下图所示：

- request 合法域名：

```
https://official.opensso.tencent-cloud.com
https://yun.tim.qq.com
https://cloud.tencent.com
https://webim.tim.qq.com
https://query.tencent-cloud.com
https://web.sdk.qcloud.com
```

- socket合法域名：

```
wss://wss.im.qcloud.com
wss://wss.tim.qq.com
```



步骤三：创建新的应用

1. 登录实时音视频控制台，选择**开发辅助** > 快速跑通Demo 。
2. 输入应用名称，例如 `TestVoiceRoom` ，单击**创建**。

> ⚠ **注意**
> 本功能同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 属于增值服务，详细计费规则请参见 即时通信 IM 价格说明 。

步骤四：下载 App 源码

单击进入 TUIVoiceRoom，Clone 或者下载源码。

步骤五：配置 Demo 工程文件

1. 进入修改配置页，根据您下载的源码包，选择相应的开发环境。
2. 找到并打开 `./debug/GenerateTestUserSig.js` 文件。
3. 设置 `GenerateTestUserSig.js` 文件中的相关参数：
   - SDKAPPID：默认为0，请设置为实际的 SDKAppID。
   - SECRETKEY：默认为空字符串，请设置为实际的密钥信息。

```
function genTestUserSig(userID) {
  /**
   * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
   *
   * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId，
   * 它是腾讯云用于区分客户的唯一标识。
   */
  var SDKAPPID    = 0;
  /**
   * 签名过期时间，建议不要设置的过短
   * <p>
   * 时间单位：秒
   * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
   */
  var EXPIRETIME = 604800;

  /**
   * 计算签名用的加密密钥，获取步骤如下：
   *
   * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一个，
   * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
   * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
   *
   * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄露导致的流量盗用。
   * 文档：https://cloud.tencent.com/document/product/647/17275#Server
   */
  var SECRETKEY = "";
```

将 SDKAppID 粘贴到此处

将 密钥 粘贴到此处

4. 粘贴完成后，单击**已复制粘贴，下一步**即创建成功。
5. 编译完成后，单击**回到控制台概览**即可。

> **注意**
>
> - 本文提到的生成 UserSig 的方案是在客户端代码中配置 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 。

步骤六：编译运行

1. 打开微信开发者工具，选择**小程序**，单击新建图标，选择**导入项目**。
2. 填写您微信小程序的 AppID，单击**导入**。



> **注意**
> 此处应输入您微信小程序的 AppID，而非 SDKAppID。

3. 单击**预览**，生成二维码，通过手机微信扫码二维码即可进入小程序。

技术咨询

了解更多详情您可 QQ 咨询：**941036374（技术交流群）**

参考文档

- 实时音视频 trtc-wx API 手册
- 即时通信 IM sdk API 手册
- 小程序 TUIVoiceRoom 源码
- 小程序 TUIRoom 源码
- 小程序如何支持 typescript + sass/less

TUIVoiceRoom API 查询 TRTCVoiceRoom (iOS)

最近更新时间：2023-09-07 17:07:41

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。

exitRoom	退出房间（听众调用）。
getUserInfoList	获取指定 userId 的用户信息，如果为 nil，则获取房间内所有人的信息。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位 (麦上主播端可调用)。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectMana	获取背景音乐音效管理对象 TXAudioEffectManager 。

ger	

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。

版权所有：腾讯云计算（北京）有限责任公司

onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
/**
* 获取 TRTCVoiceRoom 单例对象
*
* - returns: TRTCVoiceRoom 实例
* - note: 可以调用 {@link TRTCVoiceRoom#destroySharedInstance()} 销毁单例对象
*/
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCVoiceRoom 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCVoiceRoom 实例不能再使用，需要重新调用 {@link
TRTCVoiceRoom#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCVoiceRoom 事件回调，您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知。

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCVoiceRoom 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，可
使用 {@link TRTCVoiceRoom#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCVoiceRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

> **说明**
> setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送到主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
* - parameter queue 队列，TRTCVoiceRoom 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
       userId:(NSString *)userId
      userSig:(NSString *)userSig
     callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback:));
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在**实时音视频控制台 >**应用管理> 应用信息中查看 SDKAppID。
userId	NSString	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	NSString	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback
_Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	NSString	昵称。
avatarURL	NSString	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(VoiceRoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	VoiceRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	NSInteger	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录

上麦是否需要请求房主同意等。

4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(VoiceRoomUserListCallback _Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	NSArray<NSString>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> **说明**
> 上麦成功后，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

版权所有：腾讯云计算（北京）有限责任公司

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

moveSeat

移动麦位 (麦上主播端可调用)。

> **说明**
> 移动麦位成功后，房间内所有成员会收到 onSeatListChange 、 onAnchorLeaveSeat 和 onAnchorEnterSeat 的事件通知。(主播调用后，只是修改麦位座位号信息，并不会切换该用户的主播身份。)

```
- (NSInteger)moveSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(moveSeat(seatIndex:callback:))
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要移动到的麦位序号。
callback	ActionCallback	操作回调。

返回值：

返回值	类型	含义
code	NSInteger	移动麦位操作结果（0为成功，其它为失败，10001为接口调用限频）。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> **说明**
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> **说明**
> 房主抱人上麦，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(pickSeat(seatIndex:userId:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要抱上麦的麦位序号。
userId	NSString	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> **说明**
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(kickSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> **说明**
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(muteSeat(seatIndex:isMute:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。

isMute	BOOL	YES：静音对应麦位；NO：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> **说明**
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(closeSeat(seatIndex:isClose:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	NSInteger	需要操作的麦位序号。
isClose	BOOL	YES：封禁对应麦位； NO：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义

quality	NSInteger	音频质量，详情请参见 TRTC SDK 。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	静音/取消静音，详情请参见 TRTC SDK 。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
useSpeaker	BOOL	YES：扬声器；NO：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)voluem NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	采集音量，0 – 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	NSInteger	播放音量，0 – 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

版权所有：腾讯云计算（北京）有限责任公司

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	指定的用户 ID。
mute	BOOL	YES：开启静音；NO：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
mute	BOOL	YES：开启静音；NO：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enable:));
```

参数如下表所示：

参数	类型	含义
enable	BOOL	YES：开启耳返；NO：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	NSString	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback
_Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	NSString	命令字，由开发者自定义，主要用于区分不同消息类型。
message	NSString	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
                      userId:(NSString *)userId
                     content:(NSString *)content
                    callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(sendInvitation(cmd:userId:content:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	NSString	业务自定义指令。
userId	NSString	邀请的用户 ID。
content	NSString	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	NSString	用于标识此次邀请 ID。

版权所有：腾讯云计算（北京）有限责任公司

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(acceptInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(rejectInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(cancelInvitation(id:callback:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> **说明**

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。

```
- (void)onError:(int)code
        message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	NSString	错误信息。

onWarning

警告回调。

```
- (void)onWarning:(int)code
          message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	NSString	警告信息。

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	NSString	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)roomId
NS_SWIFT_NAME(onRoomDestroy(roomId:));
```

参数如下表所示：

参数	类型	含义
roomId	NSString	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(VoiceRoomInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
userId	NSString	用户 ID。
mute	BOOL	YES：静音麦位； NO：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:
(NSInteger)totalVolume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
roomInfo	VoiceRoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用muteLocalAudio，房间内的其他用户都会收到此通知。

```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userVolumes	NSArray	用户列表。
totalVolume	NSInteger	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

版权所有：腾讯云计算（北京）有限责任公司

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<VoiceRoomSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	NSArray<VoiceRoomSeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
                     user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	成员上麦的麦位。
user	VoiceRoomUserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
                     user:(VoiceRoomUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	下麦的麦位。
user	VoiceRoomUserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
            isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	操作的麦位。
isMute	BOOL	YES：静音麦位； NO：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
            isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

参数	类型	含义
index	NSInteger	操作的麦位。
isClose	BOOL	YES：封禁麦位； NO： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	VoiceRoomUserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

版权所有：腾讯云计算（北京）有限责任公司

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
                 userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)command
                    message:(NSString *)message
                   userInfo:(VoiceRoomUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(command:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
command	NSString	命令字，由开发者自定义，主要用于区分不同消息类型。
message	NSString	文本消息。
userInfo	VoiceRoomUserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
                       inviter:(NSString *)inviter
                           cmd:(NSString *)cmd
                       content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(id:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。
cmd	NSString	业务指定的命令字，由开发者自定义。

content	NSString	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
invitee	NSString	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
                      invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(id:invitee:));
```

参数如下表所示：

参数	类型	含义
id	NSString	邀请 ID。
inviter	NSString	邀请人的用户 ID。

TRTCVoiceRoom (Android)

最近更新时间：2023-09-07 17:07:42

TRTCVoiceRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的语音聊天室开播，听众进入语聊房间收听/互动。
- 房主可以邀请听众上麦、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，可以和其他人语音互动，也可以随时下麦成为普通的听众。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物等。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCVoiceRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 语音聊天室（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCVoiceRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

版权所有：腾讯云计算（北京）有限责任公司

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
moveSeat	移动麦位（麦上主播端可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。

版权所有：腾讯云计算（北京）有限责任公司

sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCVoiceRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述

版权所有：腾讯云计算（北京）有限责任公司

onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

SDK 基础函数

sharedInstance

获取 TRTCVoiceRoom 单例对象。

```
public static synchronized TRTCVoiceRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCVoiceRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCVoiceRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCVoiceRoom 事件回调，您可以通过 TRTCVoiceRoomDelegate 获得 TRTCVoiceRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCVoiceRoomDelegate delegate);
```

> **说明**
> setDelegate 是 TRTCVoiceRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCVoiceRoom 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
 String userId, String userSig,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在**实时音视频控制台 >**应用管理**>** 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig 。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCVoiceRoomDef.RoomParam roomParam,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的语音聊天室，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
public abstract void destroyRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的语音聊天室列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个语音聊天室，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCVoiceRoomCallback.UserListCallback
userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> **说明**
> 上麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
public abstract void enterSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

moveSeat

移动麦位 (麦上主播端可调用)。

> **说明**
> 移动麦位成功后，房间内所有成员会收到 onSeatListChange 、 onAnchorLeaveSeat 和 onAnchorEnterSeat 的事件通知。(主播调用后，只是修改麦位座位号信息，并不会切换该用户的主播身份。)

```
public abstract int moveSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要移动到的麦位序号。
callback	ActionCallback	操作回调。

返回值：

返回值	类型	含义
code	int	移动麦位操作结果（0为成功，其它为失败，10001 为接口调用限频）。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> **说明**
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void leaveSeat(TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

> **说明**
> 房主抱人上麦，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
public abstract void pickSeat(int seatIndex, String userId, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 sendInvitation 向听众申请，收到 onInvitationAccept 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> **说明**
> 房主踢人下麦，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void kickSeat(int seatIndex, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

版权所有：腾讯云计算（北京）有限责任公司

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> **说明**
> 静音/解除静音某个麦位，房间内所有成员会收到 onSeatListChange 和 onSeatMute 的事件通知。

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCVoiceRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> **说明**
> 房主封禁/解禁对应麦位，房间内所有成员会收到 onSeatListChange 和 onSeatClose 的事件通知。

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCVoiceRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK 。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK 。

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

版权所有：腾讯云计算（北京）有限责任公司

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content,
TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCVoiceRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

版权所有：腾讯云计算（北京）有限责任公司

TRTCVoiceRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCVoiceRoomDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用 muteLocalAudio，房间内的其他用户都会收到此通知。

```
void onUserMicrophoneMute(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	true：静音麦位； false：解除静音。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVolume);
```

参数如下表所示：

参数	类型	含义
userVolumes	ListList<TRTCCloudDef.TRTCVolumeInfo>	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCVoiceRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。

版权所有：腾讯云计算（北京）有限责任公司

isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义

版权所有：腾讯云计算（北京）有限责任公司

message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCVoiceRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

版权所有：腾讯云计算（北京）有限责任公司

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

视频互动直播
集成 TUILiveRoom (Android)

最近更新时间：2023-08-28 17:13:42

组件介绍

TUILiveRoom 是一个开源的视频直播 UI 组件，通过在项目中集成 TUILiveRoom 组件，您只需要编写几行代码就可以为您的 App 添加“视频互动直播”场景。TUILiveRoom 包含 Android、iOS 等平台的源代码，基本功能如下图所示：

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：下载并导入 TUILiveRoom 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 `Android/debug` ， `Android/tuiaudioeffect` ， `Android/tuibarrage` ， `Android/tuibeauty` ， `Android/tuigift` 和 `Android/tuiliveroom` 目录到您的工程中，并完成如下导入动作：

- 在 `setting.gradle` 中完成导入，参考如下：

```
include ':debug'
include ':tuibeauty'
include ':tuibarrage'
include ':tuiaudioeffect'
```

```
include ':tuigift'
include ':tuiliveroom'
```

- 在 app 的 build.gradle 文件中添加对 tuiliveroom 的依赖：

```
api project(":tuiliveroom")
```

- 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
  liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
  imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

1. 在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请相机、读取存储权限）：

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

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

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

步骤三：初始化并登录组件

```
// 1.添加事件监听及登录
TUILogin.addLoginListener(new TUILoginListener() {
    @Override
    public void onConnecting() {      // 正在连接中
        super.onConnecting();
    }
    @Override
    public void onConnectSuccess() {  // 连接成功通知
        super.onConnectSuccess();
    }
    @Override
    public void onConnectFailed(int errorCode, String errorMsg) {  // 连接失败通知
        super.onConnectFailed(errorCode, errorMsg);
    }
```

```
    @Override
    public void onKickedOffline() {  // 登录被踢下线通知（示例：账号在其他设备登录）
        super.onKickedOffline();
    }
    @Override
    public void onUserSigExpired() { // userSig过期通知
        super.onUserSigExpired();
    }
});
TUILogin.login(mContext, "Your SDKAppID", "Your userId", "Your userSig", new TUICallback() {
    @Override
    public void onSuccess() {
    }
    @Override
    public void onError(int errorCode, String errorMsg) {
        Log.d(TAG, "errorCode: " + errorCode + " errorMsg:" + errorMsg);
    }
});

// 2.初始化TUILiveRoom组件
TUILiveRoom mLiveRoom = TUILiveRoom.sharedInstance(mContext);
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **Secretkey：TRTC 应用密钥**和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。建议结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 userSig，也可以参照我们的 示例工程 自行计算，更多信息请参见 如何计算及使用 UserSig 。

步骤四：实现视频互动直播间

1. **主播端开播**

```
mLiveRoom.createRoom(int roomId, String roomName, String coverUrl);
```

2. **观众端观看**

```
mLiveRoom.enterRoom(roomId);
```

3. **观众与主播连麦** TRTCLiveRoom#requestJoinAnchor

```
// 1.观众端发起连麦请求
// LINK_MIC_TIMEOUT为超时时间
TRTCLiveRoom mTRTCLiveRoom=TRTCLiveRoom.sharedInstance(mContext);
mTRTCLiveRoom.requestJoinAnchor(mSelfUserId + "请求和您连麦", LINK_MIC_TIMEOUT
 new TRTCLiveRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
        // 主播接受了观众的请求
        TXCloudVideoView view = new TXCloudVideoView(context);
        parentView.add(view);
        // 观众启动预览，开启推流
        mTRTCLiveRoom.startCameraPreview(true, view, null);
        mTRTCLiveRoom.startPublish(mSelfUserId + "_stream", null);
    }
 }
});

// 2.主播端收到连麦请求
mTRTCLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
 @Override
 public void onRequestJoinAnchor(final TRTCLiveRoomDef.TRTCLiveUserInfo userInfo,
    String reason, final int timeout) {
    // 同意对方的连麦请求
    mTRTCLiveRoom.responseJoinAnchor(userInfo.userId, true, "同意连麦");
 }

   @Override
 public void onAnchorEnter(final String userId) {
    // 主播收到连麦观众的上麦通知
    TXCloudVideoView view = new TXCloudVideoView(context);
    parentView.add(view);
    // 主播播放观众画面
    mTRTCLiveRoom.startPlay(userId, view, null);
 }
});
```

4. **主播与主播 PK** TRTCLiveRoom#requestRoomPK

```
// 主播 A 创建12345的房间
mLiveRoom.createRoom(12345, "roomA", "Your coverUrl");
// 主播 B 创建54321的房间
mLiveRoom.createRoom(54321, "roomB", "Your coverUrl");
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 主播 A:
TRTCLiveRoom mTRTCLiveRoom=TRTCLiveRoom.sharedInstance(mContext);

// 1.主播 A 向主播 B 发起 PK 请求
mTRTCLiveRoom.requestRoomPK(54321, "B",
 new TRTCLiveRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    // 5.收到是否同意的回调
    if (code == 0) {
       // 用户接受
    } else {
       // 用户拒绝
    }
 }
});

mTRTCLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
 @Override
 public void onAnchorEnter(final String userId) {
    // 6.收到 B 进房的通知
    mTRTCLiveRoom.startPlay(userId, mTXCloudVideoView, null);
 }
});

// 主播 B:
// 2.主播 B 收到主播 A 的消息
mTRTCLiveRoom.setDelegate(new TRTCLiveRoomDelegate() {
 @Override
 public void onRequestRoomPK(
   final TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, final int timeout) {
    // 3.主播 B 回复主播 A 接受请求
    mTRTCLiveRoom.responseRoomPK(userInfo.userId, true, "");
 }
 @Override
 public void onAnchorEnter(final String userId) {
    // 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
    mTRTCLiveRoom.startPlay(userId, mTXCloudVideoView, null);
 }
});
```

步骤五：美颜特效（可选）

TUILiveRoom 美颜使用了 腾讯特效 SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License 申请指引 。

```
TUIBeautyView.getBeautyService().setLicense(context, “XMagicLicenseURL”, “XMagicLicenseKey”);
```

交流与反馈

- 如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

集成 TUILiveRoom (iOS)

最近更新时间：2023-11-15 20:32:14

组件介绍

TUILiveRoom 是一个开源的视频直播 UI 组件，通过在项目中集成 TUILiveRoom 组件，您只需要编写几行代码就可以为您的 App 添加“视频互动直播”场景。TUILiveRoom包含 Android、iOS 等平台的源代码，基本功能如下图所示：

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。



组件集成

步骤一：导入 TUILiveRoom 组件

通过 cocoapods 导入组件，具体步骤如下：

1. 在您的工程 Podfile 文件同一级目录下创建 TUILiveRoom 文件夹。
2. 单击进入 Github/TUILiveRoom ，选择克隆/下载代码，然后将 TUILiveRoom/iOS/ 目录下的 Source 、Resources 、TUIBeauty 、TUIAudioEffect 、TUIBarrage 、TUIGift 、TUIKitCommon 文件夹 和 TUILiveRoom.podspec 文件拷贝到您在 步骤1 创建的 TUILiveRoom 文件夹下。
3. 在您的 Podfile 文件中添加以下依赖，之后执行 pod install 命令，完成导入。

```
# :path => "指向TUILiveRoom.podspec的相对路径"
pod 'TUILiveRoom', :path => "./TUILiveRoom/TUILiveRoom.podspec", :subspecs => ["TRTC"]
```

```
# :path => "指向TUIKitCommon.podspec的相对路径"
pod 'TUIKitCommon', :path => "./TUILiveRoom/TUIKitCommon/"
# :path => "指向TUIBeauty.podspec的相对路径"
pod 'TUIBeauty', :path => "./TUILiveRoom/TUIBeauty/"
# :path => "指向TUIAudioEffect.podspec的相对路径"
pod 'TUIAudioEffect', :path => "./TUILiveRoom/TUIAudioEffect/", :subspecs => ["TRTC"]
# :path => "指向TUIBarrage.podspec的相对路径"
pod 'TUIBarrage', :path => "./TUILiveRoom/TUIBarrage/"
# :path => "指向TUIGift.podspec的相对路径"
pod 'TUIGift', :path => "./TUILiveRoom/TUIGift/"
```

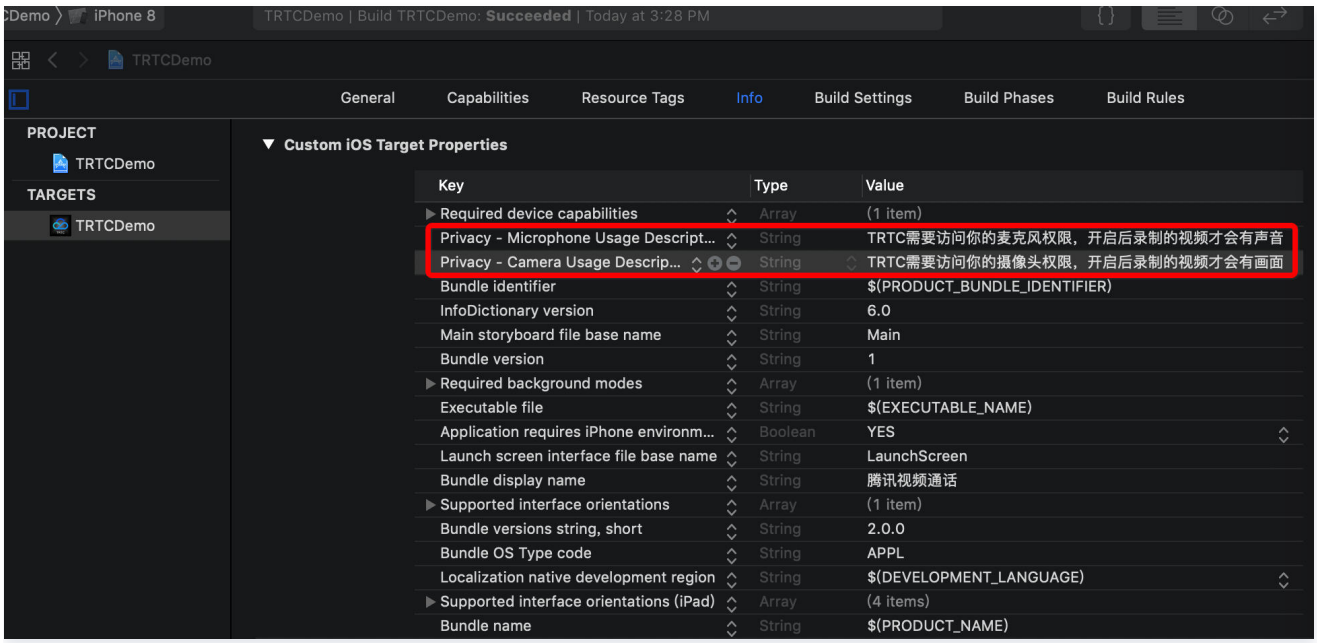
注意：

- `Source` 、 `Resources` 文件夹和 `TUILiveRoom.podspec` 文件必需在同一目录下。
- TUIKitCommon.podspec 在 TUIKitCommon 文件夹下。

步骤二：配置权限

使用音视频功能，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。

```
<key>NSCameraUsageDescription</key>
<string>RoomApp需要访问您的相机权限，开启后录制的视频才会有画面</string>
<key>NSMicrophoneUsageDescription</key>
<string>RoomApp需要访问您的麦克风权限，开启后录制的视频才会有声音</string>
```



步骤三：初始化并登录组件

Objective-C

```
@import TUILiveRoom;
```

```objc
@import TUICore;

// 1.组件登录
[TUILogin login:@"您的SDKAppID" userID:@"您的UserID" userSig:@"您的UserSig" succ:^{

} fail:^(int code, NSString *msg) {

}];
// 2.初始化TUILiveRoom实例
TUILiveRoom *mLiveRoom = [TUILiveRoom sharedInstance];
```

Swift

```swift
import TUILiveRoom
import TUICore

// 1.组件登录
TUILogin.login("您的SDKAppID", userID: "您的UserID", userSig: "您的UserSig") {

} fail: { code, msg in

}
// 2.初始化TUILiveRoom实例
let mLiveRoom = TUILiveRoom.sharedInstance
```

参数说明：

- **SDKAppID：TRTC 应用 ID**，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**，SDKAppID 信息如下图所示：



- **SecretKey：TRTC 应用密钥**，和 SDKAppID 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。

- **UserID**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **UserSig**：根据 SDKAppId、UserID，SecretKey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUILiveRoom 示例工程 自行计算，更多信息见 如何计算及使用 UserSig 。

步骤四：实现视频互动直播间

1. 主播端开播

Objective-C

```
[mLiveRoom createRoomWithRoomId:123 roomName:@"test room" coverUrl:@""];
```

Swift

```
mLiveRoom.createRoom(roomId: 123, roomName: "test room", coverUrl:"")
```

2. 观众端观看

Objective-C

```
[mLiveRoom enterRoomWithRoomId:123];
```

Swift

```
mLiveRoom.createRoom(roomId: 123)
```

3. 观众与主播连麦 TRTCLiveRoom#requestJoinAnchor

Objective-C

```
// 1.观众端发起连麦请求
[TRTCLiveRoom shareInstance].delegate = self;
// @param mSelfUserId String 当前用户id
NSString *mSelfUserId = @"1314";
[[TRTCLiveRoom shareInstance] requestJoinAnchor:[NSString stringWithFormat:@"%@ 请求和你连麦",
mSelfUserId] timeout:30 responseCallback:^(BOOL agreed, NSString * _Nullable reason) {
    if (agreed) {
        // 主播接受了观众的请求
      UIView *playView = [UIView new];
          [self.view addSubView:playView];
```

```
        // 观众启动预览，开启推流
        [[TRTCLiveRoom shareInstance] startCameraPreviewWithFrontCamera:YES view:playView callback:nil];
        [[TRTCLiveRoom shareInstance] startPublishWithStreamID:[NSString stringWithFormat:@"%@_stream",
mSelfUserId] callback:nil];
    }
}];

// 2.主播端收到连麦请求
#pragma mark - TRTCLiveRoomDelegate
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onRequestJoinAnchor:(TRTCLiveUserInfo *)user reason:
(NSString *)reason {
    // 同意对方的连麦请求
    [[TRTCLiveRoom shareInstance] responseJoinAnchor:user.userId agree:YES reason:@"同意连麦"];
}

- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onAnchorEnter:(NSString *)userID {
    // 主播收到连麦观众的上麦通知
    UIView *playView = [UIView new];
    [self.view addSubview:playView];
    // 主播播放观众画面
    [[TRTCLiveRoom shareInstance] startPlayWithUserID:userID view:playView callback:nil];
}
```

Swift

```
// 1.观众端发起连麦请求
TRTCLiveRoom.shareInstance().delegate = self
let mSelfUserId = "1314"
TRTCLiveRoom.shareInstance().requestJoinAnchor(reason: mSelfUserId + "请求和您连麦", timeout: 30) {
[weak self] (agree, msg) in
    guard let self = self else { return }
  if agree {
        // 主播接受了观众的请求
        let playView = UIView()
        self.view.addSubView(playView)
        // 观众启动预览，开启推流
        TRTCLiveRoom.shareInstance().startCameraPreview(frontCamera: true, view: playView)
        TRTCLiveRoom.shareInstance().startPublish(streamID: mSelfUserId + "_stream")
  }
}

// 2.主播端收到连麦请求
extension ViewController: TRTCLiveRoomDelegate {

    func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestJoinAnchor user: TRTCLiveUserInfo, reason:
String?) {
        // 同意对方的连麦请求
        TRTCLiveRoom.shareInstance().responseRoomPK(userID: user.userId, agree: true, reason: "同意连麦")
    }

    func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
        // 主播收到连麦观众的上麦通知
        let playView = UIView()
        view.addSubview(playView)
```

```
        // 主播播放观众画面
        TRTCLiveRoom.shareInstance().startPlay(userID: userID, view: playView);
    }
}
```

4. 主播与主播 PK TRTCLiveRoom#requestRoomPK

Objective-C

```
// 主播 A 创建12345的房间
[[TUILiveRoom sharedInstance] createRoomWithRoomId:12345 roomName:@"roomA" coverUrl:@"roomA
coverUrl"];
// 主播 B 创建54321的房间
[[TUILiveRoom sharedInstance] createRoomWithRoomId:54321 roomName:@"roomB" coverUrl:@"roomB
coverUrl"];

// 主播 A
// 主播 A 向 主播 B 发起PK请求
[[TRTCLiveRoom shareInstance] requestRoomPKWithRoomID:543321 userID:@"roomB userId" timeout:30
responseCallback:^(BOOL agreed, NSString * _Nullable reason) {
    if (agreed) {
        // 用户B接受
    } else {
        // 用户B拒绝
    }
}];

// 主播 B:
// 2.主播 B 收到主播 A 的消息
#pragma mark - TRTCLiveRoomDelegate
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onRequestRoomPK:(TRTCLiveUserInfo *)user {
    // 3.主播 B 回复主播 A 接受请求
    [[TRTCLiveRoom shareInstance] responseRoomPKWithUserID:user.userId agree:YES reason:@""];
}

- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom onAnchorEnter:(NSString *)userID {
    // 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
    [[TRTCLiveRoom shareInstance] startPlayWithUserID:userID view:playAView callback:nil];
}
```

Swift

```
// 主播 A 创建12345的房间
TUILiveRoom.sharedInstance.createRoom(roomId: 12345, roomName: "roomA")
// 主播 B 创建54321的房间
TUILiveRoom.sharedInstance.createRoom(roomId: 54321, roomName: "roomB")

// 主播 A
// 主播 A 向 主播 B 发起PK请求
```

```
TRTCLiveRoom.shareInstance().requestRoomPK(roomID: 543321, userID: "roomB userId", timeout: 30) {
[weak self] (agreed, msg) in
    guard let self = self else { return }
    if agreed {
        // 用户B接受
    } else {
        // 用户B拒绝
    }
}

// 主播 B：
// 2.主播 B 收到主播 A 的消息
extension ViewController: TRTCLiveRoomDelegate {
    func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onRequestRoomPK user: TRTCLiveUserInfo) {
        // 3.主播 B 回复主播 A 接受请求
        TRTCLiveRoom.shareInstance().responseRoomPK(userID: user.userId, agree: true, reason: "")
    }

    func trtcLiveRoom(_ trtcLiveRoom: TRTCLiveRoom, onAnchorEnter userID: String) {
        // 4.主播 B 收到主播 A 进房的通知，播放主播 A 的画面
        TRTCLiveRoom.shareInstance().startPlay(userID: userID, view: playAView);
    }
}
```

步骤五：美颜特效（可选）

TUILiveRoom 美颜使用了 腾讯特效 SDK，在使用美颜功能时，需要先设置 XMagic License，XMagic License 申请请参见 XMagic License 申请指引。

Objective-C

```
@import TUIBeauty;

- (void)setXMagicLicense {
    [[TUIBeautyView getBeautyService] setLicenseUrl:@"XMagicLicenseURL" key:@"XMagicLicenseKey"
completion:^(NSInteger authResult, NSString * _Nonnull errorMsg) {

    }];
}
```

Swift

```
import TUIBeauty

func setXMagicLicence() {
    // [Option] Tencent Effect: XMagic Beauty License
    TUIBeautyView.getBeautyService().setLicenseUrl(XMagicLicenseURL, key: XMagicLicenseKey) { code,
msg in
        debugPrint("auth result code:\(code) msg:\(msg)")
```

```
    }
}
```

```
```

交流与反馈

如果您是开发者，欢迎您加入我们的技术交流 QQ 群：**770645461**，进行技术交流和产品沟通。

集成 TUIPusher&TUIPlayer (Web)

最近更新时间：2023-08-25 15:47:43

组件介绍

TUIPusher & TUIPlayer 是 Web 端开源的含 UI 直播互动组件。TUIPusher & TUIPlayer 集成 实时音视频 TRTC 、 即时通信 IM 等基础 SDK，为企业直播、电商带货、行业培训、远程教学等多种直播场景提供快速上线 Web 端直播推拉流工具的解决方案。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TUIPusher & TUIPlayer 的优势：

- 贴合直播场景需求，提供了含 UI 的直播场景通用解决方案，覆盖了直播场景常见功能（如设备选择、美颜、直播推流、观众拉流、聊天等），助力业务快速上线。
- 直接接入腾讯云实时音视频 TRTC、腾讯云即时通信 IM 以及腾讯云超级播放器 TCPlayer 等基础 SDK，方便客户灵活扩展业务功能。
- Web 端易于用户使用，易于功能迭代的天然优势。



快速体验

为了方便您快速体验 TUIPusher & TUIPlayer 的功能，我们结合用户管理系统和房间管理系统提供了 TUIPusher 体验链接 及 TUIPlayer 体验链接 。

> **注意**
> 同时体验 TUIPusher 和 TUIPlayer 需要使用两个不同的账号登录。

TUIPusher 推流组件功能介绍

- 支持采集摄像头和麦克风的流并推流
  - 可根据需求设置视频参数（帧率，分辨率，码率）

- 支持开启美颜并设置视频美颜参数
- 支持采集屏幕分享流并推流
- 支持推流到腾讯云实时音视频后台，推流到腾讯云 CDN
- 支持在线聊天室，和在线观众进行聊天互动
- 支持获取观众列表，对在线观众进行禁言操作

TUIPlayer 拉流组件功能介绍

- 支持同时播放音视频流和屏幕分享流
- 支持在线聊天室，和主播及其他观众进行聊天互动
- 支持超低延时直播（300ms 延时），快直播（1000ms 以内延时）以及标准直播（支持超高并发观看）三种拉流线路
- 兼容桌面浏览器及移动端浏览器，支持移动端浏览器横屏观看

> **注意**
> 部分浏览器不支持 WebRTC，只能使用标准直播线路观看，如需体验其他线路，请尝试更换浏览器。

组件集成

步骤一：开通腾讯云服务

> **注意**
> - TUIPusher & TUIPlayer 基于腾讯云实时音视频和即时通信服务进行开发。实时音视频 TRTC 应用与 即时通信 IM 应用的 SDKAppID 一致，才能复用账号与鉴权。
> - 即时通信 IM 应用针对文本消息，提供基础版本的 安全打击 能力，如果希望使用自定义不雅词功能，可以单击 **升级** 或在 购买页 购买 **安全打击-高级版** 服务。
> - 本地计算 UserSig 的方式仅用于本地开发调试，请勿直接发布到线上，一旦 SECRETKEY 泄露，攻击者就可以盗用您的腾讯云流量。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 。

方式1：基于实时音视频

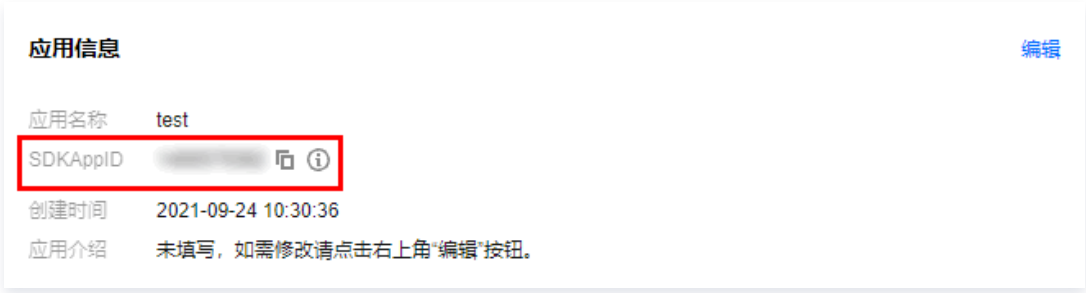
步骤1：创建实时音视频 TRTC 应用

1. 注册腾讯云账号 并开通 实时音视频 和 即时通信 服务。
2. 在 实时音视频控制台 单击 **应用管理 > 创建应用** 创建新应用。



步骤2: 获取 TRTC 密钥信息

1. 在 **应用管理 > 应用信息** 中获取 SDKAppID 信息。



2. 在 **应用管理 > 快速上手** 中获取应用的 secretKey 信息。



> **说明**
> - 首次创建实时音视频应用的腾讯云账号，可获赠一个10000分钟的音视频资源免费试用包。
> - 创建实时音视频应用之后会自动创建一个 SDKAppID 相同的即时通信 IM 应用，可在 即时通信控制台 配置该应用的套餐信息。

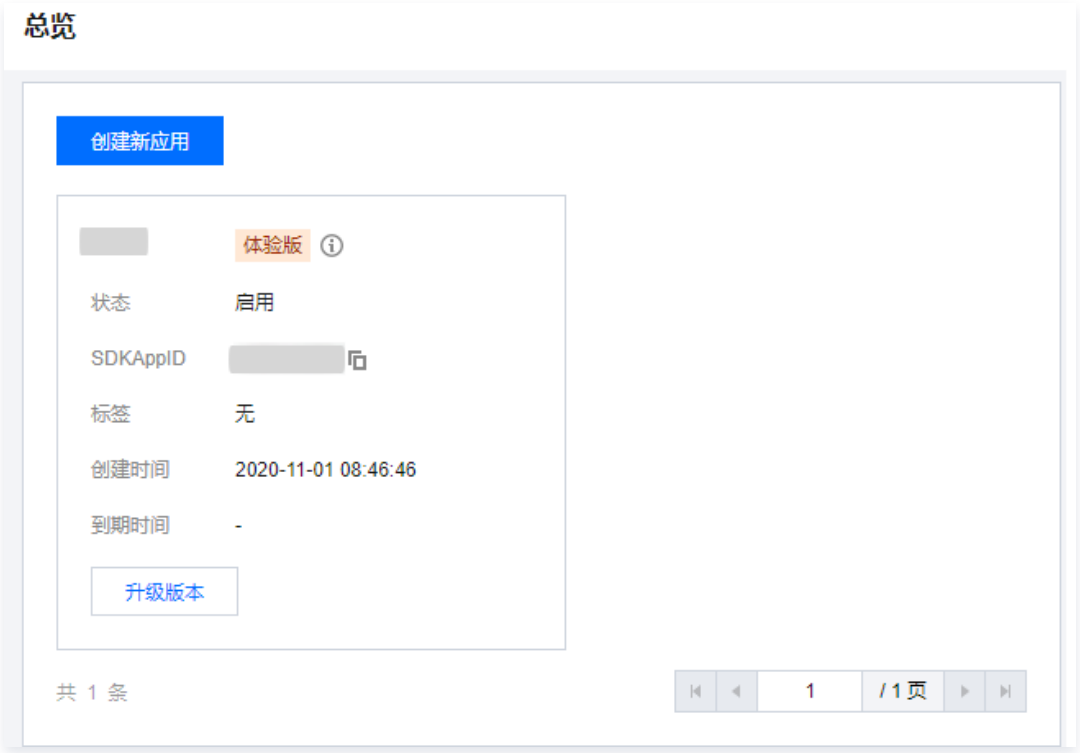
方式2：基于即时通信 IM

步骤1：创建即时通信 IM 应用

1. 登录 即时通信 IM 控制台，单击 **创建新应用** 将弹出对话框。



2. 输入您的应用名称，单击 **确认** 即可完成创建。



3. 您可在 即时通信 IM 控制台 总览页面查看新建应用的状态、业务版本、SDKAppID、创建时间以及到期时间。请记录 SDKAppID 信息。

步骤2：获取 IM 密钥并开通实时音视频服务

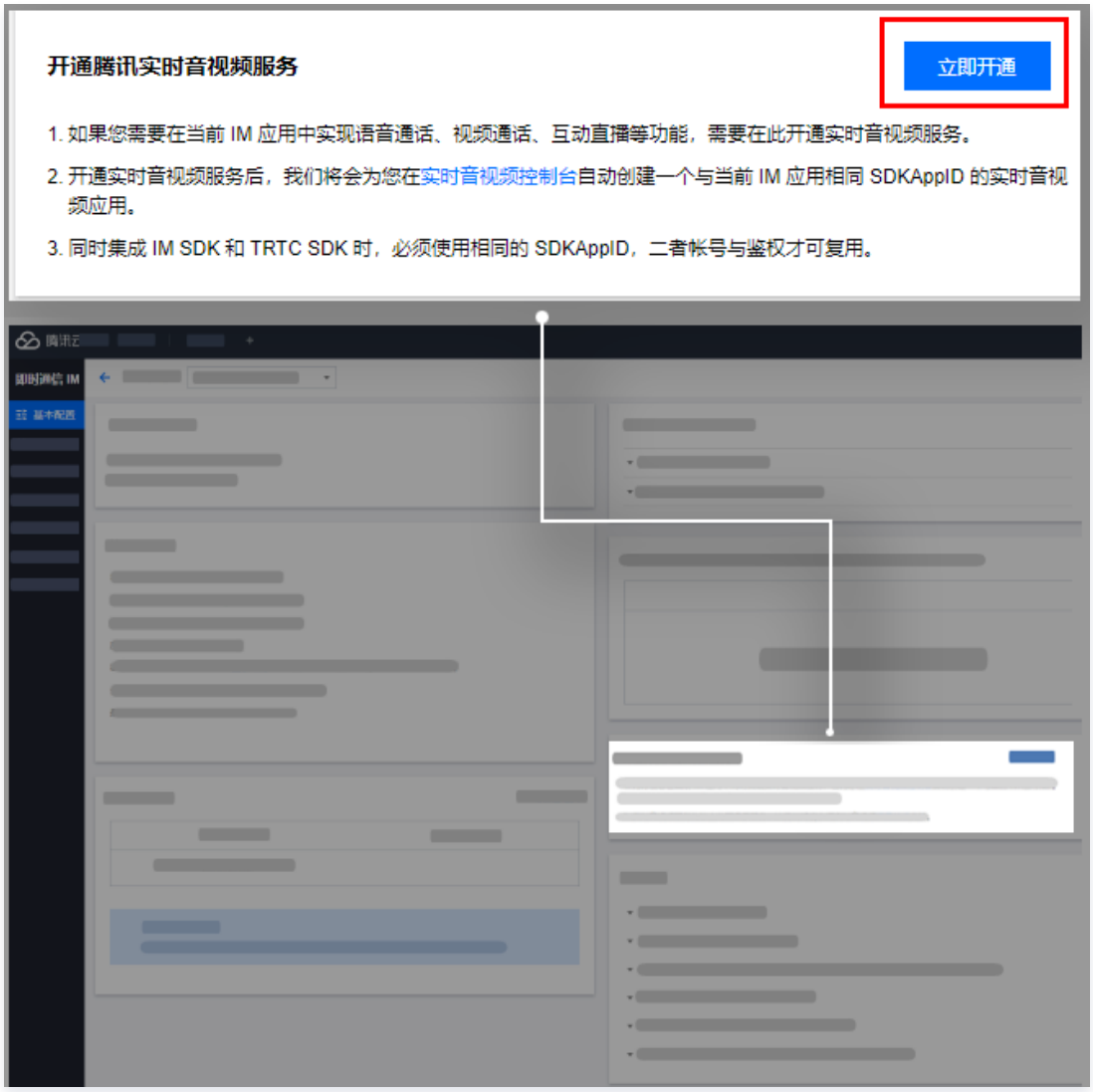
1. 在 即时通信 IM 控制台 总览页单击您创建完成的即时通信 IM 应用，随即跳转至该应用的基础配置页。在 **基本信息** 区域，单击 **显示密钥**，复制并保存密钥信息。



注意

请妥善保管密钥信息，谨防泄露。

2. 在该应用的基础配置页，开通腾讯云实时音视频服务。



步骤二：项目准备

开发环境要求：

- Vue 2
- node（v14.16.0）
- npm（版本请与 node 版本匹配）

1. 在 GitHub 下载 TUIPusher & TUIPlayer 代码。
2. 为 TUIPusher & TUIPlayer 安装依赖。

```
cd Web/TUIPusher
npm install

cd Web/TUIPlayer
npm install
```

3. 将 sdkAppId 和 secretKey 填入 `TUIPusher/src/config/basic-info-config.js` 及 `TUIPlayer/src/config/basic-info-config.js` 配置文件中。

```
/**
 * 腾讯云 SDKAppId，需要替换为您自己账号下的 SDKAppId。
 *
 * 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav ) 创建应用，即可看到 SDKAppId，
 * 它是腾讯云用于区分客户的唯一标识。
 */
export const sdkAppId = 0;
/**
 * 签名过期时间，建议不要设置的过短
 * <p>
 * 时间单位：秒
 * 默认时间：7 x 24 x 60 x 60 = 604800 = 7 天
 */
export const expireTime = 604800;

/**
 * 计算签名用的加密密钥，获取步骤如下：
 *
 * step1. 进入腾讯云实时音视频[控制台](https://console.cloud.tencent.com/rav )，如果还没有应用就创建一
 * step2. 单击“应用配置”进入基础配置页面，并进一步找到“帐号体系集成”部分。
 * step3. 点击“查看密钥”按钮，就可以看到计算 UserSig 使用的加密的密钥了，请将其拷贝并复制到如下的变量中
 *
 * 注意：该方案仅适用于调试Demo，正式上线前请将 UserSig 计算代码和密钥迁移到您的后台服务器上，以避免加密密钥泄
 * 文档：https://cloud.tencent.com/document/product/647/17275#Server
 */
export const secretKey = '';
```

4. 本地开发环境运行 TUIPusher & TUIPlayer。

```
cd Web/TUIPusher
npm run serve

cd Web/TUIPlayer
npm run serve
```

5. 可打开 http://localhost:8080 和 http://localhost:8081 体验 TUIPusher 和 TUIPlayer 功能。
6. 可更改 TUIPusher/src/config/basic-info-config.js 及 TUIPlayer/src/config/basic-info-config.js 配置文件中的房间，主播及观众等信息，**注意保持 TUIPusher 和 TUIPlayer 的房间信息，主播信息一致**。

> **注意**
> - 完成以上配置，您可以使用 TUIPusher & TUIPlayer 进行超低延时直播，如您需要支持快直播和标准直播，请继续阅读 步骤三：旁路直播 。
> - 本地计算 UserSig 的方式仅用于本地开发调试，请勿直接发布到线上，一旦您的 SECRETKEY 泄露，攻击者就可以盗用您的腾讯云流量。
> - 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 。

步骤三：旁路直播

TUIPusher & TUIPlayer 实现的快直播和标准直播依托于腾讯云 云直播服务 ，因此支持快直播和标准直播线路需要您开启旁路推流功能。

1. 在 实时音视频控制台 中为您正在使用的应用开启旁路推流配置，可按需开启指定流旁路或全局自动旁路。



2. 请在 **域名管理** 页面添加自有播放域名，具体请参见 添加自有域名 。

3. 在 `TUIPlayer/src/config/basic-info-config.js` 配置文件中配置播放域名。

完成以上配置，您可以体验 TUIPusher & TUIPlayer 支持超低延时直播，快直播以及标准直播的所有功能。

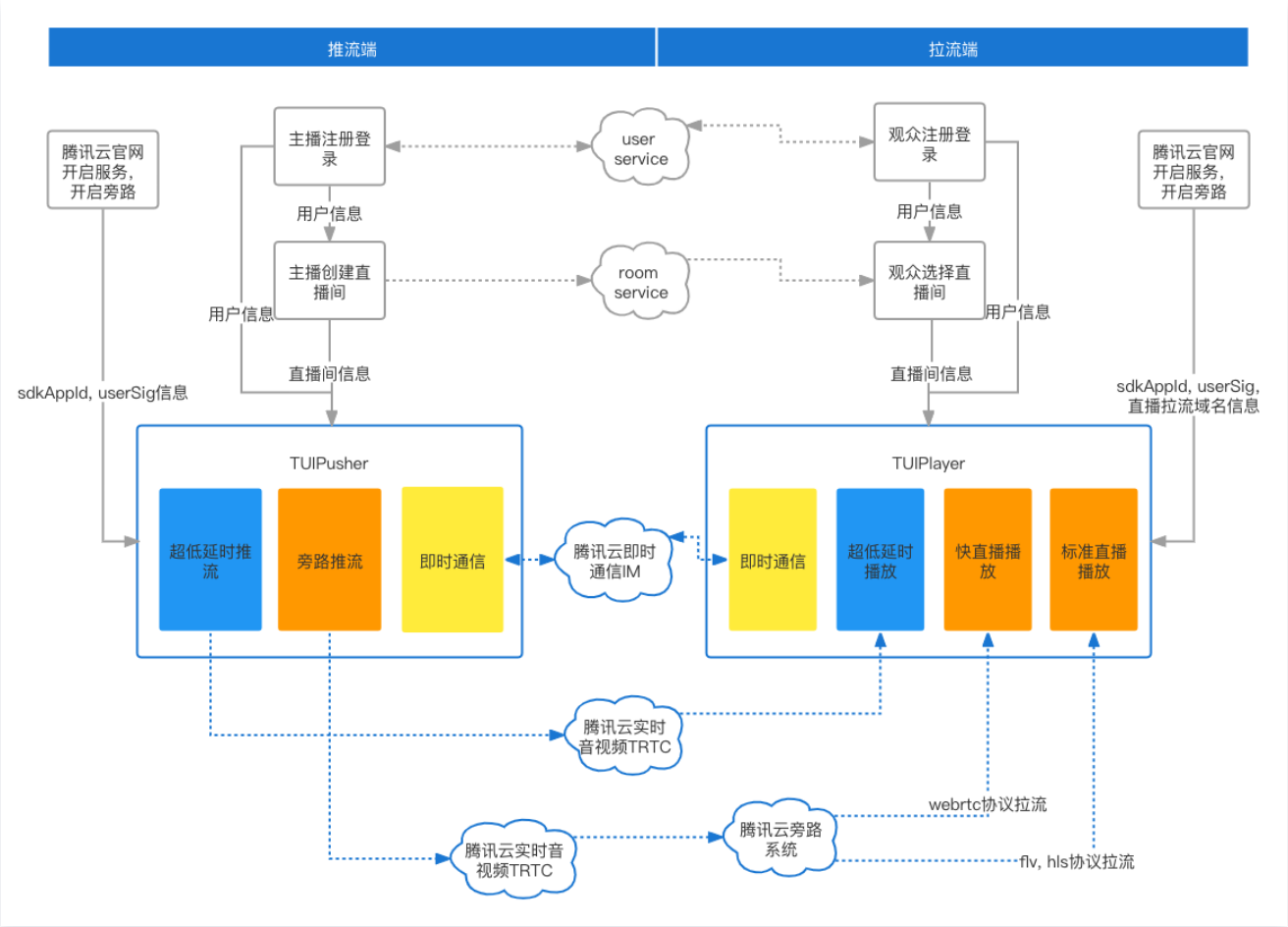
步骤四：生产环境应用

当您将 TUIPusher & TUIPlayer 用于生产应用时，在接入 TUIPusher & TUIPlayer 之外，您需要：

- 创建用户管理系统，用于管理产品用户信息，包括但不限于用户 ID，用户名，用户头像等。
- 创建房间管理系统，用于管理产品直播间信息，包括但不限于直播间 ID、直播间名称，直播间主播信息等。
- 服务端生成 UserSig。

> **注意**
> - 本文生成 UserSig 的方式，是在客户端根据您填入的 sdkAppId 及 secretKey 生成 userSig，该方式的 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 TUIPusher & TUIPlayer 进行功能调试**。
> - 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 时由您的应用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 。

- 参考 `TUIPusher/src/pusher.vue` 及 `TUIPlayer/src/player.vue` 文件，将用户信息、直播间信息、SDKAppId 及 UserSig 等账号信息提交到 vuex 的 store 进行全局存储，您就可以跑通推拉流两个客户端的所有功能。详细业务流程参见下图：



相关问题

Web 端如何实现美颜功能?

请参见 开启美颜 。

Web 端如何实现屏幕共享?

请参见 屏幕分享 。

Web 端如何实现云端录制?

1. 开启**云端录制**功能，具体操作请参见 实现云端录制与回放 。
2. 开启**云端录制> 指定用户录制**之后，Web 端可通过在调用 TRTC.createClient 接口时传入 userDefineRecordId 参数开启录制。

Web 端如何实现推流到 CDN？

Web 端推流到 CDN 请参见 实现推流到 CDN 。

Web 端如何实现快直播拉流?

实现快直播拉流的方式是通过 Web SDK 推流到 CDN 之后使用 WebRTC 协议拉流，具体请参见 快直播拉流 > Web（H5）播放器 。

Web 端如何实现标准直播拉流?

实现标准直播拉流，请参见 标准直播拉流 > Web（H5）播放器 。

注意事项

平台支持

操作系统	浏览器类型	浏览器最低版本要求	TUIPlayer	TUIPusher	TUIPusher 屏幕分享
Mac OS	桌面版 Safari 浏览器	11+	支持	支持	支持（需要 Safari13+ 版本）
Mac OS	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Mac OS	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Mac OS	桌面版 Edge 浏览器	80+	支持	支持	支持
Mac OS	桌面版微信内嵌网页	-	支持	不支持	不支持
Mac OS	桌面版企业微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版 Chrome 浏览器	56+	支持	支持	支持（需要 Chrome72+ 版本）
Windows	桌面版 QQ 浏览器（极速内核）	10.4+	支持	支持	不支持
Windows	桌面版 Firefox 浏览器	56+	支持	支持	支持（需要 Firefox66+ 版本）
Windows	桌面版 Edge 浏览器	80+	支持	支持	支持
Windows	桌面版微信内嵌网页	-	支持	不支持	不支持
Windows	桌面版企业微信内嵌网页	-	支持	不支持	不支持
iOS	微信内嵌浏览器	-	支持	不支持	不支持
iOS	企业微信内嵌浏览器	-	支持	不支持	不支持
iOS	移动版 Safari 浏览器	-	支持	不支持	不支持
iOS	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	微信内嵌浏览器	-	支持	不支持	不支持
Android	企业微信浏览器	-	支持	不支持	不支持
Android	移动版 Chrome 浏览器	-	支持	不支持	不支持
Android	移动版 QQ 浏览器	-	支持	不支持	不支持
Android	移动版 Firefox 浏览器	-	支持	不支持	不支持
Android	移动端 UC 浏览器	-	支持（仅支持标准直播观看）	不支持	不支持

域名要求

出于对用户安全、隐私等问题的考虑，浏览器限制网页在 HTTPS 协议下才能正常使用 TUIPusher & TUIPlayer 的全部功能。为确保生产环境用户顺畅接入和体验 TUIPusher & TUIPlayer 的全部功能，请使用 HTTPS 协议访问音视频应用页面。

> **注意**
> 本地开发可以通过 `http://localhost` 协议进行访问。

URL 域名及协议支持情况请参考如下表格：

应用场景	协议	TUIPlayer	TUIPusher	TUIPusher 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	-

防火墙限制

TUIPusher & TUIPlayer 依赖以下端口进行数据传输，请将其加入防火墙白名单。

- TCP 端口：8687
- UDP 端口：8000，8080，8800，843，443，16285
- 域名：qcloud.rtc.qq.com

结语

在后续的迭代中, TRTC Web 端推拉流组件会逐渐与 iOS、Andriod 等各端连通，并在 Web 端实现观众连麦、高级美颜、自定义布局、转推多平台、上传图片文字音乐等能力，欢迎大家多多使用、提出您的宝贵意见。
如果有任何需要或者反馈，可扫描下方二维码，或者单击 反馈链接 同步给我们。

此外，我们欢迎加入 TUIRoomKit 技术交流 QQ 群（群号：**770645461**）进行技术交流和问题反馈。

TUILiveRoom API 查询
TRTCLiveRoom API (iOS)

最近更新时间：2023-08-24 16:22:41

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（iOS）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
delegate	设置事件回调。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，`enterRoom()` 成功后调用才有效。
getAudienceList	获取房间内所有的观众信息，`enterRoom()` 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播移除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager 。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

调试相关接口函数

API	描述
showVideoDebugLog	是否在界面中展示 Debug 信息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	直播房间信息变更回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述

onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onKickoutJoinAnchor	连麦观众收到被移出连麦的通知。

主播 PK 事件回调

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

delegate

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
@property(nonatomic, weak)id<TRTCLiveRoomDelegate> delegate;
```

> **说明**
> delegate 是 TRTCLiveRoom 的代理回调。

login

登录。

```
/// 登录到组件系统
/// - Parameters:
///   - sdkAppID: 您可以在实时音视频控制台 > 【[应用管理](https://console.cloud.tencent.com/trtc/app)】> 应用信息中查看 SDKAppID。
///   - userID: 当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（\_）。
///   - userSig: 腾讯云设计的一种安全保护签名，获取方式请参见 [如何计算及使用 UserSig](https://cloud.tencent.com/document/product/647/17275)。
///   - config: 全局配置信息，请在登录时初始化，登录之后不可变更。 isAttachedTUIKit 项目中是否引入并使用TUIKit
///   - callback: 登录回调，成功时 code 为0。
/// - Note:
///   - userSig 建议设定 7 天，能够有效规避 usersign 过期导致的 IM 收发消息失败、TRTC 连麦失败等情况
- (void)loginWithSdkAppID:(int)sdkAppID
                   userID:(NSString *)userID
                  userSig:(NSString *)userSig
                   config:(TRTCLiveRoomConfig *)config
                 callback:(Callback _Nullable)callback
NS_SWIFT_NAME(login(sdkAppID:userID:userSig:config:callback:));
```

参数如下表所示：

参数	类型	含义
sdkAppID	Int	您可以在实时音视频控制台 >应用管理> 应用信息中查看 SDKAppID。
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 • useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 > 域名管理页面中进行设置。
callback	(_ code: Int, _ message: String?) -> Void	登录回调，成功时 code 为0。

logout

登出。

```
// 退出登录
/// - Parameter callback:  登出回调，成功时 code 为0
- (void)logout:(Callback _Nullable)callback
NS_SWIFT_NAME(logout(_:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
/// 设置用户信息，您设置的用户信息会被存储于腾讯云 IM 云服务中。
/// - Parameters:
///   - name: 用户昵称
///   - avatarURL: 用户头像地址
///   - callback: 个人信息设置回调，成功时 code 为0
- (void)setSelfProfileWithName:(NSString *)name
                     avatarURL:(NSString * _Nullable)avatarURL
                      callback:(Callback _Nullable)callback
NS_SWIFT_NAME(setSelfProfile(name:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
name	String	昵称。
avatarURL	String	头像地址。
callback	(_ code: Int, _ message: String?) -> Void	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
/// 创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
/// 主播开播的正常调用流程是：
/// 1.【主播】调用 startCameraPreview() 打开摄像头预览，此时可以调整美颜参数。
/// 2.【主播】调用 createRoom() 创建直播间，房间创建成功与否会通过 callback 通知给主播。
/// 3.【主播】调用 startPublish() 开始推流。
/// - Parameters:
///   - roomID: 房间标识，需要由您分配并进行统一管理。多个 roomid 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
///   - roomParam: TRTCCreateRoomParam | 房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
///   - callback:  进入房间的结果回调，成功时 code 为0。
/// - Note:
///   - 主播开始直播的时候调用，可重复创建自己已创建过的房间。
- (void)createRoomWithRoomID:(UInt32)roomID
                  roomParam:(TRTCCreateRoomParam *)roomParam
                   callback:(Callback _Nullable)callback
NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您自行管理，可忽略该参数。
callback	(_ code: Int, _ message: String?) -> Void	创建房间的结果回调，成功时 code 为0。

主播开播的正常调用流程如下：

1. 主播调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. 主播调用 `createRoom()` 创建直播间，房间创建成功与否会通过 callback 通知给主播。
3. 主播调用 `starPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
/// 销毁房间（主播调用）
/// 主播在创建房间后，可以调用这个函数来销毁房间。
/// - Parameter callback: 销毁房间的结果回调，成功时 code 为0。
/// - Note:
///   - 主播在创建房间后，可以调用该函数来销毁房间。
- (void)destroyRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（观众调用）。

```
/// 进入房间（观众调用）
/// 观众观看直播的正常调用流程是：
/// 1.【观众】向您的服务端获取最新的直播间列表，其中有多个直播间的 roomid 和房间信息。
/// 2.【观众】观众选择一个直播间以后，调用 enterRoom() 进入该房间。
/// 3.【观众】如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom() 成功后调用
startPlay(userID) 即可播放主播的画面。
/// 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自
TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
/// - Parameters:
///   - roomID: 房间标识。
///   - useCDNFirst: 是否优先使用CDN播放
///   - cdnDomain: CDN域名
///   - callback: 进入房间的结果回调，成功时 code 为0。
/// - Note:
///   - 观众进入直播房间的时候调用
///   - 主播不可调用这个接口进入自己已创建的房间，而要用createRoom
- (void)enterRoomWithRoomID:(UInt32)roomID
              useCDNFirst:(BOOL)useCDNFirst
                cdnDomain:(NSString * _Nullable)cdnDomain
                 callback:(Callback)callback
NS_SWIFT_NAME(enterRoom(roomID:useCDNFirst:cdnDomain:callback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	房间标识。
callback	(_ code: Int, _ message: String?) -> Void	进入房间的结果回调，成功时 code 为0。

观众观看直播的正常调用流程如下：

1. 观众向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. 观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. 观众调用 `startPlay(userID)` 并传入主播的 userID 开始播放。
   - 若直播间列表已包含主播端的 userID 信息，观众端可直接调用 `startPlay(userID)` 即可开始播放。
   - 若在进房前暂未获取主播的 userID，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 的事件回调，该回调中携带主播的 userID 信息，再调用 `startPlay(userID)` 即可播放。

exitRoom

离开房间。

```
/// 退出房间（观众调用）
/// - Parameter callback: 退出房间的结果回调，成功时 code 为0。
/// - Note:
///   - 观众离开直播房间的时候调用
///   - 主播不可调用这个接口离开房间

- (void)exitRoom:(Callback _Nullable)callback
NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	退出房间的结果回调，成功时 code 为0。

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> **说明**
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
/// 获取房间列表的详细信息
/// 其中的信息是主播在创建 createRoom() 时通过 roomInfo 设置进来的，如果房间列表和房间信息都由您自行管理，可忽略该函数。
/// - Parameter roomIDs: 房间号列表
/// - Parameter callback: 房间详细信息回调
- (void)getRoomInfosWithRoomIDs:(NSArray<NSNumber *> *)roomIDs
                       callback:(RoomInfoCallback _Nullable)callback
NS_SWIFT_NAME(getRoomInfos(roomIDs:callback:));
```

参数如下表所示：

参数	类型	含义
roomIDs	[UInt32]	房间号列表。
callback	(_ code: Int, _ message: String?, _ roomList: [TRTCLiveRoomInfo]) -> Void	房间详细信息回调。

getAnchorList

获取房间内所有的主播列表， enterRoom() 成功后调用才有效。

```
/// 获取房间内所有的主播列表，enterRoom() 成功后调用才有效。
/// - Parameter callback: 用户详细信息回调
- (void)getAnchorList:(UserListCallback _Nullable)callback
NS_SWIFT_NAME(getAnchorList(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String, _ userList: [TRTCLiveUserInfo]) -> Void	用户详细信息回调。

getAudienceList

获取房间内所有的观众信息， enterRoom() 成功后调用才有效。

```
/// 获取房间内所有的观众信息，enterRoom() 成功后调用才有效。
/// - Parameter callback: 用户详细信息回调
- (void)getAudienceList:(UserListCallback _Nullable)callback
NS_SWIFT_NAME(getAudienceList(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String, _ userList: [TRTCLiveUserInfo]) -> Void	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
/// 开启本地视频的预览画面
/// - Parameters:
///   - frontCamera: true：前置摄像头；false：后置摄像头。
///   - view: 承载视频画面的控件。
///   - callback: 操作回调。
- (void)startCameraPreviewWithFrontCamera:(BOOL)frontCamera
                                     view:(UIView *)view
                                 callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startCameraPreview(frontCamera:view:callback:));
```

参数如下表所示：

参数	类型	含义
frontCam	Bool	true：前置摄像头；false：后置摄像头。

era		
view	UIView	承载视频画面的控件。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

stopCameraPreview

停止本地视频采集及预览。

```
/// 停止本地视频采集及预览
- (void)stopCameraPreview;
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用
- 观众开始连麦时调用

```
/// 开始直播（推流），适用于如下两种场景：
/// 1. 主播开播的时候调用
/// 2. 观众开始连麦时调用
/// - Parameters:
///   - streamID: 用于绑定直播 CDN 的 streamId，如果您希望您的观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
///   - callback: 操作回调
- (void)startPublishWithStreamID:(NSString *)streamID
                        callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startPublish(streamID:callback:));
```

参数如下表所示：

参数	类型	含义
streamID	String	用于绑定直播 CDN 的 streamID，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamID。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用。
- 观众结束连麦时调用。

```
/// 停止直播（推流），适用于如下两种场景：
/// 1. 主播结束直播时调用
/// 2. 观众结束连麦时调用
/// - Parameter callback: 操作回调。
- (void)stopPublish:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPublish(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
/// 播放远端视频画面，可以在普通观看和连麦场景中调用
///【普通观看场景】
/// 1. 如果您的服务器所管理的房间列表中包含每一个房间的主播 userID，则可以直接在 enterRoom() 成功后调用 startPlay(userID) 即可播放主播的画面。
/// 2. 如果您管理的房间列表只有 roomid 也没有关系，观众在 enterRoom() 成功后很快会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调。
/// 此时使用回调中的 userID 调用 startPlay(userID) 即可播放主播的画面。
///【直播连麦场景】
/// 发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userID) 回调，此时使用回调中的 userID 调用 startPlay(userID) 即可播放连麦画面。
/// - Parameters:
///   - userID: 需要观看的用户 ID。
///   - view: 承载视频画面的 view 控件。
///   - callback: 操作回调。
- (void)startPlayWithUserID:(NSString *)userID
                       view:(UIView *)view
                   callback:(Callback _Nullable)callback
NS_SWIFT_NAME(startPlay(userID:view:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	需要观看的用户 ID。
view	UIView	承载视频画面的 view 控件。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

- **普通观看场景**
  - 若直播间列表已包含主播端的 userID 信息，观众端可以直接在 `enterRoom()` 成功后调用 `startPlay(userID)` 播放主播的画面。
  - 若在进房前暂未获取主播的 userID，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 的事件回调，该回调中携带主播的 userID 信息，再调用 `startPlay(userID)` 即可播放主播的画面。
- **直播连麦场景**
  发起连麦后，主播会收到来自 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userID)` 回调，此时使用回调中的 userID 调用 `startPlay(userID)` 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 `onAnchorExit()` 回调时，调用该接口。

```
/// 停止渲染远端视频画面
/// - Parameters:
///   - userID: 对方的用户信息。
///   - callback: 操作回调。
/// - Note:
///   - 在 onAnchorExit 回调时，调用这个接口
- (void)stopPlayWithUserID:(NSString *)userID
                  callback:(Callback _Nullable)callback
NS_SWIFT_NAME(stopPlay(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	对方的用户信息。
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
/// 观众端请求连麦
/// - Parameters:
///   - reason: 连麦请求原因。
///   - responseCallback: 请求连麦的回调。
/// - Note: 观众发起请求后，主播端会收到`onRequestJoinAnchor`回调
- (void)requestJoinAnchor:(NSString *)reason
                  timeout:(double)timeout
          responseCallback:(ResponseCallback _Nullable)responseCallback
NS_SWIFT_NAME(requestJoinAnchor(reason:timeout:responseCallback:));
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	long	主播响应回调。
responseCallback	(_ agreed: Bool, _ reason: String?) -> Void	主播响应回调。

主播和观众的连麦流程如下：

1. 观众调用 `requestJoinAnchor()` 向主播发起连麦请求。
2. 主播会收到 `TRTCLiveRoomDelegate` 的 `onRequestJoinAnchor()` 回调通知。
3. 主播调用 `responseJoinAnchor()` 决定是否接受来自观众的连麦请求。
4. 观众会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 观众如果请求被同意，则调用 `startCameraPreview()` 开启本地摄像头。

版权所有：腾讯云计算（北京）有限责任公司

6. **观众调用** `startPublish()` **正式进入推流状态。**
7. **主播一旦观众进入连麦状态，主播会收到** `TRTCLiveRoomDelegate` **的** `onAnchorEnter()` **通知。**
8. **主播调用** `startPlay()` **即可看到连麦观众的视频画面。**
9. **观众如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到** `onAnchorEnter()` **通知，调用** `startPlay()` **播放其他连麦者的视频画面。**

responseJoinAnchor

主播处理连麦请求。主播在收到 `TRTCLiveRoomDelegate` **的** `onRequestJoinAnchor()` **回调后，需要调用该接口来处理观众的连麦请求。**

```
/// 主播回复观众连麦请求
/// - Parameters:
///   - user: 观众 ID。
///   - agree: true：同意；false：拒绝。
///   - reason: 同意/拒绝连麦的原因描述。
/// - Note: 主播回复后，观众端会收到`requestJoinAnchor`传入的`responseCallback`回调
- (void)responseJoinAnchor:(NSString *)userID
                     agree:(BOOL)agree
                    reason:(NSString *)reason
NS_SWIFT_NAME(responseJoinAnchor(userID:agree:reason:));
```

参数如下表所示：

参数	类型	含义
userID	String	观众 ID。
agree	Bool	true：同意；false：拒绝。
reason	String?	同意/拒绝连麦的原因描述。

kickoutJoinAnchor

主播移除连麦观众。主播调用此接口移除连麦观众后，被移连麦观众会收到 `TRTCLiveRoomDelegate` **的** `onKickoutJoinAnchor()` **回调通知。**

```
/// 主播移除连麦观众
/// - Parameters:
///   - userID: 连麦观众 ID。
///   - callback: 操作回调。
/// - Note: 主播调用此接口移除连麦观众后，被移连麦观众会收到 trtcLiveRoomOnKickoutJoinAnchor() 回调通知
- (void)kickoutJoinAnchor:(NSString *)userID
                 callback:(Callback _Nullable)callback
NS_SWIFT_NAME(kickoutJoinAnchor(userID:callback:));
```

参数如下表所示：

参数	类型	含义
userID	String	连麦观众 ID。
callback	(_ code: Int, _ message: String?) ->	操作回调。

	Void	

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
/// 主播请求跨房 PK
/// - Parameters:
///   - roomID: 被邀约房间 ID。
///   - userID: 被邀约主播 ID。
///   - responseCallback: 请求跨房 PK 的结果回调。
/// - Note: 发起请求后，对方主播会收到 `onRequestRoomPK` 回调
- (void)requestRoomPKWithRoomID:(UInt32)roomID
                         userID:(NSString *)userID
                responseCallback:(ResponseCallback _Nullable)responseCallback
NS_SWIFT_NAME(requestRoomPK(roomID:userID:responseCallback:));
```

参数如下表所示：

参数	类型	含义
roomID	UInt32	被邀约房间 ID。
userID	String	被邀约主播 ID。
responseCallback	(_ agreed: Bool, _ reason: String?) -> Void	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 主播 A调用 `requestRoomPK()` 向主播 B 发起连麦请求。
2. 主播 B会收到 `TRTCLiveRoomDelegate` 的 `onRequestRoomPK()` 回调通知。
3. 主播 B调用 `responseRoomPK()` 决定是否接受主播 A 的 PK 请求。
4. 主播 B如果接受主播 A 的请求，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来显示主播 A 的视频画面。
5. 主播 A会收到 `responseCallback` 回调通知，该通知会携带来自主播 B 的处理结果。
6. 主播 A如果请求被同意，等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调。

```
/// 响应跨房 PK 请求
/// 主播响应其他房间主播的 PK 请求。
/// - Parameters:
///   - user: 发起 PK 请求的主播 ID
///   - agree: true：同意；false：拒绝
///   - reason: 同意/拒绝 PK 的原因描述
/// - Note: 主播回复后，对方主播会收到 `requestRoomPK` 传入的 `responseCallback` 回调
- (void)responseRoomPKWithUserID:(NSString *)userID
                          agree:(BOOL)agree
```

```
                reason:(NSString *)reason
NS_SWIFT_NAME(responseRoomPK(userID:agree:reason:));
```

参数如下表所示：

参数	类型	含义
userID	String	发起 PK 请求的主播 ID。
agree	Bool	true：同意；false：拒绝。
reason	String?	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `trtcLiveRoomOnQuitRoomPK()` 回调通知。

```
/// 主播退出跨房 PK
/// - Parameter callback: 退出跨房 PK 的结果回调
/// - Note: 当两个主播中的任何一个退出跨房 PK 状态后，另一个主播会收到 `trtcLiveRoomOnQuitRoomPK` 回调通知。
- (void)quitRoomPK:(Callback _Nullable)callback
NS_SWIFT_NAME(quitRoomPK(callback:));
```

参数如下表所示：

参数	类型	含义
callback	(_ code: Int, _ message: String?) -> Void	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
/// 切换前后摄像头
- (void)switchCamera;
```

setMirror

设置是否镜像展示。

```
/// 设置是否镜像展示
/// - Parameter isMirror: 开启/关闭镜像。
- (void)setMirror:(BOOL)isMirror
NS_SWIFT_NAME(setMirror(isMirror:));
```

参数如下表所示：

参数	类型	含义

isMirror	Bool	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
/// 静音本地音频。
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteLocalAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteLocalAudio(isMuted:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
/// 静音远端音频
/// - Parameters:
///   - userID: 远端的用户ID。
///   - isMuted: true：开启静音；false：关闭静音。
- (void)muteRemoteAudioWithUserID:(NSString *)userID isMuted:(BOOL)isMuted
NS_SWIFT_NAME(muteRemoteAudio(userID:isMuted:));
```

参数如下表所示：

参数	类型	含义
userID	String	远端的用户 ID。
isMuted	Bool	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
/// 静音所有远端音频
/// - Parameter isMuted: true：开启静音；false：关闭静音。
- (void)muteAllRemoteAudio:(BOOL)isMuted
NS_SWIFT_NAME(muteAllRemoteAudio(_:));
```

参数如下表所示：

参数	类型	含义
isMuted	Bool	true：开启静音；false：关闭静音。

setAudioQuality

设置音频质量

```
/// 设置音频质量，支持的值为1 2 3，代表低中高
/// - Parameter quality 音频质量
- (void)setAudioQuality:(NSInteger)quality
NS_SWIFT_NAME(setAudioiQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	NSInteger	1：语音；2：标准； 3：音乐。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
/// 获取音效管理对象
- (TXAudioEffectManager *)getAudioEffectManager;
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager 。

```
/* 获取美颜管理对象 TXBeautyManager
*
* 通过美颜管理，您可以使用以下功能：
* - 设置"美颜风格"、“美白”、“红润”、“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”等美容效果。
* - 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”
* - 设置人脸挂件（素材）等动态效果
* - 添加美妆
* - 进行手势识别
*/
- (TXBeautyManager *)getBeautyManager;
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“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”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
/// 发送文本消息，房间内所有成员都可见
/// - Parameters:
///   - message: 文本消息。
///   - callback: 发送回调。
- (void)sendRoomTextMsg:(NSString *)message callback:(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
/// 发送自定义消息
/// - Parameters:
///   - command: 命令字，由开发者自定义，主要用于区分不同消息类型
///   - message: 本文消息。
///   - callback: 发送回调。
- (void)sendRoomCustomMsgWithCommand:(NSString *)command message:(NSString *)message callback:
(Callback _Nullable)callback
NS_SWIFT_NAME(sendRoomCustomMsg(command:message:callback:));
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	(_ code: Int, _ message: String?) -> Void	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示 Debug 信息。

```
/// 是否在界面中展示debug信息
/// - Parameter isShow: 开启/关闭 Debug 信息显示。
- (void)showVideoDebugLog:(BOOL)isShow
NS_SWIFT_NAME(showVideoDebugLog(_:));
```

参数如下表所示：

参数	类型	含义
isShow	Bool	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
           onError:(NSInteger)code
           message:(NSString  *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onError:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码。
message	String?	错误信息。

onWarning

警告回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
         onWarning:(NSInteger)code
           message:(NSString *)message
NS_SWIFT_NAME(trtcLiveRoom(_:onWarning:message:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
code	Int	错误码 TRTCWarningCode。
message	String?	警告信息。

onDebugLog

Log 回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
```

```
        onDebugLog:(NSString *)log
NS_SWIFT_NAME(trtcLiveRoom(_:onDebugLog:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
log	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
      onRoomDestroy:(NSString *)roomID
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomDestroy:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
roomID	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
    onRoomInfoChange:(TRTCLiveRoomInfo *)info
NS_SWIFT_NAME(trtcLiveRoom(_:onRoomInfoChange:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
info	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
      onAnchorEnter:(NSString *)userID
```

```
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	新进房用户 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
       onAnchorExit:(NSString *)userID
NS_SWIFT_NAME(trtcLiveRoom(_:onAnchorExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
userID	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
    onAudienceEnter:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceEnter:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
     onAudienceExit:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onAudienceExit:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
 onRequestJoinAnchor:(TRTCLiveUserInfo *)user
              reason:(NSString * _Nullable)reason
             timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestJoinAnchor:reason:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	请求连麦观众信息。
reason	String?	连麦原因描述。
timeout	Double	处理请求的超时时间。

onKickoutJoinAnchor

连麦观众收到被移出连麦的通知。连麦观众收到被主播移除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

```
- (void)trtcLiveRoomOnKickoutJoinAnchor:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnKickoutJoinAnchor(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK ，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
    onRequestRoomPK:(TRTCLiveUserInfo *)user
            timeout:(double)timeout
NS_SWIFT_NAME(trtcLiveRoom(_:onRequestRoomPK:timeout:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
user	TRTCLiveUserInfo	发起跨房连麦的主播信息。
timeout	Double	处理请求的超时时间。

onQuitRoomPK

收到断开跨房 PK 通知。

```
- (void)trtcLiveRoomOnQuitRoomPK:(TRTCLiveRoom *)liveRoom
NS_SWIFT_NAME(trtcLiveRoomOnQuitRoomPK(_:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
   onRecvRoomTextMsg:(NSString *)message
            fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomTextMsg:fromUser:));
```

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)trtcLiveRoom:(TRTCLiveRoom *)trtcLiveRoom
onRecvRoomCustomMsgWithCommand:(NSString *)command
             message:(NSString *)message
            fromUser:(TRTCLiveUserInfo *)user
NS_SWIFT_NAME(trtcLiveRoom(_:onRecvRoomCustomMsg:message:fromUser:));
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
trtcLiveRoom	TRTCLiveRoomImpl	当前 TRTCLiveRoom 组件实例。
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
user	TRTCLiveUserInfo	发送者用户信息。

版权所有：腾讯云计算（北京）有限责任公司

TRTCLiveRoom API (Android)

最近更新时间：2023-08-25 15:47:43

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Android）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。

版权所有：腾讯云计算（北京）有限责任公司

getAudienceList	获取房间内所有的观众信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播踢除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

版权所有：腾讯云计算（北京）有限责任公司

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager 。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

调试相关接口函数

API	描述
showVideoDebugLog	是否在界面中展示 debug 信息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	直播房间信息变更回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述

版权所有：腾讯云计算（北京）有限责任公司

onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onKickoutJoinAnchor	连麦观众收到被踢出连麦的通知。

主播 PK 事件回调

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom 单例对象。

```
public static synchronized TRTCLiveRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCLiveRoomDelegate delegate);
```

> **说明**
> setDelegate 是 TRTCLiveRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCLiveRoom 中的各种状态通知回调会通过该 handler 通知给您，请勿与 setDelegate 混用。

login

登录。

```
public abstract void login(int sdkAppId,
 String userId, String userSig,
 TRTCLiveRoomDef.TRTCLiveRoomConfig config,
 TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
sdkAppID	Int	您可以在实时音视频控制台 >应用管理> 应用信息中查看 SDKAppID。
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。u • seCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 >域名管理 页面中进行设置。
callback	(_ code: Int, _ message: String?) -> Void	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
public abstract void createRoom(int roomId, TRTCLiveRoomDef.TRTCCreateRoomParam roomParam, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

主播开播的正常调用流程如下：

1. 主播 调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. 主播 调用 `createRoom()` 创建直播间，房间创建成功与否会通过 ActionCallback 通知给主播。
3. 主播 调用 `startPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
public abstract void destroyRoom(TRTCLiveRoomCallback.ActionCallback callback);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（观众调用）。

```
public abstract void enterRoom(int roomId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

观众观看直播的正常调用流程如下：

1. **观众** 向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. **观众** 观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. **观众** 调用 `startPlay(userId)` 并传入主播的 userId 开始播放。
   - 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay(userId)` 即可开始播放。
   - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放。

exitRoom

离开房间。

```
public abstract void exitRoom(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> **说明**
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfos(List<Integer> roomIdList, TRTCLiveRoomCallback.RoomInfoCallback
callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getAnchorList

获取房间内所有的主播列表，`enterRoom()` 成功后调用才有效。

```
public abstract void getAnchorList(TRTCLiveRoomCallback.UserListCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

getAudienceList

获取房间内所有的观众信息，`enterRoom()` 成功后调用才有效。

```
public abstract void getAudienceList(TRTCLiveRoomCallback.UserListCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	UserListCallback	用户详细信息回调。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
public abstract void startCameraPreview(boolean isFront, TXCloudVideoView view,
TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
isFront	boolean	true：前置摄像头；false：后置摄像头。
view	TXCloudVideoView	承载视频画面的控件。
callback	ActionCallback	操作回调。

stopCameraPreview

停止本地视频采集及预览。

```
public abstract void stopCameraPreview();
```

startPublish

开始直播（推流），适用于以下场景：

- **主播开播的时候调用**
- **观众开始连麦时调用**

```
public abstract void startPublish(String streamId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
streamId	String	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。
callback	ActionCallback	操作回调。

stopPublish

停止直播（推流），适用于以下场景：

- **主播结束直播时调用**
- **观众结束连麦时调用**

```
public abstract void stopPublish(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
public abstract void startPlay(String userId, TXCloudVideoView view, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	需要观看的用户id。
view	TXCloudVideoView	承载视频画面的 view 控件。
callback	ActionCallback	操作回调。

普通观看场景

- 若直播间列表已包含主播端的 userId 信息，观众端可以直接在 enterRoom() 成功后调用 startPlay(userId) 播放主播的画面。
- 若在进房前暂未获取主播的 userId，观众端在进房后会收到 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 的事件回调，该回调中携带主播的 userId 信息，再调用 startPlay(userId) 即可播放主播的画面。

直播连麦场景

发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 回调，此时使用回调中的 userId 调用 startPlay(userId) 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 onAnchorExit() 回调时，调用该接口。

```
public abstract void stopPlay(String userId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	对方的用户信息。
callback	ActionCallback	操作回调。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
public abstract void requestJoinAnchor(String reason, int timeout, TRTCLiveRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
reason	String	连麦原因。
timeout	int	超时时间。
callback	ActionCallback	主播响应回调。

主播和观众的连麦流程如下：

1. 观众调用 requestJoinAnchor() 向主播发起连麦请求。
2. 主播会收到 TRTCLiveRoomDelegate 的 onRequestJoinAnchor() 回调通知。
3. 主播调用 responseJoinAnchor() 决定是否接受来自观众的连麦请求。
4. 观众会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 观众如果请求被同意，则调用 startCameraPreview() 开启本地摄像头。
6. 观众然后调用 startPublish() 正式进入推流状态。
7. 主播一旦观众进入连麦状态，主播会收到 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知。
8. 主播调用 startPlay() 即可看到连麦观众的视频画面。
9. 观众如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 onAnchorEnter() 通知，调用 startPlay() 播放其他连麦

版权所有：腾讯云计算（北京）有限责任公司

者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 TRTCLiveRoomDelegate 的 onRequestJoinAnchor() 回调后需要调用此接口来处理观众的连麦请求。

```
public abstract void responseJoinAnchor(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	观众 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝连麦的原因描述。

kickoutJoinAnchor

主播踢除连麦观众。主播调用此接口踢除连麦观众后，被踢连麦观众会收到 TRTCLiveRoomDelegate 的 onKickoutJoinAnchor() 回调通知。

```
public abstract void kickoutJoinAnchor(String userId, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userId	String	连麦观众 ID。
callback	ActionCallback	操作回调。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
public abstract void requestRoomPK(int roomId, String userId, TRTCLiveRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。
callback	ActionCallback	请求跨房 PK 的结果回调。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 主播 A调用 requestRoomPK() 向主播 B 发起连麦请求。

2. 主播 B会收到 TRTCLiveRoomDelegate 的 onRequestRoomPK() 回调通知。
3. 主播 B调用 responseRoomPK() 决定是否接受主播 A 的 PK 请求。
4. 主播 B如果接受主播 A 的要求，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 来显示主播 A 的视频画面。
5. 主播 A会收到 responseCallback 回调通知，该通知会携带来自主播 B 的处理结果。
6. 主播 A如果请求被同意，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 requestRoomPK 传入的 responseCallback 回调。

```
public abstract void responseRoomPK(String userId, boolean agree, String reason);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	boolean	true：同意；false：拒绝。
reason	String	同意/拒绝 PK 的原因描述。

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 TRTCLiveRoomDelegate 的 onQuitRoomPk() 回调通知。

```
public abstract void quitRoomPK(TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
public abstract void switchCamera();
```

setMirror

设置是否镜像展示。

```
public abstract void setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	boolean	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音所有远端音频。

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager 。

```
public abstract TXBeautyManager getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“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”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message,
TRTCLiveRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

调试相关接口函数

showVideoDebugLog

是否在界面中展示debug信息。

```
public abstract void showVideoDebugLog(boolean isShow);
```

参数如下表所示：

参数	类型	含义
isShow	boolean	开启/关闭 Debug 信息显示。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义

message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

直播房间信息变更回调。多用于直播连麦、PK下房间状态变化通知场景。

```
void onRoomInfoChange(TRTCLiveRoomDef.TRTCLiveRoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	TRTCLiveRoomInfo	房间信息。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。

```
void onAnchorEnter(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。

```
void onAnchorExit(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	退房用户 ID。

onAudienceEnter

收到观众进房通知。

```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	进房观众信息。

onAudienceExit

收到观众退房通知。

```
void onAudienceExit(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	退房观众信息。

主播和观众连麦事件回调

onRequestJoinAnchor

主播收到观众连麦请求时的回调。

```
void onRequestJoinAnchor(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, String reason, int timeOut);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	请求连麦观众信息。
reason	String	连麦原因描述。
timeout	int	处理请求的超时时间，如果上层超过该时间没有处理，则会自动将该次请求废弃。

onKickoutJoinAnchor

连麦观众收到被踢出连麦的通知。连麦观众收到被主播踢除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

```
void onKickoutJoinAnchor();
```

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK ，您需要等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 来播放邀约主播的流。

```
void onRequestRoomPK(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo, int timeout);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveUserInfo	发起跨房连麦的主播信息。
timeout	int	处理请求的超时时间。

onQuitRoomPK

收到断开跨房 PK 通知。

```
void onQuitRoomPK();
```

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	TRTCLiveUserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	TRTCLiveUserInfo	发送者用户信息。

版权所有：腾讯云计算（北京）有限责任公司

TRTCAudioEffectManager

playBGM

播放背景音乐。

```
void playBGM(String url, int loopTimes, int bgmVol, int micVol, TRTCCloud.BGMNotify notify);
```

参数如下表所示：

参数	类型	含义
url	String	背景音乐文件路径。
loopTimes	int	循环次数
bgmVol	int	BGM 音量
micVol	int	采集音量
notify	TRTCCloud.BGMNotify	播放通知

stopBGM

停止播放背景音乐。

```
void stopBGM();
```

pauseBGM

暂停播放背景音乐。

```
void pauseBGM();
```

resumeBGM

继续播放背景音乐。

```
void resumeBGM();
```

setBGMVolume

设置背景音乐的音量大小，播放背景音乐混音时使用，用来控制背景音的音量大小。

```
void setBGMVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	音量大小，100表示正常音量，取值范围为0 – 100，默认值为100。

版权所有：腾讯云计算（北京）有限责任公司

setBGMPosition

设置背景音乐播放进度。

```
int setBGMPosition(int position);
```

参数如下表所示：

参数	类型	含义
position	int	背景音乐播放进度，单位为毫秒（ms）。

返回
0：成功。

setMicVolume

设置麦克风的音量大小，播放背景音乐混音时使用，用来控制麦克风音量大小。

```
void setMicVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	Int	音量大小，取值0 – 100，默认值为100。

setReverbType

设置混响效果。

```
void setReverbType(int reverbType);
```

参数如下表所示：

参数	类型	含义
reverbType	int	混响类型，详情请参见 TRTCCloudDef 中的 TRTC_REVERB_TYPE 定义。

setVoiceChangerType

设置变声类型。

```
void setVoiceChangerType(int type);
```

参数如下表所示：

参数	类型	含义
type	int	混响类型，详情请参见 TRTCCloudDef 中的 TRTC_VOICE_CHANGER_TYPE 定义。

playAudioEffect

播放音效，每个音效都需要您指定具体的 ID，您可以通过该 ID 对音效的开始、停止、音量等进行设置。支持 aac、mp3 以及 m4a 格式。

```
void playAudioEffect(int effectId, String path, int count, boolean publish, int volume);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。
path	String	音效路径。
count	int	循环次数。
publish	boolean	是否推送 / true 推送给观众, false 本地预览。
volume	int	音量大小，取值范围为0 – 100，默认值为100。

pauseAudioEffect

暂停音效播放。

```
void pauseAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

resumeAudioEffect

恢复音效播放。

```
void resumeAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

stopAudioEffect

停止音效播放。

```
void stopAudioEffect(int effectId);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。

版权所有：腾讯云计算（北京）有限责任公司

stopAllAudioEffects

停止全部音效播放。

```
void stopAllAudioEffects();
```

setAudioEffectVolume

设置音效音量。

```
void setAudioEffectVolume(int effectId, int volume);
```

参数如下表所示：

参数	类型	含义
effectId	int	音效 ID。
volume	int	音量大小，取值范围为0 – 100，默认值为100。

setAllAudioEffectsVolume

设置所有音效的音量。

```
void setAllAudioEffectsVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	音量大小，取值范围为0 – 100，默认值为100。

TRTCLiveRoom API (Flutter)

最近更新时间：2023-08-24 16:22:42

TRTCLiveRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的，支持以下功能：

- 主播创建新的直播间开播，观众进入直播间观看。
- 主播和观众进行视频连麦互动。
- 两个不同房间的主播 PK 互动。
- 支持发送各种文本消息和自定义消息，自定义消息可用于实现弹幕、点赞和礼物。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCLiveRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 视频连麦直播（Flutter）。

- TRTC SDK：使用 TRTC SDK 作为低延时直播组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现直播聊天室的功能，同时，通过 IM 消息串联主播间的连麦流程。

TRTCLiveRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
registerListener	设置事件回调。
unRegisterListener	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（主播调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（主播调用）。
enterRoom	进入房间（观众调用）。
exitRoom	离开房间（观众调用）。
getRoomInfos	获取房间列表的详细信息。
getAnchorList	获取房间内所有的主播列表，enterRoom() 成功后调用才有效。

版权所有：腾讯云计算（北京）有限责任公司

getRoomMemberList	获取房间内所有的成员信息，enterRoom() 成功后调用才有效。

推拉流相关接口函数

API	描述
startCameraPreview	开启本地视频的预览画面。
stopCameraPreview	停止本地视频采集及预览。
startPublish	开始直播（推流）。
stopPublish	停止直播（推流）。
startPlay	播放远端视频画面，可以在普通观看和连麦场景中调用。
stopPlay	停止渲染远端视频画面。

主播和观众连麦

API	描述
requestJoinAnchor	观众请求连麦。
responseJoinAnchor	主播处理连麦请求。
kickoutJoinAnchor	主播移除连麦观众。

主播跨房间 PK

API	描述
requestRoomPK	主播请求跨房 PK。
responseRoomPK	主播响应跨房 PK 请求。
quitRoomPK	退出跨房 PK。

音视频控制相关接口函数

API	描述
switchCamera	切换前后摄像头。
setMirror	设置是否镜像展示。
muteLocalAudio	静音本地音频。
muteRemoteAudio	静音远端音频。
muteAllRemoteAudio	静音所有远端音频。

背景音乐音效相关接口函数

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

版权所有：腾讯云计算（北京）有限责任公司

美颜滤镜相关接口函数

API	描述
getBeautyManager	获取美颜管理对象 TXBeautyManager 。

消息发送相关接口函数

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

TRTCLiveRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onKickedOffline	其他用户登录了同一账号，被移下线。

房间事件回调

API	描述
onEnterRoom	本地进房回调。
onUserVideoAvailable	远端用户是否存在可播放的主路画面（一般用于摄像头）。
onRoomDestroy	房间被销毁的回调。

主播和观众进出事件回调

API	描述
onAnchorEnter	收到新主播进房通知。
onAnchorExit	收到主播退房通知。
onAudienceEnter	收到观众进房通知。
onAudienceExit	收到观众退房通知。

主播和观众连麦事件回调

API	描述
onRequestJoinAnchor	主播收到观众连麦请求时的回调。
onAnchorAccepted	主播同意观众的连麦请求。
onAnchorRejected	主播拒绝观众的连麦请求。

版权所有：腾讯云计算（北京）有限责任公司

onKickoutJoinAnchor	连麦观众收到被移出连麦的通知。

主播 PK 事件回调

API	描述
onRequestRoomPK	收到请求跨房 PK 通知。
onRoomPKAccepted	主播接受跨房 PK 请求。
onRoomPKRejected	主播拒绝跨房 PK 请求。
onQuitRoomPK	收到断开跨房 PK 通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

SDK 基础函数

sharedInstance

获取 TRTCLiveRoom 单例对象。

```
static Future<TRTCLiveRoom> sharedInstance()
```

destroySharedInstance

销毁 TRTCLiveRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCLiveRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
static void destroySharedInstance()
```

registerListener

TRTCLiveRoom 事件回调，您可以通过 TRTCLiveRoomDelegate 获得 TRTCLiveRoom 的各种状态通知。

```
void registerListener(VoiceListenerFunc func);
```

> **说明**
> registerListener 是 TRTCLiveRoom 的代理回调。

unRegisterListener

移除组件事件监听接口。

```
void unRegisterListener(VoiceListenerFunc func);
```

login

登录。

```
Future<ActionCallback> login(
    int sdkAppId, String userId, String userSig, TRTCLiveRoomConfig config);
```

参数如下表所示：

参数	类型	含义
sdkAppID	Int	您可以在实时音视频控制台 >应用管理> 应用信息中查看 SDKAppID。
userID	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
config	TRTCLiveRoomConfig	全局配置信息，请在登录时初始化，登录之后不可变更。 • useCDNFirst 属性：用于设置观众观看方式。true 表示普通观众通过 CDN 观看，计费便宜但延时较高。false 表示普通观众通过低延时观看，计费价格介于 CDN 和连麦之间，但延迟可控制在1s以内。 • CDNPlayDomain 属性：在 useCDNFirst 设置为 true 时才会生效，用于指定 CDN 观看的播放域名，您可以登录直播控制台 > 域名管理 页面中进行设置。

logout

登出。

```
Future<ActionCallback> logout();
```

setSelfProfile

修改个人信息。

```
Future<ActionCallback> setSelfProfile(String userName, String avatarURL);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。

房间相关接口函数

createRoom

创建房间（主播调用）。

```
Future<ActionCallback> createRoom(int roomId, TRTCCreateRoomParam roomParam);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个直播间列表，腾讯云暂不提供直播间列表的管理服务，请自行管理您的直播间列表。
roomParam	RoomParam	房间信息，用于房间描述的信息，例如房间名称，封面信息等。如果房间列表和房间信息都由您的服务器自行管理，可忽略该参数。

主播开播的正常调用流程如下：

1. **主播** 调用 `startCameraPreview()` 打开摄像头预览，此时可以调整美颜参数。
2. **主播** 调用 `createRoom()` 创建直播间，房间创建成功与否会通过 ActionCallback 通知给主播。
3. **主播** 调用 `starPublish()` 开始推流。

destroyRoom

销毁房间（主播调用）。主播在创建房间后，可以调用该函数来销毁房间。

```
Future<ActionCallback> destroyRoom();
```

enterRoom

进入房间（观众调用）。

```
Future<ActionCallback> enterRoom(int roomId);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。

观众观看直播的正常调用流程如下：

1. **观众** 向您的服务端获取最新的直播间列表，可能包含多个直播间的 roomID 和房间信息。
2. **观众** 观众选择一个直播间，并调用 `enterRoom()` 进入该房间。
3. **观众** 调用 `startPlay(userId)` 并传入主播的 userId 开始播放。
   - 若直播间列表已包含主播端的 userId 信息，观众端可直接调用 `startPlay(userId)` 即可开始播放。
   - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放。

exitRoom

离开房间。

版权所有：腾讯云计算（北京）有限责任公司

```
Future<ActionCallback> exitRoom();
```

getRoomInfos

获取房间列表的详细信息，房间信息是主播在创建 `createRoom()` 时通过 roomInfo 设置的。

> **说明**
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
Future<RoomInfoCallback> getRoomInfos(List<String> roomIdList);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<String>	房间号列表。

getAnchorList

获取房间内所有的主播列表， `enterRoom()` 成功后调用才有效。

```
Future<UserListCallback> getAnchorList();
```

getRoomMemberList

获取房间内所有的观众信息， `enterRoom()` 成功后调用才有效。

```
Future<UserListCallback> getRoomMemberList(int nextSeq)
```

参数如下表所示：

参数	类型	含义
nextSeq	int	分页拉取标志，第一次拉取填0，回调成功如果 nextSeq 不为零，需要分页，传入再次拉取，直至为0。

推拉流相关接口函数

startCameraPreview

开启本地视频的预览画面。

```
Future<void> startCameraPreview(bool isFrontCamera, int viewId);
```

参数如下表所示：

参数	类型	含义
isFrontCamera	bool	true：前置摄像头；false：后置摄像头。

viewId	int	视频 view 的回调 ID。

stopCameraPreview

停止本地视频采集及预览。

```
Future<void> stopCameraPreview();
```

startPublish

开始直播（推流），适用于以下场景：

- 主播开播的时候调用。
- 观众开始连麦时调用。

```
Future<void> startPublish(String streamId);
```

参数如下表所示：

参数	类型	含义
streamId	String?	用于绑定直播 CDN 的 streamId，如果您希望观众通过直播 CDN 进行观看，需要指定当前主播的直播 streamId。

stopPublish

停止直播（推流），适用于以下场景：

- 主播结束直播时调用。
- 观众结束连麦时调用。

```
Future<void> stopPublish();
```

startPlay

播放远端视频画面，可以在普通观看和连麦场景中调用。

```
Future<void> startPlay(String userId, int viewId);
```

参数如下表所示：

参数	类型	含义
userId	String	需要观看的用户 ID。
viewId	int	视频view的回调 ID。

- **普通观看场景：**
  - 若直播间列表已包含主播端的 userId 信息，观众端可以直接在 `enterRoom()` 成功后调用 `startPlay(userId)` 播放主播的画面。
  - 若在进房前暂未获取主播的 userId，观众端在进房后会收到 `TRTCLiveRoomDelegate` 中的 `onAnchorEnter(userId)` 的事件回调，该回调中携带主播的 userId 信息，再调用 `startPlay(userId)` 即可播放主播的画面。
- **直播连麦场景：**

发起连麦后，主播会收到来自 TRTCLiveRoomDelegate 中的 onAnchorEnter(userId) 回调，此时使用回调中的 userId 调用 startPlay(userId) 即可播放连麦画面。

stopPlay

停止渲染远端视频画面。需在 onAnchorExit() 回调时，调用该接口。

```
Future<void> stopPlay(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	对方的用户信息。

主播和观众连麦

requestJoinAnchor

观众请求连麦。

```
Future<ActionCallback> requestJoinAnchor();
```

主播和观众的连麦流程如下：

1. 观众调用 requestJoinAnchor() 向主播发起连麦请求。
2. 主播会收到 TRTCLiveRoomDelegate 的 onRequestJoinAnchor() 回调通知。
3. 主播调用 responseJoinAnchor() 决定是否接受来自观众的连麦请求。
4. 观众会收到 responseCallback 回调通知，该通知会携带主播的处理结果。
5. 观众如果请求被同意，则调用 startCameraPreview() 开启本地摄像头。
6. 观众然后调用 startPublish() 正式进入推流状态。
7. 主播一旦观众进入连麦状态，主播会收到 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知。
8. 主播调用 startPlay() 即可看到连麦观众的视频画面。
9. 观众如果直播间里已有其他观众正在跟主播连麦，新加入的连麦观众会收到 onAnchorEnter() 通知，调用 startPlay() 播放其他连麦者的视频画面。

responseJoinAnchor

主播处理连麦请求。主播在收到 TRTCLiveRoomDelegate 的 onRequestJoinAnchor() 回调后需要调用此接口来处理观众的连麦请求。

```
Future<ActionCallback> responseJoinAnchor(String userId, boolean agreee);
```

参数如下表所示：

参数	类型	含义
userId	String	观众 ID。
agree	bool	true：同意；false：拒绝。

kickoutJoinAnchor

主播移除连麦观众。主播调用此接口移除连麦观众后，被移连麦观众会收到 TRTCLiveRoomDelegate 的 onKickoutJoinAnchor() 回调通知。

```
Future<ActionCallback> kickoutJoinAnchor(String userId);
```

参数如下表所示：

参数	类型	含义
userId	String	连麦观众 ID。

主播跨房间 PK

requestRoomPK

主播请求跨房 PK。

```
Future<ActionCallback> requestRoomPK(int roomId, String userId);
```

参数如下表所示：

参数	类型	含义
roomId	int	被邀约房间 ID。
userId	String	被邀约主播 ID。

主播和主播之间可以跨房间 PK，两个正在直播中的主播 A 和 B 之间的跨房 PK 流程如下：

1. 主播 A调用 requestRoomPK() 向主播 B 发起连麦请求。
2. 主播 B会收到 TRTCLiveRoomDelegate 的 onRequestRoomPK() 回调通知。
3. 主播 B调用 responseRoomPK() 决定是否接受主播 A 的 PK 请求。
4. 主播 B如果接受主播 A 的要求，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 来显示主播 A 的视频画面。
5. 主播 A会收到 onRoomPKAccepted 或 onRoomPKRejected 回调通知。
6. 主播 A如果请求被同意，等待 TRTCLiveRoomDelegate 的 onAnchorEnter() 通知，然后调用 startPlay() 显示主播 B 的视频画面。

responseRoomPK

主播响应跨房 PK 请求。主播响应后，对方主播会收到 requestRoomPK 传入的 responseCallback 回调。

```
Future<ActionCallback> responseRoomPK(String userId, boolean agree);
```

参数如下表所示：

参数	类型	含义
userId	String	发起 PK 请求的主播 ID。
agree	bool	true：同意；false：拒绝。

版权所有：腾讯云计算（北京）有限责任公司

quitRoomPK

退出跨房 PK。PK 中的任何一个主播退出跨房 PK 状态后，另一个主播会收到 `TRTCLiveRoomDelegate` 的 `onQuitRoomPk()` 回调通知。

```
Future<ActionCallback> quitRoomPK();
```

音视频控制相关接口函数

switchCamera

切换前后摄像头。

```
Future<void> switchCamera(boolean isFrontCamera);
```

setMirror

设置是否镜像展示。

```
Future<void> setMirror(boolean isMirror);
```

参数如下表所示：

参数	类型	含义
isMirror	bool	开启/关闭镜像。

muteLocalAudio

静音本地音频。

```
Future<void> muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

muteRemoteAudio

静音远端音频。

```
Future<void> muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	远端的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

版权所有：腾讯云计算（北京）有限责任公司

muteAllRemoteAudio

静音所有远端音频。

```
Future<void> muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
getAudioEffectManager();
```

美颜滤镜相关接口函数

getBeautyManager

获取美颜管理对象 TXBeautyManager 。

```
getBeautyManager();
```

通过美颜管理，您可以使用以下功能：

- 设置“美颜风格”、“美白”、“红润”、“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”等美容效果。
- 调整“发际线”、“眼间距”、“眼角”、“嘴形”、“鼻翼”、“鼻子位置”、“嘴唇厚度”、“脸型”。
- 设置人脸挂件（素材）等动态效果。
- 添加美妆。
- 进行手势识别。

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
Future<ActionCallback> sendRoomTextMsg(String message);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。

sendRoomCustomMsg

发送自定义文本消息。

```
Future<ActionCallback> sendRoomCustomMsg(String cmd, String message);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

TRTCLiveRoomDelegate事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

参数如下表所示：

参数	类型	含义
errCode	int	错误码。
errMsg	String	错误信息。

onWarning

警告回调。
参数如下表所示：

参数	类型	含义
warningCode	int	错误码。
warningMsg	String	警告信息。

onKickedOffline

其他用户登录了同一账号，被移下线。

房间事件回调

onRoomDestroy

房间被销毁的回调。主播退房时，房间内的所有用户都会收到此通知。

onEnterRoom

本地进房。
参数如下表所示：

参数	类型	含义
result	int	result > 0 时为进房耗时（ms），result < 0 时为进房错误码。

onUserVideoAvailable

远端用户是否存在可播放的主路画面（一般用于摄像头）。
参数如下表所示：

参数	类型	含义
userId	String	用户标识。
available	boolean	画面是否开启。

主播和观众进出事件回调

onAnchorEnter

收到新主播进房通知。连麦观众和跨房 PK 主播进房后观众会收到新主播的进房事件，您可以调用 `TRTCLiveRoom` 的 `startPlay()` 显示该主播的视频画面。
参数如下表所示：

参数	类型	含义
userId	String	新进房主播 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAnchorExit

收到主播退房通知。房间内的主播（和连麦中的观众）会收到新主播的退房事件，您可以调用 `TRTCLiveRoom` 的 `stopPlay()` 关闭该主播的视频画面。
参数如下表所示：

参数	类型	含义
userId	String	退出主播 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAudienceEnter

收到观众进房通知。

```
void onAudienceEnter(TRTCLiveRoomDef.TRTCLiveUserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	TRTCLiveRoomDef.TRTCLiveUserInfo	进房观众用户 ID、昵称、头像等信息。

onAudienceExit

收到观众退房通知。
参数如下表所示：

参数	类型	含义
userId	String	退房观众信息。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onRequestJoinAnchor

主播收到观众连麦请求时的回调。
参数如下表所示：

参数	类型	含义
userId	String	请求连麦用户 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onAnchorAccepted

主播同意观众的连麦请求。
参数如下表所示：

参数	类型	含义
userId	String	主播的用户 ID。

onAnchorRejected

主播拒绝观众的连麦请求。
参数如下表所示：

参数	类型	含义
userId	String	主播的用户 ID。

onKickoutJoinAnchor

连麦观众收到被移出连麦的通知。连麦观众收到被主播移除连麦的消息，您需要调用 `TRTCLiveRoom` 的 `stopPublish()` 退出连麦。

主播 PK 事件回调

onRequestRoomPK

收到请求跨房 PK 通知。主播收到其他房间主播的 PK 请求，如果同意 PK ，您需要等待 `TRTCLiveRoomDelegate` 的 `onAnchorEnter()` 通知，然后调用 `startPlay()` 来播放邀约主播的流。
参数如下表所示：

参数	类型	含义
userId	String	请求跨房用户 ID。
userName	String	用户昵称。
userAvatar	String	用户头像地址。

onRoomPKAccepted

主播接受跨房 PK 请求。
参数如下表所示：

参数	类型	含义
userId	String	接收跨房 PK 的用户 ID。

onRoomPKRejected

主播接受跨房 PK 请求。
参数如下表所示：

参数	类型	含义
userId	String	拒绝跨房 PK 的用户 ID。

onQuitRoomPK

收到断开跨房 PK 通知。

消息事件回调

onRecvRoomTextMsg

收到文本消息。
参数如下表所示：

参数	类型	含义
message	String	文本消息。

onRecvRoomCustomMsg

收到自定义消息。
参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。

版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司

在线 K 歌
组件介绍（TUIKaraoke）

最近更新时间：2022-10-31 16:21:59

组件背景

艾媒咨询数据显示，2021年中国在线 K 歌用户规模约为5.1亿人，渗透率约为49.7%。在线 K 歌具有更强的沉浸感体验，多元的玩法也迎合了不同用户群体的个性化需求，目前已经成为了在线泛娱乐领域的主要项目之一。在网络技术革新的基础上，在线 K 歌 App 不断推出多元化的歌唱模式和玩法，不断丰富的功能提升了在线 K 歌 App 的实用性和可玩性。下来将介绍腾讯实时音视频（Tencent RTC）技术在线 K 歌场景下的组件实现方案。

TUIKaraoke组件介绍

TUIKaraoke 是一个开源的音视频 UI 组件，通过在项目中集成 TUIKaraoke 组件，您只需要编写几行代码就可以为您的应用添加在线 K 歌场景，体验 K 歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。TUIKaraoke 同时支持 Android、iOS 平台的源代码，基本功能如下图所示：



说明：

TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信 IM 服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明 ，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支

持100个 DAU。

快速集成（TUIKaraoke）Android

最近更新时间：2022-10-31 16:18:11

步骤一：下载并导入TUIKaraoke 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 Android 目录下的 Source 和 Debug 目录到您的工程中，并完成如下导入动作：

1. 在 setting.gradle 中完成导入，参考如下：

```
include ':tuikaraoke'
include ':debug'
```

2. 在 app 的 build.gradle 文件中添加对 TUIKaraoke 的依赖：

```
api project(':tuikaraoke')
```

3. 在根目录的 build.gradle 文件中添加 TRTC SDK 和 IM SDK 的依赖：

```
ext {
  liteavSdk = "com.tencent.liteav:LiteAVSDK_TRTC:latest.release"
  imSdk = "com.tencent.imsdk:imsdk-plus:latest.release"
}
```

步骤二：配置权限及混淆规则

在 AndroidManifest.xml 中配置 App 的权限，SDK 需要以下权限（6.0以上的 Android 系统需要动态申请麦克风、读取存储权限等）：

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />      // 使用场景：悬浮窗功能需要此权限；
<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" />              // 使用场景：使用蓝牙耳机时需要此权限；
```

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

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

步骤三：初始化并登录

相关接口详情请参见 TUIKaraoke。

```
// 1.初始化
```

```
TRTCKaraokeRoom mTRTCKaraokeRoom = TRTCKaraokeRoom.sharedInstance(this);
mTRTCKaraokeRoom.setDelegate(this);
// 2.登录
mTRTCKaraokeRoom.login(SDKAppID, UserID, UserSig, new TRTCKaraokeRoomCallback.ActionCallback() {
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
        //登录成功
        }
    }
});
```

参数说明：

- **SDKAppID**：TRTC 应用ID，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**。SDKAppID 信息如下图所示：



- **Secretkey**：TRTC 应用密钥，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUIKaraoke 示例工程 自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现在线KTV场景

1. 主播创建房间 TUIKaraoke.createRoom

```
int roomId = "房间ID";
TRTCKaraokeRoomDef.RoomParam roomParam = new TRTCKaraokeRoomDef.RoomParam();
roomParam.roomName = "房间名称";
roomParam.needRequest = false; // 上麦是否需要房主确认
roomParam.seatCount = 8;      // 房间座位数，一共8个座位
```

```
roomParam.coverUrl = "房间封面图URL";
mTRTCKaraokeRoom.createRoom(roomId, roomParam, new TRTCKaraokeRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
  if (code == 0) {
   //创建成功
  }
 }
});
```

2. 听众进入房间 TUIKaraoke.enterRoom

```
mTRTCKaraokeRoom.enterRoom(roomId, new TRTCKaraokeRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
    //进房成功
    }
 }
});
```

3. 听众主动上麦 TUIKaraoke.enterSeat

```
// 1.听众调用上麦
int seatIndex = 1;
mTRTCKaraokeRoom.enterSeat(seatIndex, new TRTCKaraokeRoomCallback.ActionCallback() {
 @Override
 public void onCallback(int code, String msg) {
    if (code == 0) {
    //上麦成功
    }
 }
});
// 2.收到 onSeatListChange 回调，刷新麦位列表
@Override
public void onSeatListChange(final List<TRTCKaraokeRoomDef.SeatInfo> seatInfoList) {
}
```

> **说明：**
> 其他关于麦位管理的相关操作，您可参考 TUIKaraoke 接口文档 按需实现，或者可以参考我们的 TUIKaraoke示例工程。

4. 实现音乐播放并体验 KTV 场景

您可以根据自己的业务获取音乐 ID 和 URL链接播放歌曲，接口详情请参见 TUIKaraoke 音乐播放接口。

```
//播放音乐
mTRTCKaraokeRoom.startPlayMusic(musicID,url);
//停止音乐
mTRTCKaraokeRoom.stopPlayMusic();
```

完成以上步骤，您可以实现 KTV 基本功能。如果您的业务还需要文字聊天、发送礼物等功能，可以接入以下能力。

步骤五：文字聊天功能（可选）

如果您需要实现各主播或听众之间文字聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口详情请参见 TRTCKaraokeRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
mTRTCKaraokeRoom.sendRoomTextMsg("Hello Word!", new TRTCKaraokeRoomCallback.ActionCallback() {
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
            //发送成功
        }
    }
});
// 接收端：监听文本消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
  @Override
  public void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo) {
      Log.d(TAG, "收到来自" + userInfo.userName + "的消息:" + message);
  }
});
```

步骤六：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送或接收礼物并展示。

```
// 发送端：通过自定义 "CMD_GIFT" 来区分礼物消息
mTRTCKaraokeRoom.sendRoomCustomMsg("CMD_GIFT",date, new
TRTCKaraokeRoomCallback.ActionCallback() {
    @Override
    public void onCallback(int code, String msg) {
        if (code == 0) {
            //发送成功
        }
    }
});

// 接收端：监听礼物消息
mTRTCKaraokeRoom.setDelegate(new TRTCKaraokeRoomDelegate() {
    @Override
    public void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInfo userInfo)
{
        if ("CMD_GIFT".equals(cmd)) {
            // 收到礼物消息
            Log.d(TAG, "收到来自" + userInfo.userName + "的礼物:" + message);
        }
    }
});
```

iOS

最近更新时间：2022-10-31 16:18:11

步骤一：下载并导入 TUIKaraoke 组件

单击进入 Github ，选择克隆/下载代码，然后拷贝 iOS目录下的 `Source` 、 `Resources` 、 `TXAppBasic` 文件夹和 `TUIKaraoke.podspec` 文件到您的工程中，并完成如下导入动作：

1. 在您的 `Podfile` 文件内添加导入命令，参考如下：

```
pod 'TUIKaraoke', :path => "./", :subspecs => ["TRTC"]
pod 'TXLiteAVSDK_TRTC'
pod 'TXAppBasic', :path => "TXAppBasic/"
```

2. 打开终端，进入 Podfile 文件所在目录下执行安装命令，参考如下：

```
pod install
```

步骤二：配置权限

在您的工程的 info.plist文件中配置 App 的权限，SDK 需要以下权限（iOS 系统需要动态申请麦克风）：

```
<key>NSMicrophoneUsageDescription</key>
<string>Karaoke 需要访问您的麦克风权限</string>
```

步骤三：初始化并登录

相关接口说明请参见 TUIKaraoke。

```
// 1.初始化
let karaokeRoom = TRTCKaraokeRoom.shared()
karaokeRoom.setDelegate(delegate: self)
// 2.登录
karaokeRoom.login(SDKAppID: Int32(SDKAppID), UserId: UserId, UserSig:
ProfileManager.shared.curUserSig()) { code, message in
    if code == 0 {
        //登录成功
    }
}
```

参数说明：

- **SDKAppID**：TRTC 应用ID，如果您未开通腾讯云 TRTC 服务，可进入 腾讯云实时音视频控制台，创建一个新的 TRTC 应用后，单击**应用信息**。SDKAppID 信息如下图所示：



- **Secretkey**：TRTC 应用密钥，和 SDKAppId 对应，进入 TRTC 应用管理 后，SecretKey 信息如上图所示。
- **userId**：当前用户的 ID，字符串类型，长度不超过32字节，不支持使用特殊字符，建议使用英文或数字，可结合业务实际账号体系自行设置。
- **userSig**：根据 SDKAppId、userId，Secretkey 等信息计算得到的安全保护签名，您可以单击 这里 直接在线生成一个调试的 UserSig，也可以参照我们的 TUICalling示例工程自行计算，更多信息见 如何计算及使用 UserSig。

步骤四：实现在线KTV场景

1. 主播创建房间 TUIKaraoke.createRoom

```
int roomId = "房间ID";
let param = RoomParam.init()
param.roomName = "房间名称";
param.needRequest = false; // 上麦是否需要房主确认
param.seatCount = 8;      // 房间座位数，一共8个座位
param.coverUrl = "房间封面图URL";

karaokeRoom.createRoom(roomID: Int32(roomInfo.roomID), roomParam: param) { [weak self] (code,
message) in
 guard let `self` = self else { return }
 if code == 0 {
    //创建成功
 }
}
```

2. 听众进入房间 TUIKaraoke.enterRoom

```
karaokeRoom.enterRoom(roomID: roomInfo.roomID) { [weak self] (code, message) in
 guard let `self` = self else { return }
 if code == 0 {
```

```
    //进房成功
 }
}
```

3. 听众主动上麦 TUIKaraoke.enterSeat

```
// 1.听众调用上麦
int seatIndex = 1;
karaokeRoom.enterSeat(seatIndex: seatIndex) { [weak self] (code, message) in
 guard let `self` = self else { return }
 if code == 0 {
    //上麦成功
 }
}
// 2.收到 onSeatListChange 回调，刷新麦位列表
func onSeatListChange(seatInfoList: [SeatInfo]) {
}
```

> **说明：**
> 其他关于麦位管理的相关操作，您可参考 TUIKaraoke 接口文档 按需实现，或者可以参考我们的 TUIKaraoke示例工程。

4. 实现音乐播放并体验 KTV 场景
您可以根据自己的业务获取音乐 ID 和 URL 链接，播放歌曲。详情请参见 TUIKaraoke 音乐播放接口。

```
//播放音乐
karaokeRoom.startPlayMusic(musicID: musicID, originalUrl: muscicLocalPath, accompanyUrl:
accompanyLocalPath);
//停止音乐
karaokeRoom.stopPlayMusic();
```

完成以上步骤，就可以实现 KTV 基本功能。如果您的业务还需要聊天、发送礼物等功能，可以接入以下能力。

步骤五：文字聊天功能（可选）

如果您需要实现各主播或听众之间文字聊天的功能，可以通过以下方法发送或接收聊天信息。
相关接口说明请参见 TRTCKaraokeRoom.sendRoomTextMsg。

```
// 发送端：发送文本消息
karaokeRoom.sendRoomTextMsg(message: message) { [weak self] (code, message) in
    if code == 0 {
        //发送成功
    }
}
// 接收端：监听文本消息
karaokeRoom.setDelegate(delegate: self)
func onRecvRoomTextMsg(message: String, userInfo: UserInfo) {
    debugPrint("收到来自：" + userInfo.userName + "的消息：" + message)
}
```

步骤六：礼物发送功能（可选）

如果您需要实现礼物发送和接收功能，可以通过以下方法发送礼物或接收礼物并展示。

```
// 发送端：通过自定义 "IMCMD_GIFT" 来区分礼物消息
karaokeRoom.sendRoomCustomMsg(cmd: kSendGiftCmd, message: message) { code, msg in
    if (code == 0) {
        //发送成功
    }
}


// 接收端：监听礼物消息
karaokeRoom.setDelegate(delegate: self)
func onRecvRoomCustomMsg(cmd: String, message: String, userInfo: UserInfo) {
    if cmd == kSendGiftCmd {
        debugPrint("收到来自：" + userInfo.userName + "的礼物：" + message)
    }
}
```

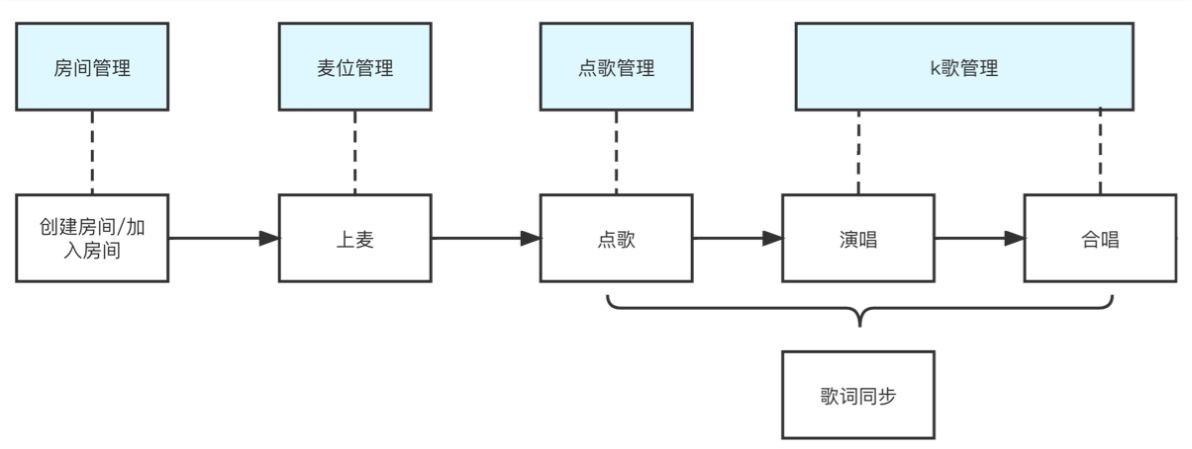
方案介绍（TUIKaraoke）
实现步骤

最近更新时间：2023-05-25 15:25:45

通常实现一个完整的在线 K 歌场景，需要涉及多个功能模块：房间管理、麦位管理、点歌管理、K 歌管理等。每个功能模块下的关键动作及功能点如下表所示。接下来会逐个介绍各个功能模块，通过介绍可以对搭建K歌房所需的功能有个完整的认知。

房间管理	麦位管理	点歌管理	K歌管理
房间列表	上/下麦	歌单展示	唱歌玩法
创建房间	麦位控制	搜索歌曲	切歌
加入房间	锁麦位	点歌	人声音量调节
退出房间	邀请上麦	歌曲置顶	混响/声效
销毁房间	麦位禁言	已点列表	歌词同步

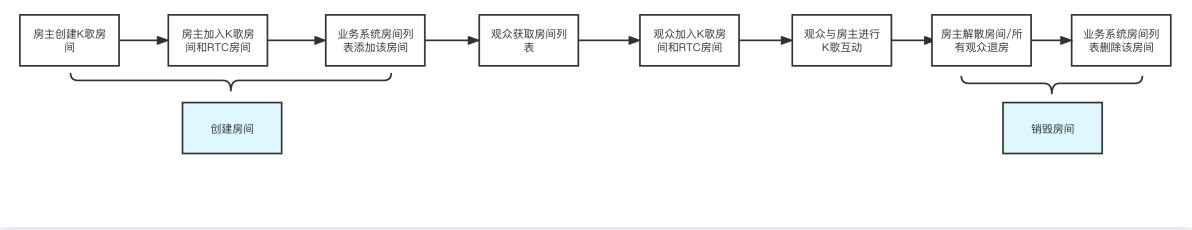
房主创建 K 歌房，用户可以选择感兴趣的歌房加入。进入房间后用户可以上麦参与互动，上麦后可以和房主语音互动。当然，用户也可以选择直接上麦参与合唱，这是两种不同的 K 歌玩法。在线 K 歌场景整体的业务流程如下图所示。



房间管理

房间管理是主要负责对房间列表的维护。主要包含的功能：创建房间、加入房间、销毁房间、退出房间。而且 K 歌房间有区别于普通房间，需要有单独的 K 歌房间标识符，以启动相关的组件管理：点歌管理、K 歌管理等功能。

- **创建房间**：用户登录业务系统后，可以创建房间，创建房间后房间列表要做新增操作。
- **销毁房间**：所有用户退出房间后，需要销毁房间，销毁房间后房间列表要做删除操作。



说明：

房间管理是实现在线K歌的必要模块，但并非主要功能模块，具体实现可以结合业务系统和 TRTC SDK，详见语聊房场景接入方案。

麦位管理

歌房内的麦位一般都是有序且有限的。麦位管理主要负责根据业务场景定义房间内的麦位数量，以及当前房间所有麦位的状态管理。麦位管理主要包含的功能：上/下麦、锁麦位、邀请上麦、麦位禁言等。

- 用户进入房间后，只有空闲状态的麦位才可以申请上麦。
- 房主同意用户上麦后，需要修改麦位状态为非空闲状态。
- 用户停止推流下麦后，需要重置麦位状态。
- 房主有权锁定麦位、邀请上麦、强制下麦、麦上禁言等。

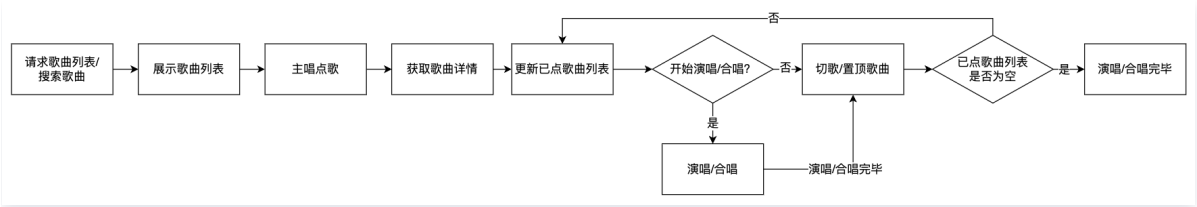
说明：
麦位管理是实现在线K歌的必要模块，但并非主要功能模块，具体实现可以结合业务系统和 IM SDK，详见语聊房场景接入方案。

点歌管理

基本介绍

点歌管理属于在线 K 歌场景比较重要的一环，主要包含的功能：歌单展示、搜索歌曲、点歌和排麦、已点列表。而且每个 K 歌房间要有一个维护已点歌曲列表和自动排麦功能，都是需要业务后台来实现，而歌单展示、搜索歌曲需要结合 音速达直播音乐版权引擎（Yinsuda Authorized Music for Live Streaming）实现。

实现流程



整个点歌管理中，主要涉及业务端 App、业务后台、音速达后台，其中各自的职能分别为：

- **业务端 App：**
  - 调用点歌接口上报歌曲信息.
  - 调用切歌接口通知业务后台更新已点歌单列表.
  - 调用演唱确认接口通知业务后台。
- **业务后台：**
  - 维护已点歌单列表。
  - 下发通知告诉业务端 App 更新当前已点歌曲列表。
- **音速达后台：**
  - 提供获取直播互动歌曲推荐 歌单列表/歌单详情 的接口。
  - 提供 获取直播互动曲目详情 的接口（播放playToken、歌词下载URL）。
  - 提供 搜索直播互动歌曲 的接口。

K 歌管理

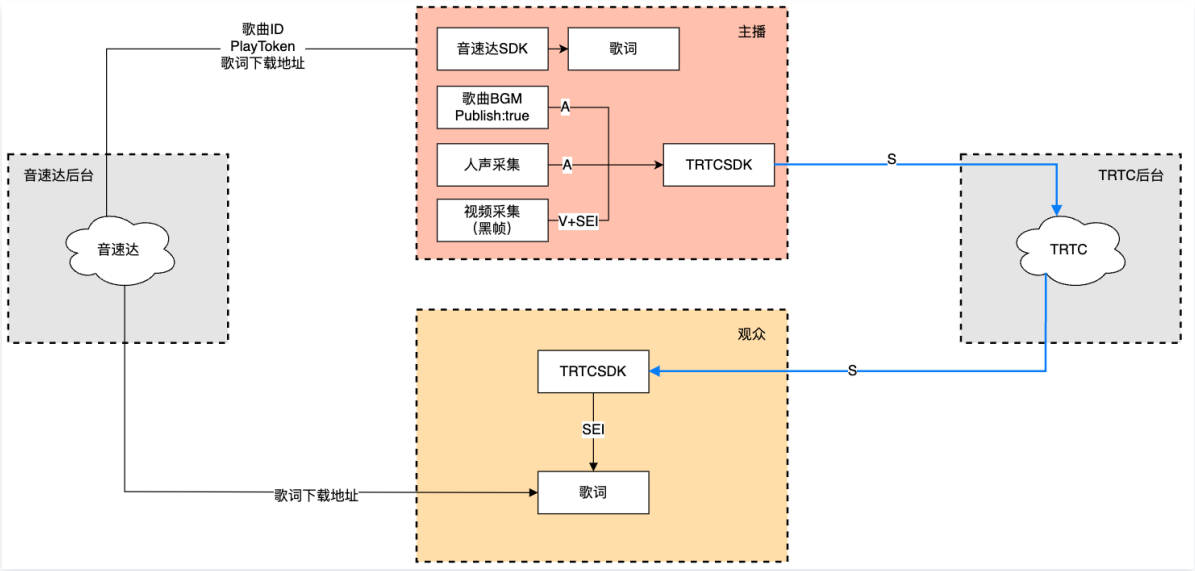
K 歌系统主要包含功能：唱歌玩法、开始/停止/切歌、人声音量调节、混响/声效、歌词同步。下面我们将通过演唱和实时合唱两种典型的 K 歌玩法来详细介绍 K 歌管理模块的实现流程。

演唱

主要是多人互动的KTV场景下，主播上麦后，才可以进行点歌，主播点歌成功后，会统一在点歌台展示，主播选择开始演唱。

（1）方案架构

在整体方案上，主要是用到的音速达SDK，实现歌曲下载，音速达后台，获取歌曲 playToken 和歌词下载地址；用到的 TRTC SDK，来实现演唱者人声的推流、歌曲的播放以及推流。整体的方案架构如下：



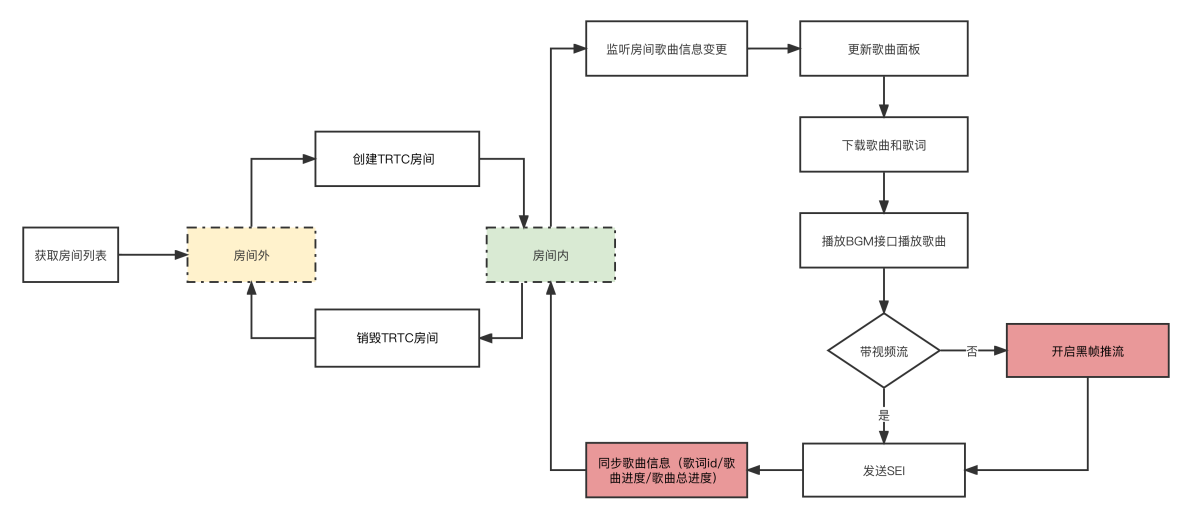
（2）具体实现

在演唱的场景，会区分不同角色有不同的实现流程，分为演唱者、观众 2种角色。

角色	描述	区别
演唱者	KTV 房间的演唱者，是房间的房主点歌以后演变而来的，房主创建房间后自动上麦点歌并演唱，退房后房间自动解散，已点歌曲自动清空	• 角色必须为主播 • 上行音视频（无视频上行黑帧） • 播放 BGM • 发送 SEI 信息（发送歌词信息） • 点歌
观众	KTV 房间的观众，播放演唱者的流	• 角色为观众，亦可上麦成为主播 • 下行音视频流 • 接收 SEI 信息（接收歌词信息）

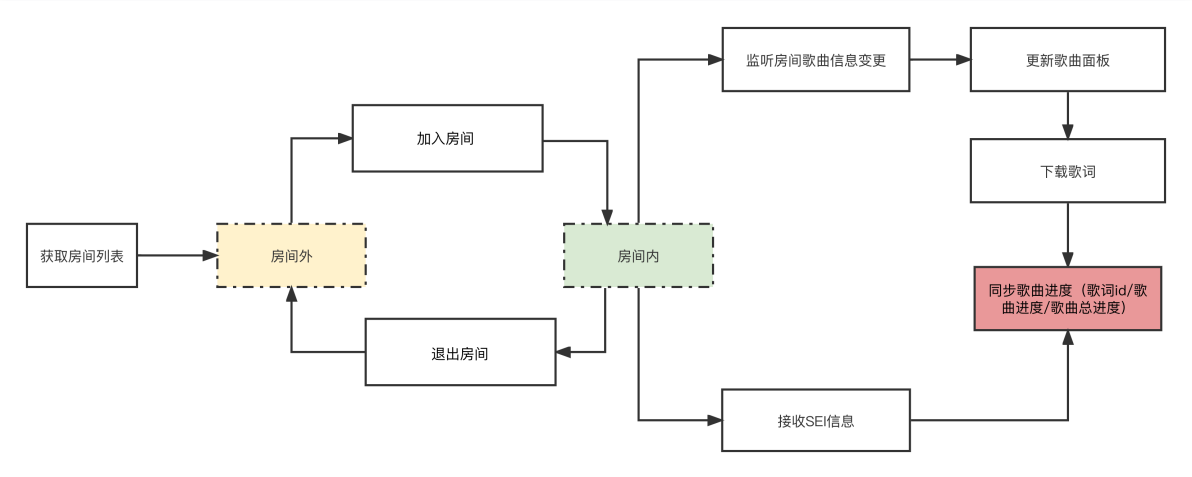
不同角色的基本实现流程如下：

房主



- 房主创建并加入 TRTC 的房间，自动上麦后点歌成为演唱者。
- 点歌后进行歌曲/歌词的下载，然后通过播放 BGM 接口播放歌曲。
- 演唱者没有带视频上行的话，需要开启视频上行。
- 通过 SEI 信息同步所有人的歌词进度。
- 演唱者在唱歌过程中可以随时切歌，并重新开始对歌曲/歌词进行下载，下载完成后进行演唱。
- 房主退房后将解散 TRTC 房间。

观众



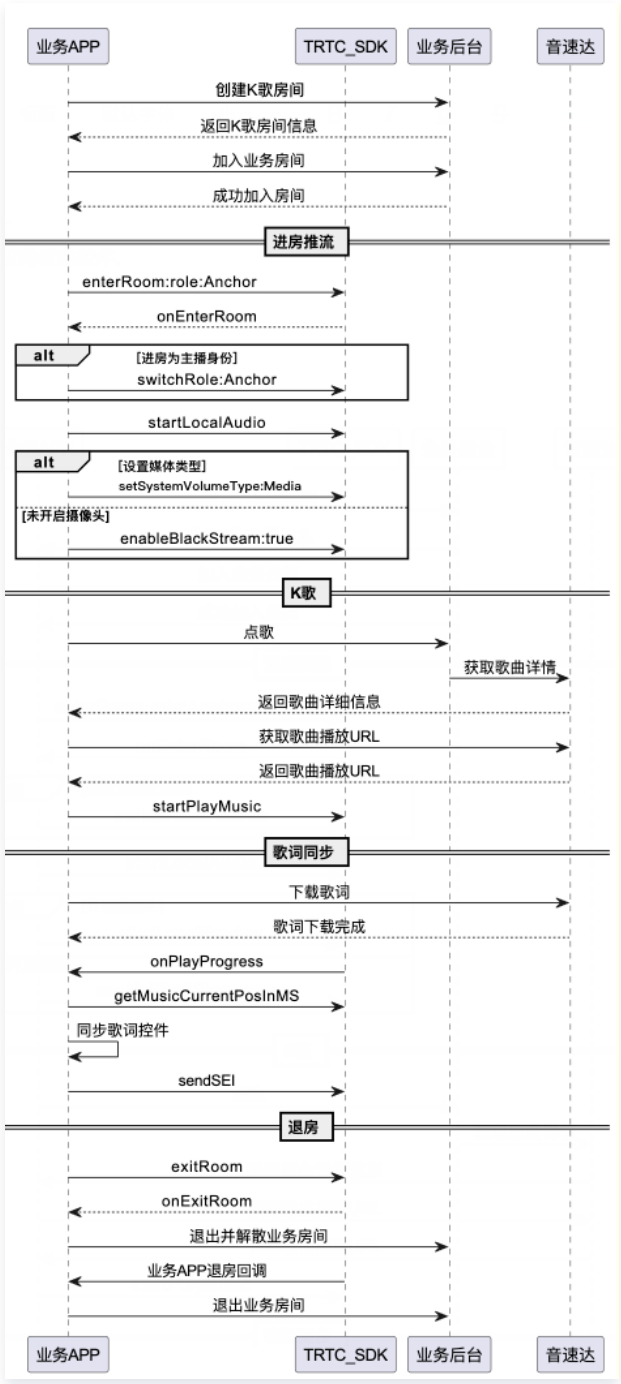
- 观众加入 TRTC 的房间。
- 监听房间歌曲变化，并加载歌词。
- 拉取演唱者的流。
- 解析演唱者发送的 SEI 信息，并同步歌词。

主要是要监听歌曲的 SEI 信息，更新对应的歌曲控件。

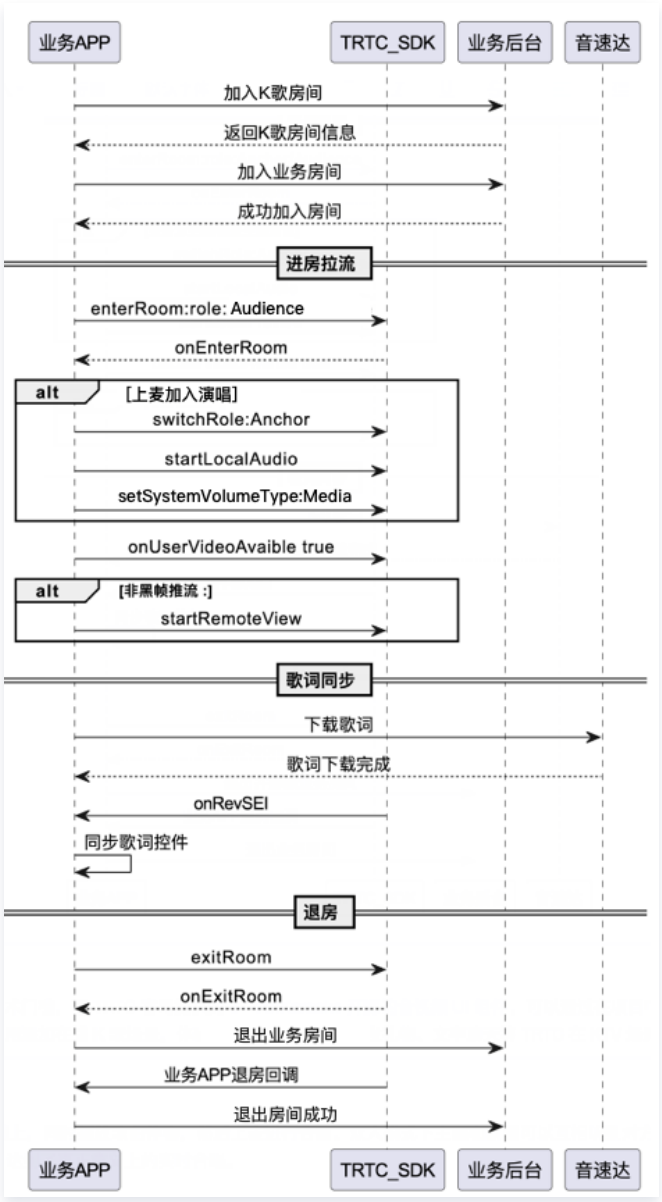
（3）API 调用时序

不同角色的 API 时序调用如下：

房主



观众



说明：

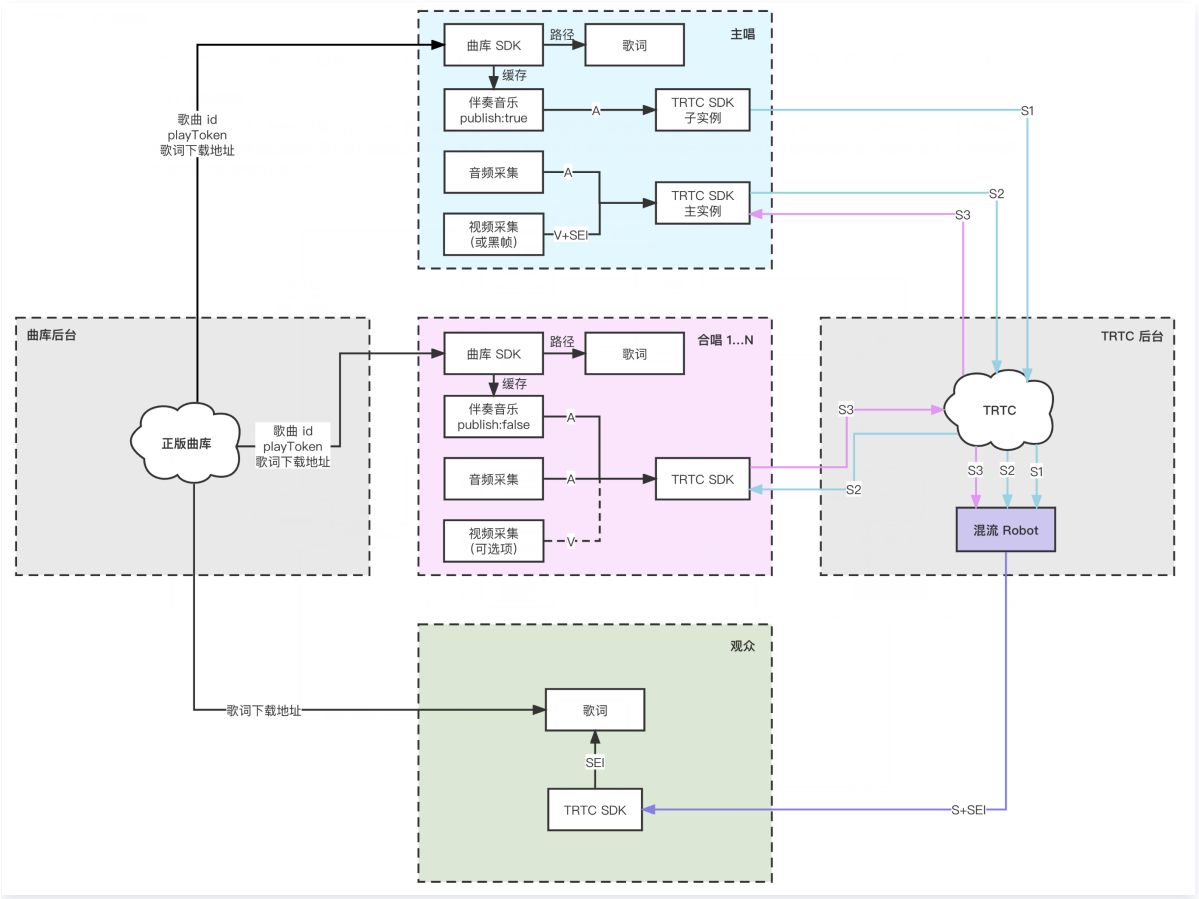
鉴于以上实现方案需要一定的技术门槛，而 TRTC 官网提供的 TUIKaraoke 开源的音视频 UI 组件，可以通过在项目中集成 TUIKaraoke 组件，您只需要编写几行代码就可以为您的应用添加在线 K 歌场景，体验 K 歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。

实时合唱

实时合唱是指各端在连麦的基础上，同时播放歌曲，然后上麦进行合唱。多人模式下合唱者之间都能听到彼此声音，几乎感受不到延迟，达到了真正意义上的实时合唱。

（1）方案架构

在媒体流方面，合唱者互相进行推拉流，同时会由一名***主唱者推出音乐***，其他***合唱者在本地播放音乐***，经过 NTP 进行时间同步。另外，歌曲和所有合唱者的声音都通过混流机器人进行混流处理形成一条流，并回推到 TRTC 房间，观众只需拉一条流即可听到各端同步的声音，完美实现多人合唱的效果。实时合唱方案架构如下图所示。



该方案的优点在于：

- 降低了端到端的时延。
- 提供了用户中途加入合唱的解决方案。
- 精准同步不同端之间的音乐、歌词、人声。
- 改善各端设备性能和本地时间不精准的情况，降低网络环境造成的时延影响。

说明：

- 根据业务需要，可以选择纯音频场景或音视频场景的实时合唱方案；若为纯音频场景则需要补黑帧以发送 SEI 消息用于歌词同步；
- 主唱需要使用子实例同时上行音乐及人声；其他合唱者仅互相拉取人声流，同时本地播放音乐；观众只需拉取一路混流；
- 图中展示的是 RTC 观看方案，混流机器人将合流回推至 RTC 房间；CDN 观看方案下，混流机器人将合流转推至直播 CDN，观众拉取 CDN 流观看。

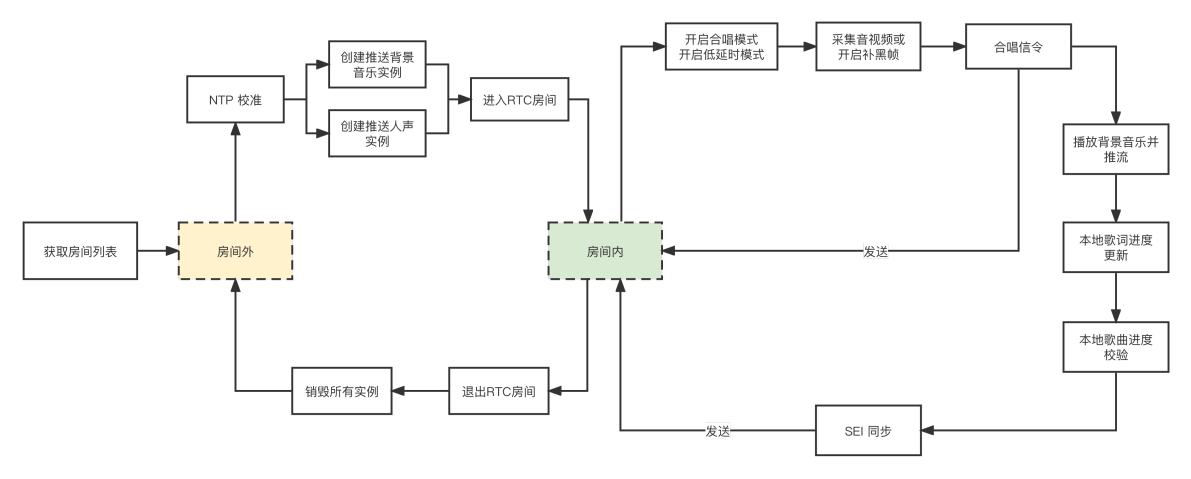
（2）具体实现

我们可以将在线 K 歌房里的用户划分为三种角色：主唱、合唱和观众，如下表所示。

角色	描述	区别
主唱	主唱负责点歌和发送合唱信令，以及 SEI 的发送	• 角色为 Anchor • 上行音乐和人声 • 点歌及发起合唱 • 混流回推 • 发送 SEI 消息
合唱	合唱者可以接收并处理合唱信令，麦上参与合唱	• 角色为 Anchor • 上行人声 • 本地播放音乐 • 接收合唱
观众	进入歌房后，在麦下拉流的观众，也可上麦合唱	• 角色为 Audience • 下行混流 • 接收 SEI 消息 • 申请上麦成为 Anchor

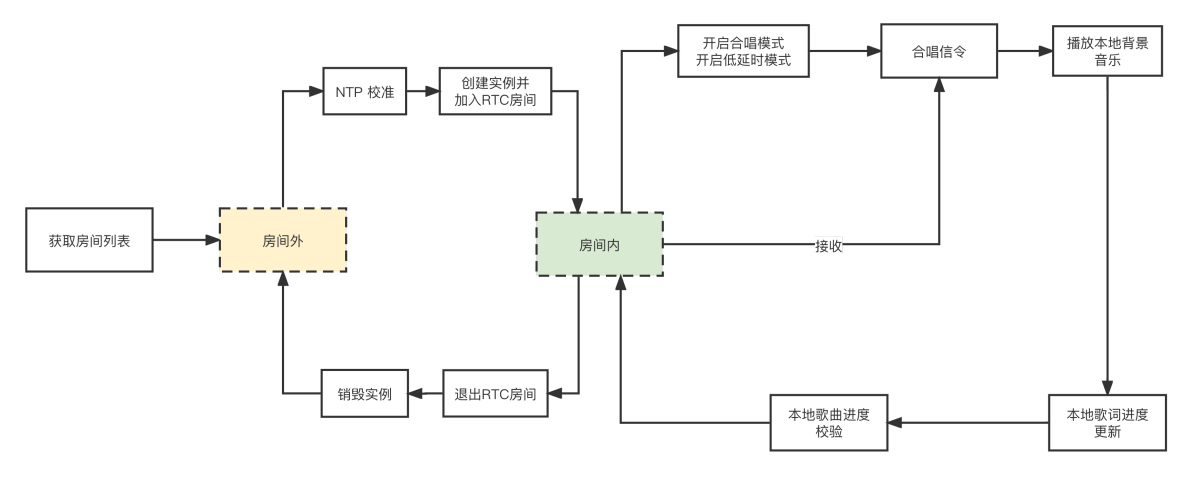
不同角色的基本实现流程如下图所示：

主唱



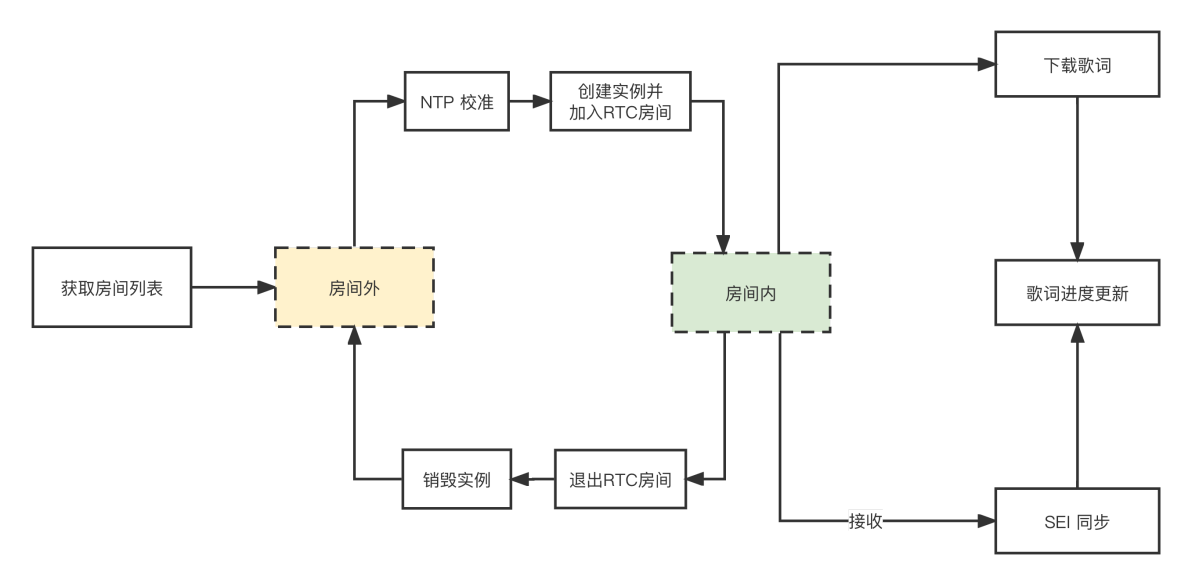
- 主唱需要点播歌曲，发送合唱信令。
- 主唱创建子实例推送人声和音乐，并拉取其他合唱者的人声。
- 在推流后，主唱负责发起混流转推任务。
- 开唱后，播放音乐，通过播放进度回调同步歌词。
- 需要通过 SEI 发送歌曲进度，以便观众端同步歌词。
- 所有演唱者需要根据 NTP 校准本地歌曲播放进度。

合唱



- 合唱者推送一路人声流，拉麦上合唱用户的人声流。
- 合唱者需要监听并接收合唱信令，预加载音乐资源。
- 开唱后，本地播放音乐，合唱者通过播放进度回调同步歌词。
- 所有演唱者需要根据 NTP 校准本地歌曲播放进度。

观众

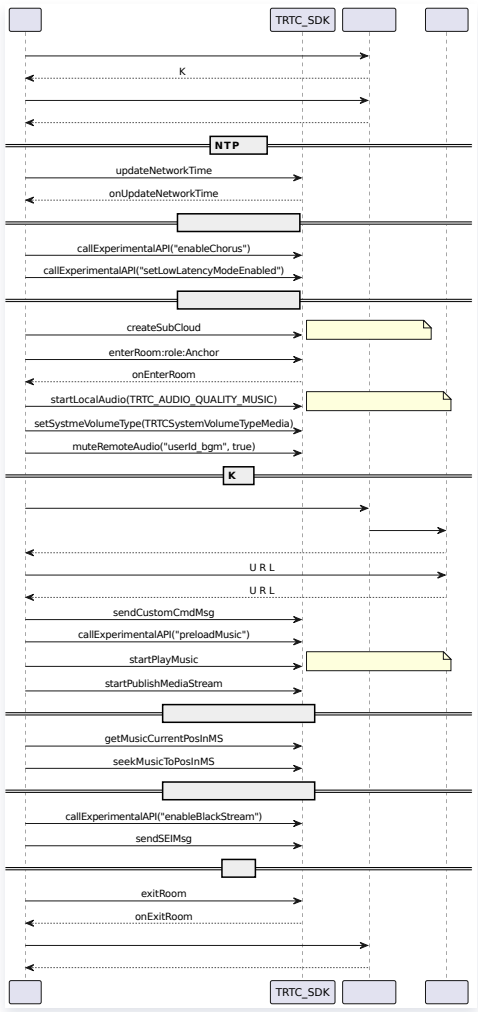


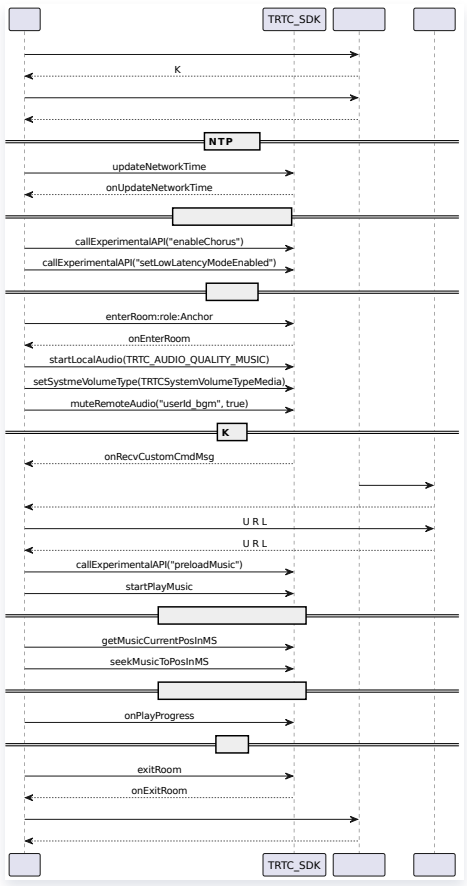
- 拉混流收听合唱。
- 解析混流中 SEI 的歌曲进度信息，用于同步歌词；
- 上麦后停止拉混流，转为拉麦上人声流，同时开启合唱模式。

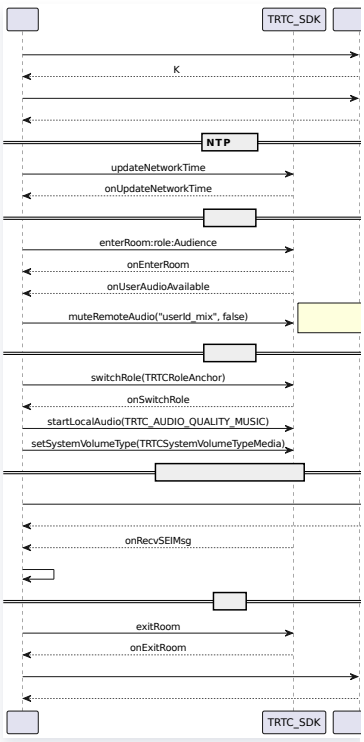
（3）API 调用时序

不同角色的 API 调用时序如下：

主唱 API 时序	合唱 API 时序	观众 API 时序







说明：

鉴于以上实现方案需要一定的技术门槛，而 TRTC 官网提供的TUIKaraoke 开源的音视频 UI 组件，可以通过在项目中集成 TUIKaraoke 组件，您只需要编写几行代码就可以为您的应用添加实时合唱场景，体验 K 歌、麦位管理、收发礼物、文字聊天等 TRTC 在 KTV 场景下的相关能力。

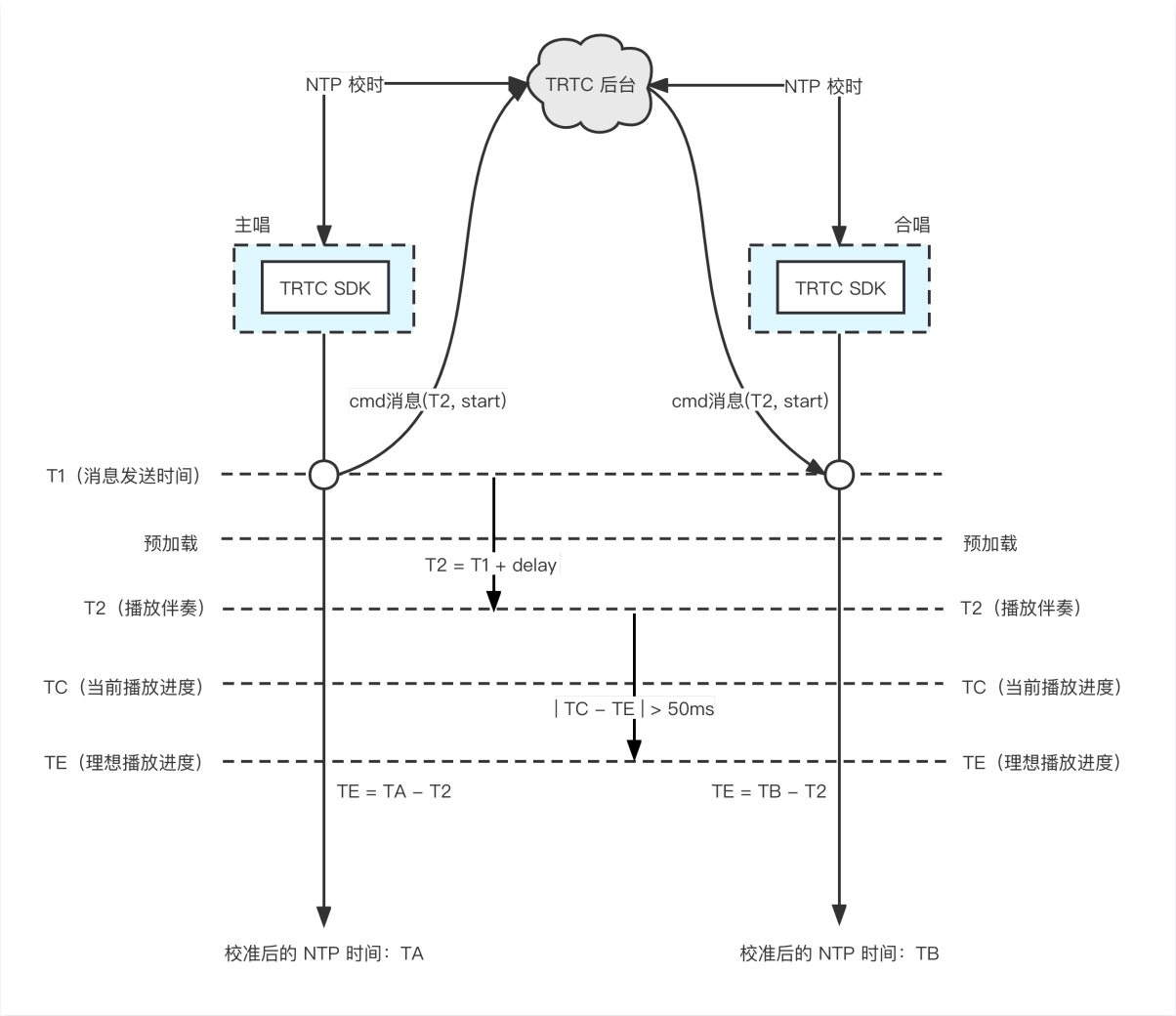
歌曲同步
Android

最近更新时间：2022-10-31 16:18:12

实时方案中需要在开唱后实时同步歌曲进度，避免因歌曲误差而增加端到端延迟。同步歌曲需要基于 NTP 时间，不同设备的本地时钟并不一致，存在一定误差，因此需要引入腾讯云自研 NTP 服务。同时，中途加入合唱的用户也需要同步歌曲进度，只有同步进度后，才能参与合唱。

实现流程

歌曲同步的做法为：主唱用户约定在未来某个时间点（如延迟 N 秒后）开始播放歌曲，其他用户参与合唱。各端的时间都以 NTP 时间为准，NTP 时间会在 TRTC SDK 初始化后开始同步。



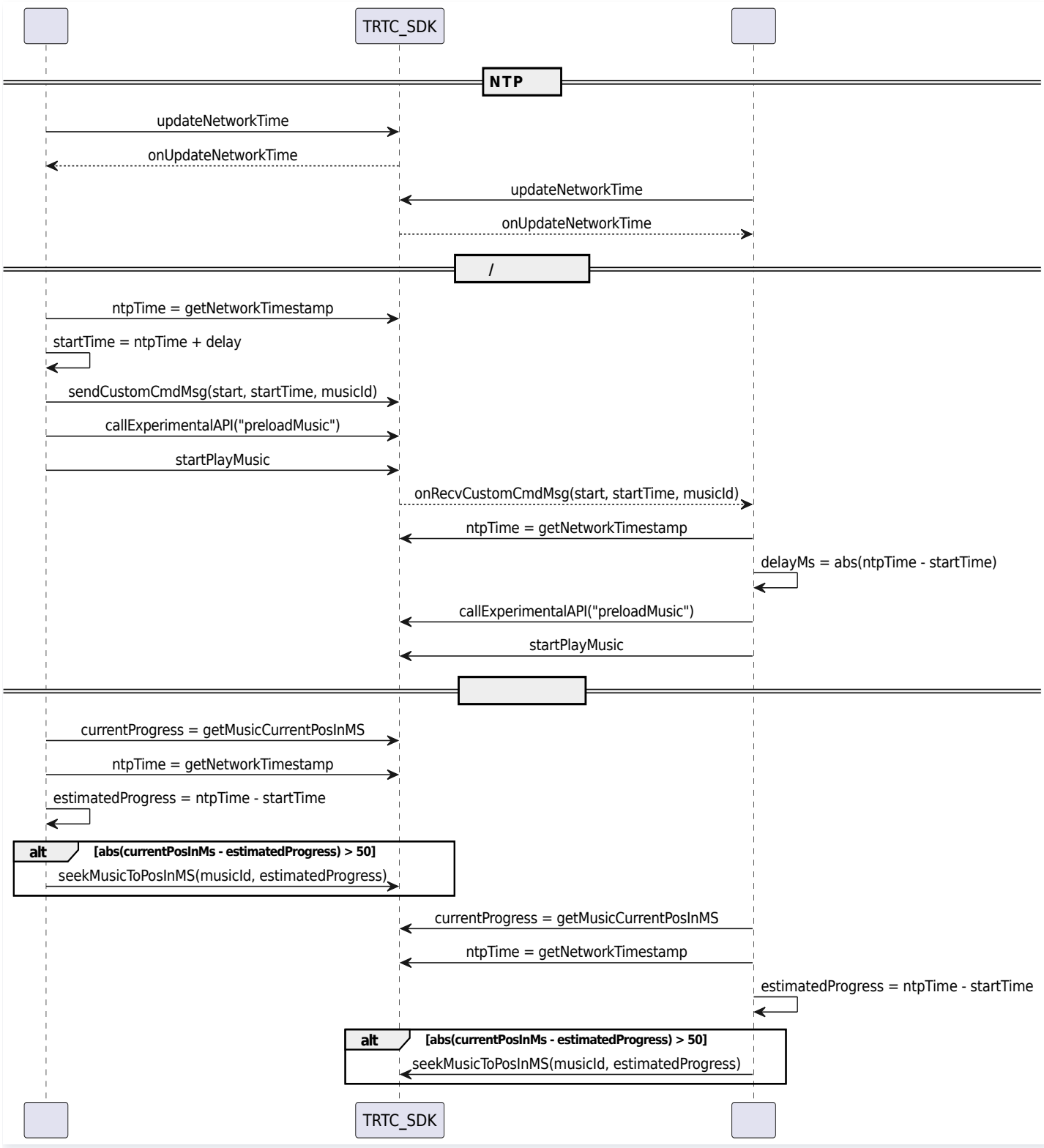
具体流程如下：

1. 各端进行 NTP 校时，向 TRTC 云端更新并获取最新的 NTP 时间 T。
2. 主唱端发送合唱信令（自定义消息），约定合唱开始时间 T2。
3. 本地根据 T2 预加载歌曲，定时播放。
4. 其他合唱用户接收到合唱信令后执行步骤 3。
5. 过程中对本地歌曲播放进度进行校验，当 TE 与 TC 差值超过 50ms 即 seek 校准。

版权所有：腾讯云计算（北京）有限责任公司

> **说明：**
> 这里的 50ms 误差是个典型值，可以根据业务容忍度适当调整，建议在 50ms 上下浮动。

时序图



歌曲同步时序主要可以分为三个部分：NTP 校时、发送及接收合唱信令、歌曲播放进度修正。下面将针对这四部分给出具体的代码实现。

关键代码实现

1. NTP 校时服务

```
TXLiveBase.setListener(new TXLiveBaseListener() {
    @Override
    public void onUpdateNetworkTime(int errCode, String errMsg) {
        super.onUpdateNetworkTime(errCode, errMsg);
        // errCode 0: 校时成功且偏差在30ms以内；1: 校时成功但偏差可能在30ms以上；-1: 校时失败
        if (errCode == 0) {
            // 调用TXLivebase的getNetworkTimestamp即可获取NTP时间戳
            long ntpTime = TXLiveBase.getNetworkTimestamp();
        } else {
            // 重新调用updateNetworkTime启动一次校时
            TXLiveBase.updateNetworkTime();
        }
    }
});
TXLiveBase.updateNetworkTime();
```

2. 主唱端发送合唱信令

```
JSONObject jsonObject = new JSONObject();
jsonObject.put("cmd", "startChorus");
// 约定时间合唱
jsonObject.put("startPlayMusicTS", startTs);
jsonObject.put("musicId", "musicId");
String body = jsonObject.toString();
mTRTCCloud.sendCustomCmdMsg(0, body.getBytes(), false, false);
```

说明：

- 建议主唱以固定时间频率循环向房间内发送合唱信令消息，以便合唱者可以中途加入合唱。
- **不使用 SEI 消息发送合唱信令**的原因：SEI 信息会插入到视频帧中，导致观众侧拉取的视频流中携带很多无效信息。

3. 合唱端接收合唱信令

```
public void onRecvCustomCmdMsg(String userId, int cmdID, int seq, byte[] message) {
    JSONObject json = new JSONObject(new String(message, "UTF-8"));
    String cmd = json.getString("cmd");
    // 合唱命令
    if (cmd.equals("startChorus")) {
    // 合唱开始时间
    long startPlayMusicTs = json.getLong("startPlayMusicTS");
    int musicId = json.getInt("musicId");
    // 合唱约定时间和当前时间差值
    long delayMs = Math.abs(startPlayMusicTs - getNtpTime());
    // 开启预加载，根据合唱约定时间和当前ntp差值，跳跃歌曲进度
    mTRTCCloud.callExperimentalAPI("{\"api\":\"preloadMusic\",\"params\":
{\"musicId\":musicId,\"path\":\"path\",\"startTimeMS\":delayMs}}");
    // 播放歌曲
```

```
    TXAudioEffectManager.AudioMusicParam param = new TXAudioEffectManager.AudioMusicParam(musicId,
musicPath);
    param.publish = false;
    mTRTCCloud.getAudioEffectManager().startPlayMusic(param);
}
```

说明：
合唱端接收到第一个 startChorus 信令后，状态应从“未合唱”转为“合唱中”，此后不再响应 startChorus 信令，避免重复启动 BGM 播放。

4. 歌曲播放进度修正

```
long mStartPlayMusicTs = "最初约定的合唱时间";
long currentProgress = subCloud.getAudioEffectManager().getMusicCurrentPosInMS(musicID);
// 当前歌曲的理想播放时间进度
long estimatedProgress = getNtpTime() - mStartPlayMusicTs;
// 当播放进度超过50ms，进行修正
if (estimatedProgress >= 0 &&; Math.abs(currentProgress - estimatedProgress) > 50) {
    subCloud.getAudioEffectManager().seekMusicToPosInMS(mMusicID, (int) estimatedProgress);
}
```

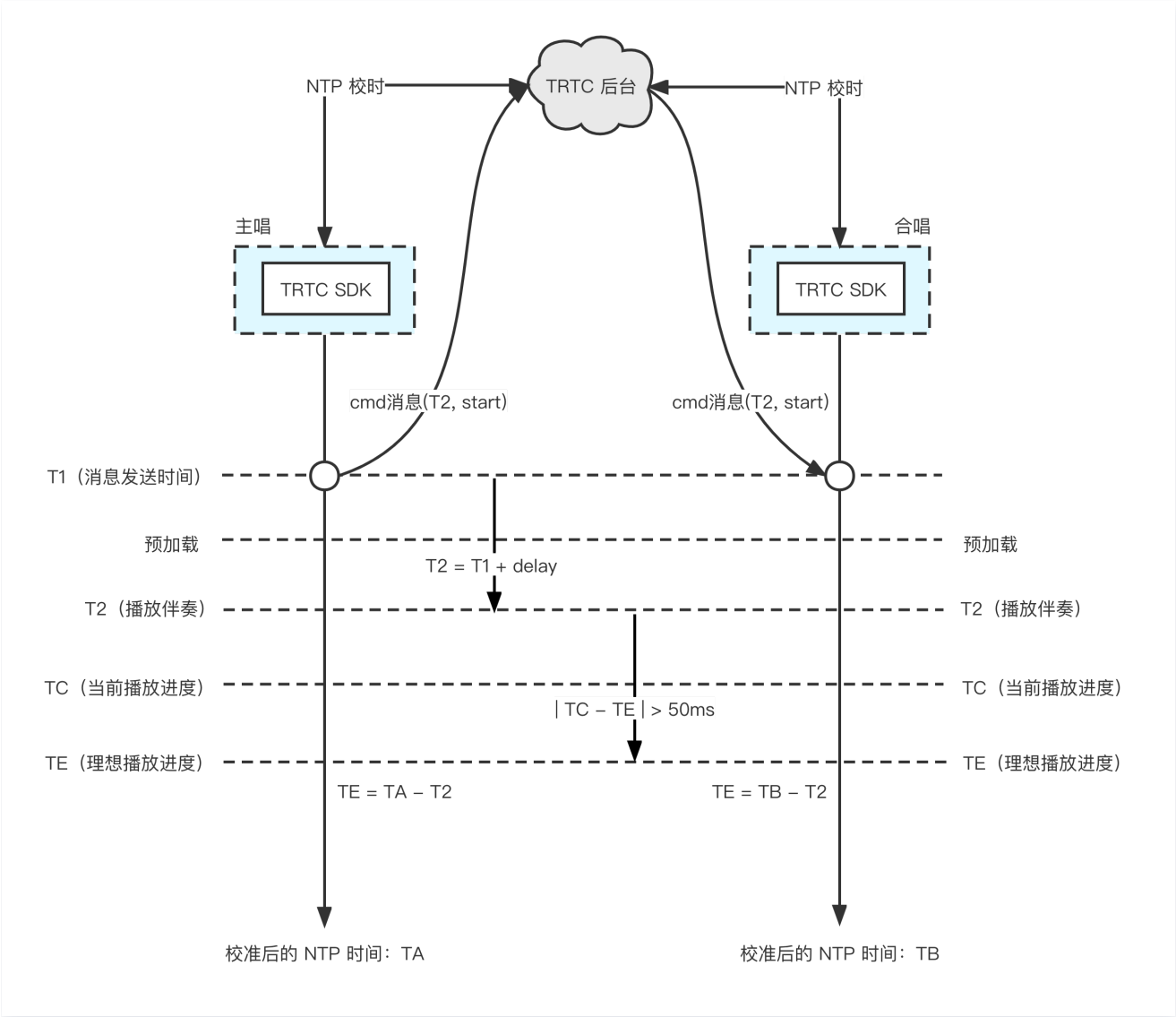
iOS

最近更新时间：2022-10-31 16:18:12

实时方案中需要在开唱后实时同步歌曲进度，避免因歌曲误差而增加端到端延迟。同步歌曲需要基于 NTP 时间，不同设备的本地时钟并不一致，存在一定误差，因此需要引入腾讯云自研 NTP 服务。同时，中途加入合唱的用户也需要同步歌曲进度，只有同步进度后，才能参与合唱。

实现流程

歌曲同步的做法为：主唱用户约定在未来某个时间点（如延迟 N 秒后）开始播放歌曲，其他用户参与合唱。各端的时间都以 NTP 时间为准，NTP 时间会在 TRTC SDK 初始化后开始同步。

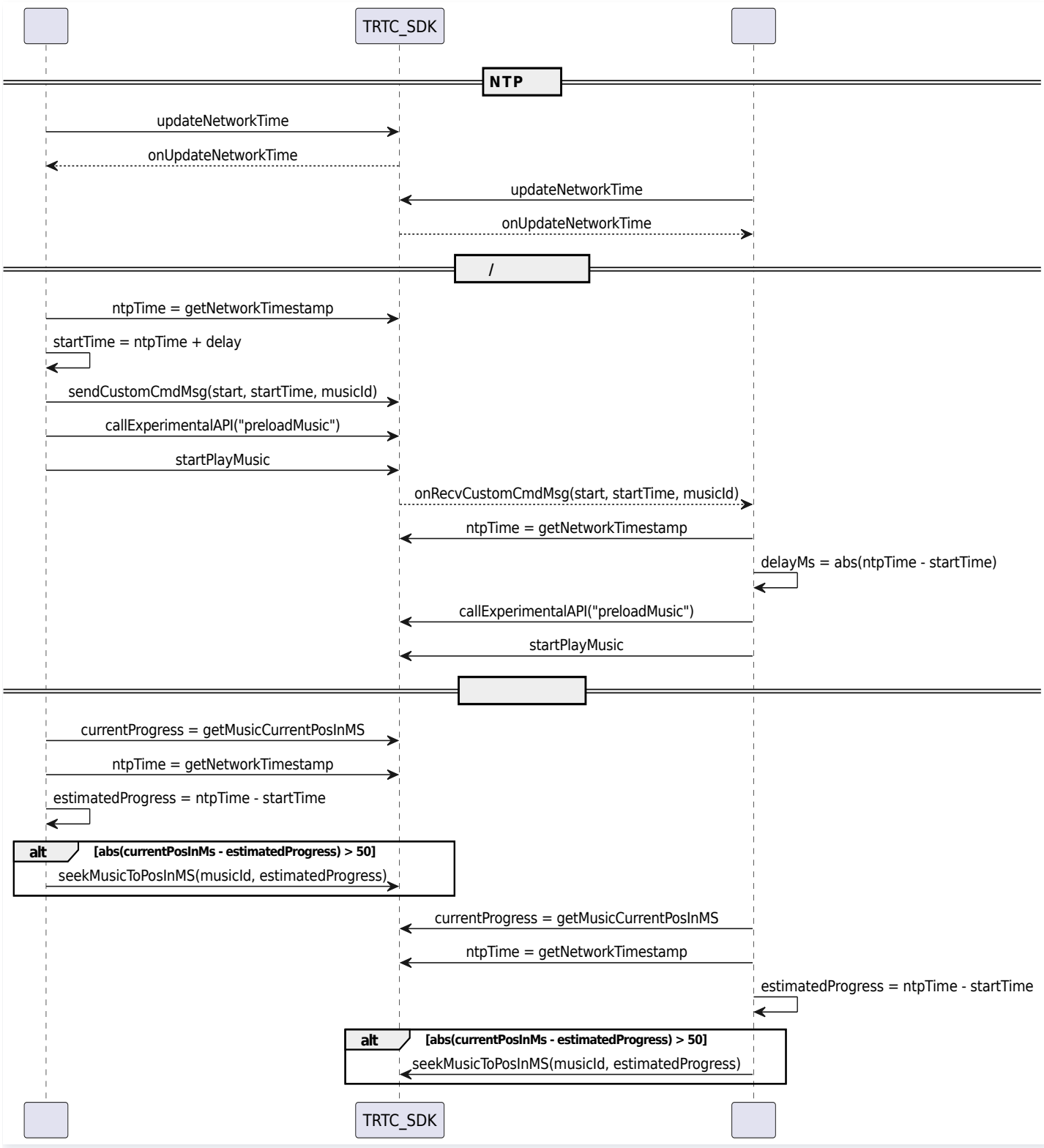


具体流程如下：

1. 各端进行 NTP 校时，向 TRTC 云端更新并获取最新的 NTP 时间 T。
2. 主唱端发送合唱信令（自定义消息），约定合唱开始时间 T2。
3. 本地根据 T2 预加载歌曲，定时播放。
4. 其他合唱用户接收到合唱信令后执行步骤 3。
5. 过程中对本地歌曲播放进度进行校验，当 TE 与 TC 差值超过 50ms 即 seek 校准。

> **说明：**
> 这里的 50ms 误差是个典型值，可以根据业务容忍度适当调整，建议在 50ms 上下浮动。

时序图



歌曲同步时序主要可以分为三个部分：NTP 校时、发送及接收合唱信令、歌曲播放进度修正。下面将针对这四部分给出具体的代码实现。

关键代码实现

1. NTP 校时服务

```
//进房时调用NTP校时接口
[TXLiveBase updateNetworkTime];

// TXLiveBaseDelegate 回调内判断是否校时成功
- (void)onUpdateNetworkTime:(int)errCode message:(NSString *)errMsg {
    // errCode 0: 校时成功且偏差在30ms以内；1: 校时成功但偏差可能在30ms以上；-1: 校时失败
    if (errCode == 0) {
        // 调用TXLivebase的getNetworkTimestamp即可获取NTP时间戳
        NSInteger ntpTime = [TXLiveBase getNetworkTimestamp];
    } else {
        // 重新调用updateNetworkTime启动一次校时
        [TXLiveBase updateNetworkTime];
    }
}
```

2. 主唱端发送合唱信令

```
NSDictionary *json = @{
                    @"cmd": @"startChorus",
                    // 约定时间合唱
                    @"startPlayMusicTS": @(startTs),
                    @"musicId": @"musicId",
                    @"musicDuration": @(musicDuration),
                   };
NSString *jsonString = [self jsonStringFrom:json];
[trtcCloud sendCustomMessage:jsonString reliable:NO];
```

说明：

- 建议主唱以固定时间频率循环向房间内发送合唱信令消息，以便合唱者可以中途加入合唱；
- **不使用 SEI 消息发送合唱信令**的原因： SEI 信息会插入到视频帧中，导致观众侧拉取的视频流中携带很多无效信息。

3. 合唱端接收合唱信令

```
- (void)onRecvCustomCmdMsgUserId:(NSString *)userId cmdID:(NSInteger)cmdId seq:(UInt32)seq message:
(NSData *)message {

    NSString *msg = [[NSString alloc] initWithData:message encoding:NSUTF8StringEncoding];
    NSError *error;
    NSDictionary *json = [NSJSONSerialization JSONObjectWithData:[msg
dataUsingEncoding:NSUTF8StringEncoding]
                                                         options:NSJSONReadingMutableContainers
                                                           error:&error];

    NSObject *cmdObj = [json objectForKey:@"cmd"];

    NSInteger musicDuration = [[json objectForKey:@"musicDuration"] integerValue];
```

```
    NSString *cmd = (NSString *)cmdObj;

    // 合唱命令
    if ([cmd isEqualToString:@"startChorus"]) {
        // 合唱开始时间
        NSObject *startPlayMusicTsObj = [json objectForKey:@"startPlayMusicTS"];
        NSString *musicId = [json objectForKey:@"musicId"];

        NSInteger startPlayMusicTs = ((NSNumber *)startPlayMusicTsObj).longLongValue;
        // 合唱约定时间和当前时间差值
        NSInteger startDelayMS = labs(startPlayMusicTs - [TXLiveBase getNetworkTimestamp]);
        // 开启预加载，根据合唱约定时间和当前ntp差值，跳跃歌曲进度
        NSDictionary *jsonDict = @{
            @"api": @"preloadMusic",
            @"params": @{
                    @"musicId": @(musicId),
                    @"path": path,
                    @"startTimeMS": @(startDelayMS),
            }
        };
        NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:0 error:NULL];
        NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
        [subCloud callExperimentalAPI:jsonString];

        // 播放歌曲
        TXAudioMusicParam *param = [[TXAudioMusicParam alloc] init];
        param.ID = musicId;
        param.path = url;
        param.loopCount = 0;
        param.publish = NO;

        [[subCloud getAudioEffectManager] startPlayMusic:param onStart:^(NSInteger errCode) {
            // 开始播放回调
        } onProgress:^(NSInteger progressMs, NSInteger durationMs) {
            // 歌曲进度回调
        } onComplete:^(NSInteger errCode) {
            // 播放完成回调
        }];

    }
}
```

说明：
合唱端接收到第一个 startChorus 信令后，状态应从“未合唱”转为“合唱中”，此后不再响应 startChorus 信令，避免重复启动 BGM 播放。

4. 歌曲播放进度修正

```
self.startPlayChorusMusicTs; //最初约定的合唱时间
// 当前播放进度
NSInteger currentProgress = [[self audioEffecManager] getMusicCurrentPosInMS:self.currentPlayMusicID];
// 当前歌曲的理想播放时间进度
NSInteger estimatedProgress = [TXLiveBase getNetworkTimestamp] - self.startPlayChorusMusicTs;
```

```
if (estimatedProgress >= 0 && labs(currentProgress - estimatedProgress) > 50) {
    // 当播放进度超过50ms，进行修正
    [[subCloud getAudioEffectManager] seekMusicToPosInMS:self.currentPlayMusicID pts:estimatedProgress];
}
```

歌词同步
Android

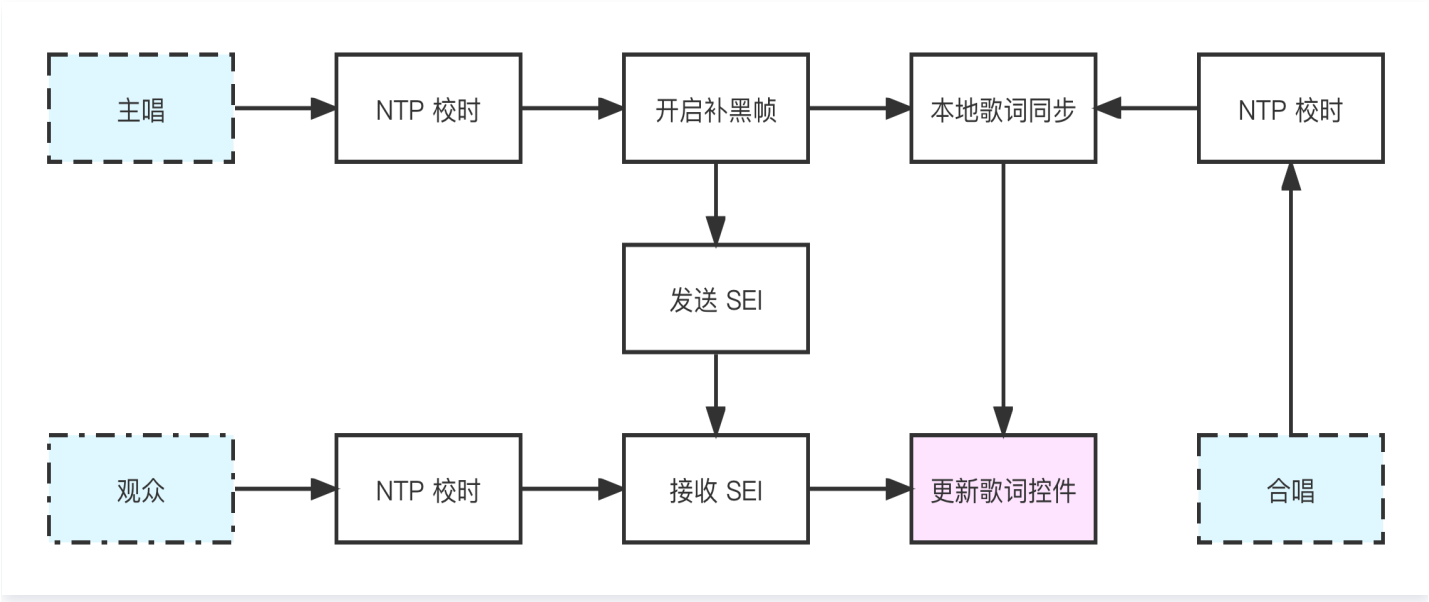
最近更新时间：2023-09-22 14:39:12

实现流程

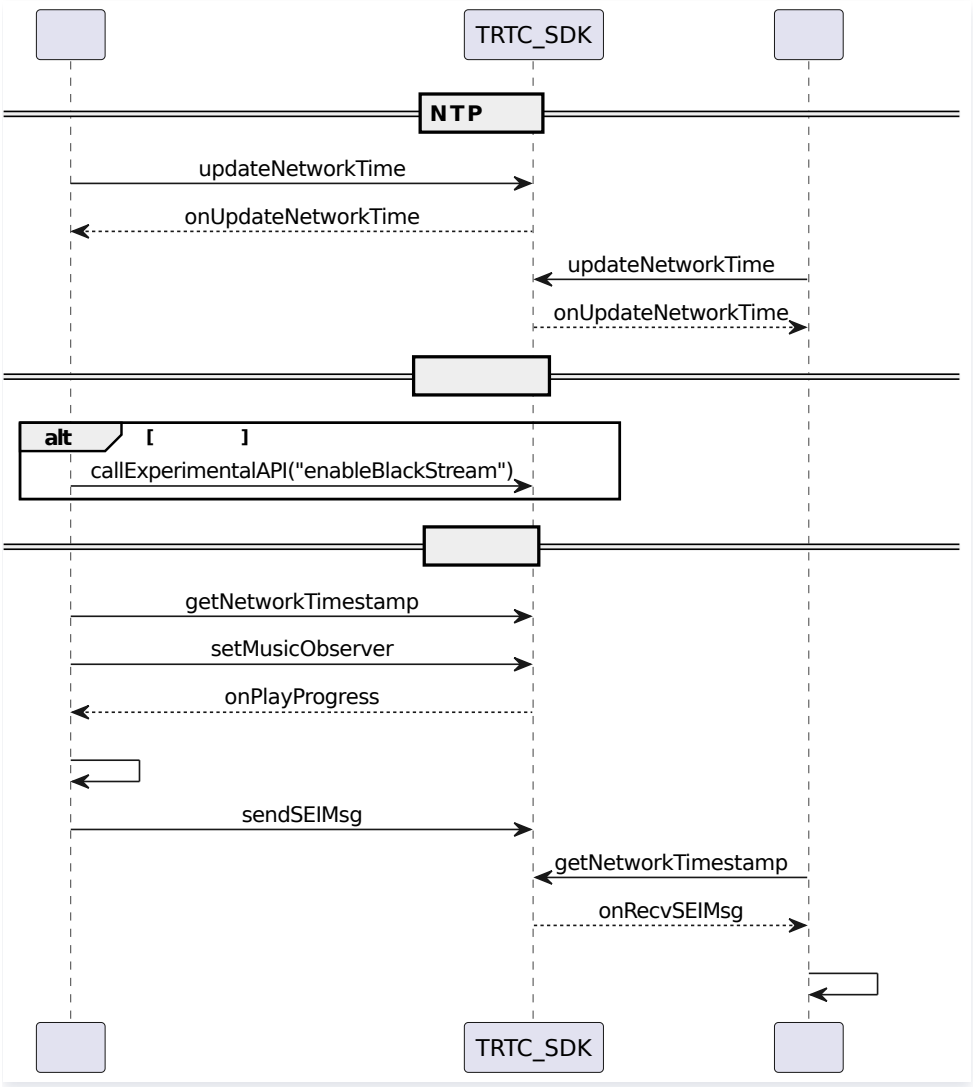
歌词同步方案中，三种不同角色的动作如下：

主唱	合唱	观众
• NTP 校时 • 开启补黑帧 • 发送 SEI 消息 • 本地歌词同步 • 更新歌词控件	• NTP 校时 • 本地歌词同步 • 更新歌词控件	• NTP 校时 • 接收 SEI 消息 • 更新歌词控件

其中，主唱及合唱根据同步后的歌曲播放进度，在本地更新歌词进度；观众端则需要接收由主唱端发送的，包含最新歌词进度的 SEI 消息来更新本地的歌词进度。



时序图



歌词同步时序主要可以分为三个部分：NTP 校时、开启补黑帧、本地及远端歌词同步。NTP 校时的代码实现在歌曲同步中已经给出，下面将针对后两个部分给出具体的代码实现。

关键代码实现

1. 开启补黑帧

```
// 纯音频模式下，主实例（人声实例）需要开启补黑帧以携带 SEI 消息
mTRTCCloud.callExperimentalAPI("{\"api\":\"enableBlackStream\",\"params\": {\"enable\":true}}");
```

说明：

- 该实验性接口 enableBlackStream 需要在进房之后调用。
- 在 Android 端，enable 参数的值类型为布尔型，在 iOS 端为整型。
- 接收端需要在收到 onUserVideoAvailable(userId, true) 时调用 startRemoteView(userId, null)。

2. 通过 SEI 消息发送歌曲进度

```
mAudioEffectManager.setMusicObserver(mCurPlayMusicId, new
TXAudioEffectManager.TXMusicPlayObserver() {
    @Override
```

```
    public void onPlayProgress(int id, long curPtsMS, long durationMS) {
        JSONObject jsonObject = new JSONObject();
        // 当前 ntp 时间
        long ntpTime = TXLiveBase.getNetworkTimestamp();
        jsonObject.put("bgmProgressTime", curTime);
        jsonObject.put("ntpTime", ntpTime);
        jsonObject.put("musicId", musicId);
        jsonObject.put("duration", duration);
        jsonObject.toString().getBytes();
        mTRTCCloud.sendSEIMsg(jsonObject.toString().getBytes(), 1);
    }
}
```

说明：

- 主唱发送 SEI 消息的频率由背景音乐播放事件回调的频率决定，一般为 200ms；
- **不直接使用 CMD 消息发送歌曲进度**的原因：SEI 通道传输的信令可以伴随视频帧一直传输到直播 CDN 上，对于拉取 CDN 流的观众具有更好的兼容性。

3. 本地及远端歌词同步

```
// 本地歌词同步
mAudioEffectManager.setMusicObserver(mCurPlayMusicId, new
TXAudioEffectManager.TXMusicPlayObserver() {
    @Override
    public void onPlayProgress(int id, long curPtsMS, long durationMS) {
        ...
        // TODO 更新歌词控件逻辑：
        // 根据最新进度和本地歌词进度误差，判断是否需要 seek 歌词控件
        ...
    }
}

// 远端歌词同步
@Override
public void onRecvSEIMsg(String userId, byte[] data) {
    String result = new String(data);
    JSONObject jsonObject = new JSONObject(result);
    long bgmProgressTime = jsonObject.getLong("bgmProgressTime");
    long ntpTime = jsonObject.getLong("ntpTime");
    String musicId = jsonObject.getString("musicId");
    long duration = jsonObject.getLong("duration");
    ...
    // TODO 更新歌词控件逻辑：
    // 根据接收到的最新进度和本地歌词进度误差，判断是否需要 seek 歌词控件
    ...
}
```

说明：

如果复用 TUIKaraoke 组件的歌词控件，请参照 TUIKaraoke LyricsView 部分的代码逻辑同步歌词控件进度。

iOS

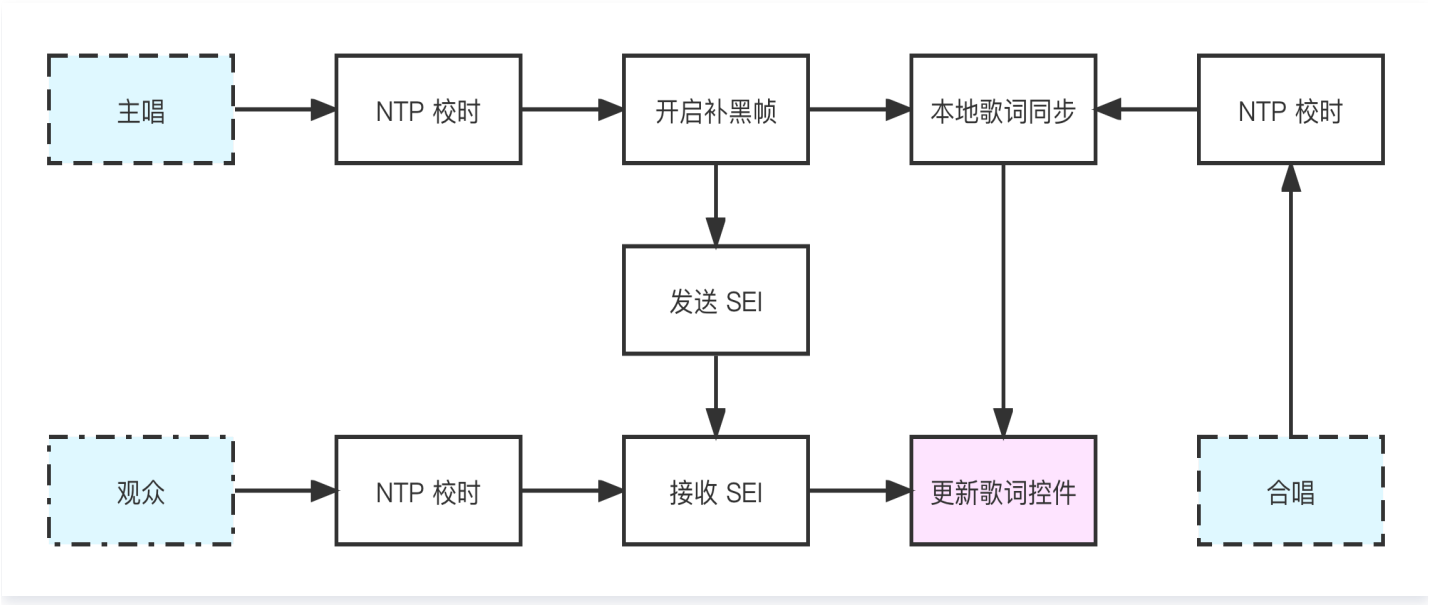
最近更新时间：2022-10-31 16:18:12

1.1 实现流程

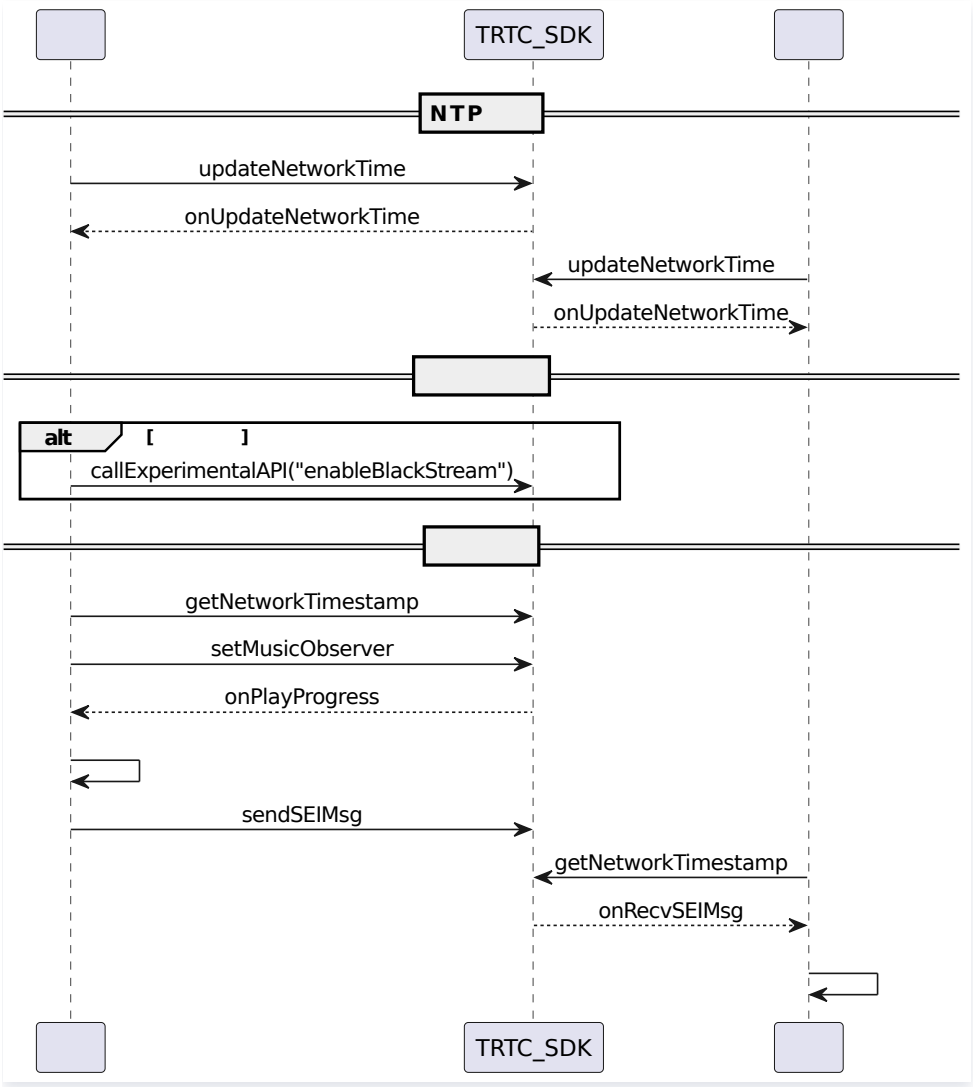
歌词同步方案中，三种不同角色的动作如下：

主唱	合唱	观众
• NTP 校时 • 开启补黑帧 • 发送 SEI 消息 • 本地歌词同步 • 更新歌词控件	• NTP 校时 • 本地歌词同步 • 更新歌词控件	• NTP 校时 • 接收 SEI 消息 • 更新歌词控件

其中，主唱及合唱根据同步后的歌曲播放进度，在本地更新歌词进度；观众端则需要接收由主唱端发送的，包含最新歌词进度的 SEI 消息来更新本地的歌词进度。



时序图



歌词同步时序主要可以分为三个部分：NTP 校时、开启补黑帧、本地及远端歌词同步。NTP 校时的代码实现在歌曲同步中已经给出，下面将针对后两个部分给出具体的代码实现。

关键代码实现

1. 开启补黑帧

```
// 纯音频模式下，主实例（人声实例）需要开启补黑帧以携带 SEI 消息
NSDictionary *jsonDic = @{
                    @"api": @"enableBlackStream",
                    @"params":
                        @{
                          @"enable": @(1)
                        }
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[trtcCloud callExperimentalAPI:jsonString];
```

说明：

- 该实验性接口 enableBlackStream 需要在进房之后调用；
- 在 Android 端，enable 参数的值类型为布尔型，在 iOS 端为整型；
- 接收端需要在收到 onUserVideoAvailable(userId, true) 时调用 startRemoteView(userId, null)。

2. 通过 SEI 消息发送歌曲进度

```
TXAudioMusicProgressBlock progressBlock = ^(NSInteger progressMs, NSInteger durationMs) {
    //当前 ntp 时间
    NSInteger ntpTime = [TXLiveBase getNetworkTimestamp];
    //通知歌曲进度，用户会在这里进行歌词的滚动
    NSDictionary *progressMsg = @{
            @"bgmProgressTime":@(progressMs),
            @"ntpTime":@(ntpTime),
            @"musicId": @(musicId),
            @"duration": @(durationMs),
    };
    NSData *jsonData = [NSJSONSerialization dataWithJSONObject:progressMsg
options:NSJSONWritingPrettyPrinted error:nil];
    NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
    [trtcCloud sendSEIMsg:[jsonString dataUsingEncoding:NSUTF8StringEncoding] repeatCount:1];
};
```

说明：

- 主唱发送 SEI 消息的频率由背景音乐播放事件回调的频率决定，一般为 200ms；
- **不直接使用 CMD 消息发送歌曲进度**的原因：SEI 通道传输的信令可以伴随视频帧一直传输到直播 CDN 上，对于拉取 CDN 流的观众具有更好的兼容性。

3. 本地及远端歌词同步

```
// 本地歌词同步
TXAudioMusicProgressBlock progressBlock = ^(NSInteger progressMs, NSInteger durationMs) {
    ...
    // TODO 更新歌词控件逻辑：
    // 根据最新进度和本地歌词进度误差，判断是否需要 seek 歌词控件
    ...
};

// 远端歌词同步
- (void)onRecvSEIMsg:(NSString *)userId message:(NSData *)message {
    NSError *err = nil;
    NSDictionary *dic = [NSJSONSerialization JSONObjectWithData:message
options:NSJSONReadingMutableContainers error:&err];
    if (err || ![dic isKindOfClass:[NSDictionary class]]) {
        // 解析出错
        return;
    }
    NSInteger bgmProgressTime = [[dic objectForKey:@"bgmProgressTime"] integerValue];
    NSInteger ntpTime = [[dic objectForKey:@"ntpTime"] integerValue];
    int32_t musicId = [[dic objectForKey:@"musicId"] intValue];
    NSInteger duration = [[dic objectForKey:@"duration"] integerValue];
    ...
```

```
    // TODO 更新歌词控件逻辑：
    // 根据接收到的最新进度和本地歌词进度误差，判断是否需要 seek 歌词控件
    ...
}
```

说明：
如果复用 TUIKaraoke 组件的歌词控件，请参照 TUIKaraoke TRTCLyricView 部分的代码逻辑同步歌词控件进度。

人声同步
Android

最近更新时间：2023-09-19 14:58:12

人声与歌曲同步介绍

因为本地人声采集的 jitter buffer、歌曲播放混音的 jitter buffer 以及声音播放到人耳到歌唱存在有一定的 GAP 的，所以演唱者完全对着歌词和 BGM 播放时候，在远端观众感觉 BGM 播放、人声、歌词是有一定的延迟的。合唱方案在 TRTC SDK 内部使用了使用低延迟的 AAudio 采集，具体只需要在进房后开启合唱模式与低延时模式。

具体代码实现

开启合唱模式

```
// 主实例（人声实例）开启合唱模式（调低缓冲区间、音频冗余保护）
mTRTCCloud.callExperimentalAPI("{\"api\":\"enableChorus\",\"params\":{\"enable\":1,\"audioSource\":0}}");
// 子实例（伴奏实例）开启合唱模式（调低缓冲区间、音频冗余保护）
subCloud.callExperimentalAPI("{\"api\":\"enableChorus\",\"params\":{\"enable\":1,\"audioSource\":1}}");
```

说明：
开启合唱模式的实验性接口 enableChorus 的参数设置：
- audioSource：0（人声）。
- audioSource：1（伴奏）。

开启低延时模式（高性能音频 AAudio）

```
// 主实例（人声实例）开启高性能音频 AAudio
mTRTCCloud.callExperimentalAPI("{\"api\":\"setLowLatencyModeEnabled\",\"params\":{\"enable\":1}}");
// 子实例（伴奏实例）开启高性能音频 AAudio
subCloud.callExperimentalAPI("{\"api\":\"setLowLatencyModeEnabled\",\"params\":{\"enable\":1}}");
```

版权所有：腾讯云计算（北京）有限责任公司

iOS

最近更新时间：2022-10-31 16:18:12

人声与歌曲同步介绍

因为本地人声采集的 jitter buffer、歌曲播放混音的 jitter buffer 以及声音播放到人耳到歌唱存在有一定的 GAP 的，所以演唱者完全对着歌词和 BGM 播放时候，在远端观众感觉 BGM 播放、人声、歌词是有一定的延迟的，合唱方案在 TRTC SDK 内部使用了使用低延迟的 AAudio 采集，具体只需要在进房后开启合唱模式与低延时模式。

具体代码实现

开启合唱模式

```
// 主实例（人声实例）开启合唱模式（调低缓冲区间、音频冗余保护）
NSDictionary *jsonDic = @{
                    @"api": @"enableChorus",
                    @"params": @{
                            @"enable": @(YES),
                            @"audioSource": @(0)
                           }
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[trtcCloud callExperimentalAPI:jsonString];
// 子实例（伴奏实例）开启合唱模式（调低缓冲区间、音频冗余保护）
NSDictionary *jsonDic = @{
                    @"api": @"enableChorus",
                    @"params": @{
                            @"enable": @(YES),
                            @"audioSource": @(1)
                           }
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[subCloud callExperimentalAPI:jsonString];
```

说明：

开启合唱模式的实验性接口 enableChorus 的参数设置：

- audioSource：0（人声）
- audioSource：1（伴奏）

开启低延时模式（高性能音频 AAudio）

```
// 主实例（人声实例）开启高性能音频 AAudio
NSDictionary *jsonDic = @{
                    @"api": @"setLowLatencyModeEnabled",
                    @"params": @{
                            @"enable": @(1)
```

```
                    }
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[trtcCloud callExperimentalAPI:jsonString];
// 子实例（伴奏实例）开启高性能音频 AAudio
NSDictionary *jsonDic = @{
                    @"api": @"setLowLatencyModeEnabled",
                    @"params": @{
                            @"enable": @(1)
                        }
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[subCloud callExperimentalAPI:jsonString];
```

混流方案
Android

最近更新时间：2022-10-31 16:18:13

具体代码实现

1. 创建主子实例

```
// 创建 TRTCCloud 主实例（人声实例）
TRTCCloud mTRTCCloud = TRTCCloud.sharedInstance(getApplicationContext());
// 创建 TRTCCloud 子实例（伴奏实例）
TRTCCloud subCloud = mTRTCCloud.createSubCloud();
```

说明：
实时合唱方案中，主唱端需要分别创建主实例-人声实例和子实例-伴奏实例，分别用于上行人声及伴奏音乐。

2. 人声实例进房推流

```
TRTCCloudDef.TRTCParams params = new TRTCCloudDef.TRTCParams();
params.sdkAppId = sdkAppId;
params.userId = mUserId;
params.userSig = userSig;
params.role = TRTCCloudDef.TRTCRoleAnchor;
params.roomId = mRoomId;
mTRTCCloud.enterRoom(params, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
// 打开音频上行，设置音质
mTRTCCloud.startLocalAudio(TRTCCloudDef.TRTC_AUDIO_QUALITY_MUSIC);
// 设置媒体类型
mTRTCCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);
// 静音远端伴奏音乐
mTRTCCloud.muteRemoteAudio(mUserId + "_bgm", true);
```

注意：
- 纯 RTC 音频场景下，进房场景推荐选用 VOICE_CHATROOM。
- 若有视频或转推 CDN 需求，进房场景则须选用 LIVE，VOICE_CHATROOM 会在转推时添加纯音频参数，从而导致 SEI 消息无法透传。
- 主唱/合唱端需要 muteRemoteAudio(true) 取消订阅伴奏实例上行的音频流，否则会重复播放本地及远端的伴奏音乐。

3. 伴奏实例进房推流

```
TRTCCloudDef.TRTCParams bgmParams = new TRTCCloudDef.TRTCParams();
bgmParams.sdkAppId = sdkAppId;
bgmParams.userId = mUserId + "_bgm";
bgmParams.userSig = userSig;
bgmParams.role = TRTCCloudDef.TRTCRoleAnchor;
bgmParams.roomId = mRoomId;
subCloud.enterRoom(bgmParams, TRTCCloudDef.TRTC_APP_SCENE_LIVE);
```

版权所有：腾讯云计算（北京）有限责任公司

```
//设置媒体类型
subCloud.setSystemVolumeType(TRTCCloudDef.TRTCSystemVolumeTypeMedia);

// 开启预加载
subCloud.callExperimentalAPI("{\"api\":\"preloadMusic\",\"params\":
{\"musicId\":musicId,\"path\":\"path\",\"startTimeMS\":startTimeMS}}");
// 播放伴奏音乐并推流（在约定时间播放）
TXAudioEffectManager.AudioMusicParam param = new TXAudioEffectManager.AudioMusicParam(musicID,
musicPath);
// 将伴奏音乐传到远端
param.publish = true;
subCloud.getAudioEffectManager().startPlayMusic(param);
```

⚠ 注意：

- 注意区分主实例和子实例的 userId，确保不重复且容易辨别；
- 伴奏实例播放背景音乐参数 AudioMusicParam 设置：
  - publish：true（音乐在本地播放的同时，远端用户也能听到该音乐）
  - publish：false（默认值，只能在本地听到该音乐，远端用户听不到）

4. 发起混流转码回推

```
// 创建 TRTCPublishTarget 对象
TRTCCloudDef.TRTCPublishTarget target = new TRTCCloudDef.TRTCPublishTarget();
// 混流后回推到房间，若发布到 CDN 应填 TRTC_PublishMixStream_ToCdn
target.mode = TRTCCloudDef.TRTC_PublishMixStream_ToRoom;
target.mixStreamIdentity.intRoomId = Integer.parseInt(mRoomId);
// 混流机器人的 userid，不能和房间内其他用户的 userid 重复
target.mixStreamIdentity.userId = mUserId + "_mix";

// 设置转码后的音频流的编码参数
TRTCCloudDef.TRTCStreamEncoderParam trtcStreamEncoderParam = new
TRTCCloudDef.TRTCStreamEncoderParam();
trtcStreamEncoderParam.audioEncodedChannelNum = 2;
trtcStreamEncoderParam.audioEncodedKbps = 64;
trtcStreamEncoderParam.audioEncodedCodecType = 2;
trtcStreamEncoderParam.audioEncodedSampleRate = 48000;

// 设置音频混流参数
TRTCCloudDef.TRTCStreamMixingConfig trtcStreamMixingConfig = new
TRTCCloudDef.TRTCStreamMixingConfig();
// 支持填写空值，会自动将所有主播的音频混合输出
trtcStreamMixingConfig.audioMixUserList = null;

// 发起混流转推请求
mTRTCCloud.startPublishMediaStream(target, trtcStreamEncoderParam, trtcStreamMixingConfig);
```

⚠ 注意：

- 优先选择主唱通过混流机器人向后台发起混流转推，将伴奏音乐和各路人声混合后回推至 TRTC 房间，或转推至直播 CDN。
- 自动订阅模式下，参与混流转码的主播默认互相拉取单流，不接收回推房间的混流；观众自动拉取回推房间的混流，不再接收单流。

- 这里的混流转推方法 startPublishMediaStream 采用全新的后台架构，旧版应用需提供 SdkAppId 申请升级后方可使用。

版权所有：腾讯云计算（北京）有限责任公司

iOS

最近更新时间：2022-10-31 16:18:13

具体代码实现

1. 创建主子实例

```
// 创建 TRTCCloud 主实例（人声实例）
TRTCCloud *trtcCloud = [TRTCCloud sharedInstance];
// 创建 TRTCCloud 子实例（伴奏实例）
TRTCCloud *subCloud = [trtcCloud createSubCloud];
```

⚠ 注意：

实时合唱方案中，主唱端需要分别创建主实例-人声实例和子实例-伴奏实例，分别用于上行人声及伴奏音乐。

2. 人声实例进房推流

```
TRTCParams *params = [[TRTCParams alloc] init];
params.sdkAppId = sdkAppId;
params.userId = userId;
params.userSig = userSign;
params.role = TRTCRoleAnchor;
params.roomId = roomIdIntValue;
[trtcCloud enterRoom:params appScene:TRTCAppSceneLIVE];
// 打开音频上行，设置音质
[trtcCloud startLocalAudio:TRTCAudioQualityMusic];
// 设置媒体类型
[trtcCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
// 静音远端伴奏音乐
[trtcCloud muteRemoteAudio:remoteAudioId mute:YES];
```

⚠ 注意：

- 纯 RTC 音频场景下，进房场景推荐选用 VOICE_CHATROOM。
- 若有视频或转推 CDN 需求，进房场景则须选用 LIVE，VOICE_CHATROOM 会在转推时添加纯音频参数，从而导致 SEI 消息无法透传。
- 主唱/合唱端需要 muteRemoteAudio(true) 取消订阅伴奏实例上行的音频流，否则会重复播放本地及远端的伴奏音乐。

3. 伴奏实例进房推流

```
TRTCParams *bgmParams = [[TRTCParams alloc] init];
bgmParams.sdkAppId = sdkAppId;
bgmParams.userId = [NSString stringWithFormat:@"%@%@",userId,@"_bgm"];
bgmParams.userSig = bgmUserSign;
bgmParams.role = TRTCRoleAnchor;
bgmParams.roomId = roomIdIntValue;
[subCloud enterRoom:bgmParams appScene:TRTCAppSceneLIVE];
//设置媒体类型
[subCloud setSystemVolumeType:TRTCSystemVolumeTypeMedia];
```

版权所有：腾讯云计算（北京）有限责任公司

```
// 开启预加载
NSDictionary *jsonDict = @{
                        @"api": @"preloadMusic",
                        @"params": @{
                                @"musicId": @(self.currentPlayMusicID),
                                @"path": path,
                                @"startTimeMS": @(startMs),
                                }
                        };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDict options:0 error:NULL];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[subCloud callExperimentalAPI:jsonString];
// 播放伴奏音乐并推流（在约定时间播放）
TXAudioMusicParam *musicParam = [[TXAudioMusicParam alloc] init];
musicParam.ID = musicID;
musicParam.path = url;
musicParam.loopCount = 0;
musicParam.publish = YES;
// 将伴奏音乐传到远端
param.publish = YES;
[[subCloud getAudioEffectManager] startPlayMusic:musicParam onStart:startBlock onProgress:progressBlock
onComplete:completedBlock]
```

⚠ 注意：

- 注意区分主实例和子实例的 userId，确保不重复且容易辨别。
- 伴奏实例播放背景音乐参数 musicParam 设置：
  - publish：YES（音乐在本地播放的同时，远端用户也能听到该音乐）
  - publish：NO（默认值，只能在本地听到该音乐，远端用户听不到）

4. 发起混流转码回推

```
// 创建 TRTCPublishTarget 对象
TRTCPublishTarget *publishTarget = [[TRTCPublishTarget alloc] init];
// 混流后回推到房间，若发布到 CDN 应填 TRTCPublishMixStreamToCdn
publishTarget.mode = TRTCPublishMixStreamToRoom;
// 混流机器人的 userid，不能和房间内其他用户的 userid 重复
publishTarget.mixStreamIdentity = [NSString stringWithFormat:@"%@%@",userId,@"_mix"];

// 设置转码后的音频流的编码参数
TRTCStreamEncoderParam *streamEncoderParam = [[TRTCStreamEncoderParam alloc] init];
streamEncoderParam.videoEncodedFPS = 15;
streamEncoderParam.videoEncodedGOP = 3;
streamEncoderParam.videoEncodedKbps = 30;
streamEncoderParam.audioEncodedSampleRate = 48000;
streamEncoderParam.audioEncodedChannelNum = 2;
streamEncoderParam.audioEncodedKbps = 64;
streamEncoderParam.audioEncodedCodecType = 2;

// 设置音频混流参数
TRTCStreamMixingConfig *streamMixingConfig = [[TRTCStreamMixingConfig alloc] init];
// 支持填写空值，会自动将所有主播的音频混合输出
streamMixingConfig.audioMixUserList = @[];
```

```
// 发起混流转推请求
[trtcCloud startPublishMediaStream:publishTarget encoderParam:streamEncoderParam
mixingConfig:streamMixingConfig];
```

注意：

- 优先选择主唱通过混流机器人向后台发起混流转推，将伴奏音乐和各路人声混合后回推至 TRTC 房间，或转推至直播 CDN。
- 自动订阅模式下，参与混流转码的主播默认互相拉取单流，不接收回推房间的混流；观众自动拉取回推房间的混流，不再接收单流。
- 这里的混流转推方法 startPublishMediaStream 采用全新的后台架构，旧版应用需提供 SdkAppId 申请升级后方可使用。

版权所有：腾讯云计算（北京）有限责任公司

API 参考（TUIKaraoke）
Android

最近更新时间：2022-10-31 16:18:13

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 方案介绍-实现步骤 。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCKaraokeRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateHandler	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。

exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。
getUserInfoList	获取指定 userId 的用户信息，如果为 null，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

版权所有：腾讯云计算（北京）有限责任公司

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	Karaoke 房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

版权所有：腾讯云计算（北京）有限责任公司

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。
onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

SDK 基础函数

sharedInstance

获取 TRTCKaraokeRoom 单例对象。

```
public static synchronized TRTCKaraokeRoom sharedInstance(Context context);
```

参数如下表所示：

参数	类型	含义
context	Context	Android 上下文，内部会转为 ApplicationContext 用于系统 API 调用

destroySharedInstance

销毁 TRTCKaraokeRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
public static void destroySharedInstance();
```

setDelegate

TRTCKaraokeRoom 事件回调，您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知。

```
public abstract void setDelegate(TRTCKaraokeRoomDelegate delegate);
```

> **说明**
> setDelegate 是 TRTCKaraokeRoom 的代理回调。

setDelegateHandler

设置事件回调所在的线程。

```
public abstract void setDelegateHandler(Handler handler);
```

参数如下表所示：

参数	类型	含义
handler	Handler	TRTCKaraokeRoom 中的各种状态通知，会派发到您指定的 handler 线程。

login

登录。

```
public abstract void login(int sdkAppId,
 String userId, String userSig,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义

sdkAppId	int	您可以在实时音视频控制台 > 应用管理 > 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig 。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
public abstract void logout(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
public abstract void setSelfProfile(String userName, String avatarURL,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。
callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
public abstract void createRoom(int roomId, TRTCKaraokeRoomDef.RoomParam roomParam,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个 Karaoke 房间列表，腾讯云暂不提供 Karaoke 房间列表的管理服务，请自行管理您的 Karaoke 房间列表。

roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的 Karaoke 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
public abstract void destroyRoom(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
public abstract void enterRoom(int roomId, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Karaoke 房间列表，可能包含多个 Karaoke 房间的 roomId 和房间信息。
2. 听众选择一个 Karaoke 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
public abstract void exitRoom(TRTCKaraokeRoomCallback.ActionCallback callback);
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> **说明**
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
public abstract void getRoomInfoList(List<Integer> roomIdList,
TRTCKaraokeRoomCallback.RoomInfoCallback callback);
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定userId的用户信息。

```
public abstract void getUserInfoList(List<String> userIdList, TRTCKaraokeRoomCallback.UserListCallback
userlistcallback);
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

音乐播放接口

startPlayMusic

播放音乐（上麦后调用）。

> **说明**
> - 播放音乐后，自身会收到 `onMusicPrepareToPlay` 的事件通知。
> - 音乐播放中，房间内所有成员会不断收到 `onMusicProgressUpdate` 的事件通知。
> - 音乐播放完成，自身会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void startPlayMusic(int musicID, String originalUrl, String accompanyUrl);
```

参数如下表所示：

参数	类型	含义
musicID	int	音乐的 ID。
originalUrl	String	原唱音乐的绝对路径。
accompanyUrl	String	伴奏音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

> **说明**
> 停止播放后，会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void stopPlayMusic();
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

> **说明**
> - `onMusicProgressUpdate` 的事件通知会暂停
> - 不会收到 `onMusicCompletePlaying` 的事件通知。

```
public abstract void pausePlayMusic();
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> **说明**
> 不会收到 `onMusicPrepareToPlay` 的事件通知。

```
public abstract void resumePlayMusic();
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> **说明**

上麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
public abstract void enterSeat(int seatIndex, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

说明

下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void leaveSeat(TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

pickSeat

抱人上麦（房主调用）。

说明

房主抱人上麦，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
public abstract void pickSeat(int seatIndex, String userId, TRTCKaraokeRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 sendInvitation 向听众申请，收到 onInvitationAccept 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> **说明**
> 房主踢人下麦，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
public abstract void kickSeat(int seatIndex, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> **说明**
> 静音/解除静音某个麦位，房间内所有成员会收到 onSeatListChange 和 onSeatMute 的事件通知。

```
public abstract void muteSeat(int seatIndex, boolean isMute, TRTCKaraokeRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> **说明**
> 房主封禁/解禁对应麦位，房间内所有成员会收到 onSeatListChange 和 onSeatClose 的事件通知。

版权所有：腾讯云计算（北京）有限责任公司

```
public abstract void closeSeat(int seatIndex, boolean isClose, TRTCKaraokeRoomCallback.ActionCallback
callback);
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
public abstract void startMicrophone();
```

stopMicrophone

停止麦克风采集。

```
public abstract void stopMicrophone();
```

setAudioQuality

设置音质。

```
public abstract void setAudioQuality(int quality);
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK 。

muteLocalAudio

静音/取消静音本地的音频。

```
public abstract void muteLocalAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK 。

版权所有：腾讯云计算（北京）有限责任公司

setSpeaker

设置开启扬声器。

```
public abstract void setSpeaker(boolean useSpeaker);
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
public abstract void setAudioCaptureVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
public abstract void setAudioPlayoutVolume(int volume);
```

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 - 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
public abstract void muteRemoteAudio(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

版权所有：腾讯云计算（北京）有限责任公司

```
public abstract void muteAllRemoteAudio(boolean mute);
```

参数如下表所示：

参数	类型	含义
mute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
public abstract void setVoiceEarMonitorEnable(boolean enable);
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
public abstract TXAudioEffectManager getAudioEffectManager();
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
public abstract void sendRoomTextMsg(String message, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
public abstract void sendRoomCustomMsg(String cmd, String message,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
public abstract String sendInvitation(String cmd, String userId, String content,
TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。
content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
public abstract void acceptInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
public abstract void rejectInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
public abstract void cancelInvitation(String id, TRTCKaraokeRoomCallback.ActionCallback callback);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

TRTCKaraokeRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

```
void onError(int code, String message);
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
void onWarning(int code, String message);
```

参数如下表所示：

参数	类型	含义

code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
void onDebugLog(String message);
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
void onRoomDestroy(String roomId);
```

参数如下表所示：

参数	类型	含义
roomId	String	房间 ID。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
void onRoomInfoChange(TRTCKaraokeRoomDef.RoomInfo roomInfo);
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用 muteLocalAudio，房间内的其他用户都会收到此通知。

```
void onUserMicrophoneMute(String userId, boolean mute);
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。

版权所有：腾讯云计算（北京）有限责任公司

mute	boolean	音量大小，取值：0 – 100。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
void onUserVolumeUpdate(List<TRTCCloudDef.TRTCVolumeInfo> userVolumes, int totalVolume);
```

参数如下表所示：

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
void onSeatListChange(List<SeatInfo> seatInfoList);
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
void onAnchorEnterSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
void onAnchorLeaveSeat(int index, TRTCKaraokeRoomDef.UserInfo user);
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	下麦用户的详细信息。

onSeatMute

房主禁麦。

```
void onSeatMute(int index, boolean isMute);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
void onSeatClose(int index, boolean isClose);
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
void onAudienceEnter(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
void onAudienceExit(TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
void onRecvRoomTextMsg(String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
void onRecvRoomCustomMsg(String cmd, String message, TRTCKaraokeRoomDef.UserInfo userInfo);
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
void onReceiveNewInvitation(String id, String inviter, String cmd, String content);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。

content	String	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
void onInviteeAccepted(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
void onInviteeRejected(String id, String invitee);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
void onInvitationCancelled(String id, String inviter);
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。

音乐播放状态回调

onMusicPrepareToPlay

准备播放音乐的回调

```
void onMusicPrepareToPlay(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。

onMusicProgressUpdate

歌曲播放进度的回调

```
void onMusicProgressUpdate(int musicID, long progress, long total);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。
progress	long	当前播放时间，单位： ms。
total	long	总时间，单位： ms。

onMusicCompletePlaying

播放完成音乐的回调

```
void onMusicCompletePlaying(int musicID);
```

参数如下表所示：

参数	类型	含义
musicID	int	播放时传入的 musicID。

版权所有：腾讯云计算（北京）有限责任公司

iOS

最近更新时间：2023-08-25 15:47:44

TRTCKaraokeRoom 是基于腾讯云实时音视频（TRTC）和即时通信 IM 服务组合而成的组件，支持以下功能：

- 房主创建新的 Karaoke 房间开播，听众进入 Karaoke 房间收听/互动。
- 房主可以管理点歌、将座位上的麦上主播踢下麦。
- 房主还能对座位进行封禁，其他听众就不能再进行申请上麦了。
- 听众可以申请上麦，变成麦上主播，上麦后可以点歌和唱歌，也可以随时下麦成为普通的听众。
- 支持发送礼物和各种文本、自定义消息，自定义消息可用于实现弹幕、点赞等。

> **说明**
> TUIKit 系列组件同时使用了腾讯云 实时音视频 TRTC 和 即时通信 IM 两个基础 PaaS 服务，开通实时音视频后会同步开通即时通信IM服务。即时通信 IM 服务详细计费规则请参见 即时通信 – 价格说明，TRTC 开通会默认关联开通 IM SDK 的体验版，仅支持100个 DAU。

TRTCKaraokeRoom 是一个开源的 Class，依赖腾讯云的两个闭源 SDK，具体的实现过程请参见 方案介绍–实现步骤 。

- TRTC SDK：使用 TRTC SDK 作为低延时语音聊天组件。
- IM SDK：使用 IM SDK 的 AVChatroom 实现聊天室的功能，同时，通过 IM 的属性接口来存储麦位表等房间信息，邀请信令可以用于上麦申请/抱麦申请。

TRTCKaraokeRoom API 概览

SDK 基础函数

API	描述
sharedInstance	获取单例对象。
destroySharedInstance	销毁单例对象。
setDelegate	设置事件回调。
setDelegateQueue	设置事件回调所在的线程。
login	登录。
logout	登出。
setSelfProfile	修改个人信息。

房间相关接口函数

API	描述
createRoom	创建房间（房主调用），若房间不存在，系统将自动创建一个新房间。
destroyRoom	销毁房间（房主调用）。
enterRoom	进入房间（听众调用）。
exitRoom	退出房间（听众调用）。
getRoomInfoList	获取房间列表的详细信息。

版权所有：腾讯云计算（北京）有限责任公司

getUserInfoList	获取指定 userId 的用户信息，如果为 nil，则获取房间内所有人的信息。

音乐播放接口

API	描述
startPlayMusic	开始播放音乐。
stopPlayMusic	停止播放音乐。
pausePlayMusic	暂停播放音乐。
resumePlayMusic	恢复播放音乐。

麦位管理接口

API	描述
enterSeat	主动上麦（听众端和房主均可调用）。
leaveSeat	主动下麦（主播调用）。
pickSeat	抱人上麦（房主调用）。
kickSeat	踢人下麦（房主调用）。
muteSeat	静音/解除静音某个麦位（房主调用）。
closeSeat	封禁/解禁某个麦位（房主调用）。

本地音频操作接口

API	描述
startMicrophone	开启麦克风采集。
stopMicrophone	停止麦克风采集。
setAudioQuality	设置音质。
muteLocalAudio	开启/关闭本地静音。
setSpeaker	设置开启扬声器。
setAudioCaptureVolume	设置麦克风采集音量。
setAudioPlayoutVolume	设置播放音量。
setVoiceEarMonitorEnable	开启/关闭 耳返。

远端用户音频操作接口

API	描述
muteRemoteAudio	静音/解除静音指定成员。
muteAllRemoteAudio	静音/解除静音所有成员。

版权所有：腾讯云计算（北京）有限责任公司

背景音乐音效相关接口

API	描述
getAudioEffectManager	获取背景音乐音效管理对象 TXAudioEffectManager 。

消息发送相关接口

API	描述
sendRoomTextMsg	在房间中广播文本消息，一般用于弹幕聊天。
sendRoomCustomMsg	发送自定义文本消息。

邀请信令相关接口

API	描述
sendInvitation	向用户发送邀请。
acceptInvitation	接受邀请。
rejectInvitation	拒绝邀请。
cancelInvitation	取消邀请。

TRTCKaraokeRoomDelegate API 概览

通用事件回调

API	描述
onError	错误回调。
onWarning	警告回调。
onDebugLog	Log 回调。

房间事件回调

API	描述
onRoomDestroy	房间被销毁的回调。
onRoomInfoChange	语聊房间信息变更回调。
onUserVolumeUpdate	用户通话音量回调。

麦位变更回调

API	描述
onSeatListChange	全量的麦位列表变化。
onAnchorEnterSeat	有成员上麦（主动上麦/房主抱人上麦）。
onAnchorLeaveSeat	有成员下麦（主动下麦/房主踢人下麦）。

版权所有：腾讯云计算（北京）有限责任公司

onSeatMute	房主禁麦。
onUserMicrophoneMute	用户麦克风是否静音。
onSeatClose	房主封麦。

听众进出事件回调

API	描述
onAudienceEnter	收到听众进房通知。
onAudienceExit	收到听众退房通知。

消息事件回调

API	描述
onRecvRoomTextMsg	收到文本消息。
onRecvRoomCustomMsg	收到自定义消息。

信令事件回调

API	描述
onReceiveNewInvitation	收到新的邀请请求。
onInviteeAccepted	被邀请人接受邀请。
onInviteeRejected	被邀请人拒绝邀请。
onInvitationCancelled	邀请人取消邀请。

歌曲事件回调

API	描述
onMusicProgressUpdate	歌曲播放进度的回调。
onMusicPrepareToPlay	准备播放音乐的回调。
onMusicCompletePlaying	播放完成音乐的回调。

SDK 基础函数

sharedInstance

获取 TRTCKaraokeRoom 单例对象。

```
/**
* 获取 TRTCKaraokeRoom 单例对象
*
* - returns: TRTCKaraokeRoom 实例
* - note: 可以调用 {@link TRTCKaraokeRoom#destroySharedInstance()} 销毁单例对象
*/
```

```
+ (instancetype)sharedInstance NS_SWIFT_NAME(shared());
```

destroySharedInstance

销毁 TRTCKaraokeRoom 单例对象。

> **说明**
> 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例无法再使用，需要重新调用 sharedInstance 获取新实例。

```
/**
* 销毁 TRTCKaraokeRoom 单例对象
*
* - note: 销毁实例后，外部缓存的 TRTCKaraokeRoom 实例不能再使用，需要重新调用 {@link
TRTCKaraokeRoom#sharedInstance()} 获取新实例
*/
+ (void)destroySharedInstance NS_SWIFT_NAME(destroyShared());
```

setDelegate

TRTCKaraokeRoom 事件回调，您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知。

```
/**
* 设置组件回调接口
*
* 您可以通过 TRTCKaraokeRoomDelegate 获得 TRTCKaraokeRoom 的各种状态通知
*
* - parameter delegate 回调接口
* - note: TRTCKaraokeRoom 中的回调事件，默认是在 Main Queue 中回调给您；如果您需要指定事件回调所在的队列，
可使用 {@link TRTCKaraokeRoom#setDelegateQueue(queue)}
*/
- (void)setDelegate:(id<TRTCKaraokeRoomDelegate>)delegate NS_SWIFT_NAME(setDelegate(delegate:));
```

> **说明**
> setDelegate 是 TRTCKaraokeRoom 的代理回调。

setDelegateQueue

设置事件回调所在的线程队列，默认发送到主线程 MainQueue 中。

```
/**
* 设置事件回调所在的队列
*
* - parameter queue 队列，TRTCKaraokeRoom 中的各种状态通知回调，会派发到您指定的queue。
*/
- (void)setDelegateQueue:(dispatch_queue_t)queue NS_SWIFT_NAME(setDelegateQueue(queue:));
```

参数如下表所示：

参数	类型	含义
queue	dispatch_queue_t	TRTCKaraokeRoom 中的各种状态通知，会派发到您指定的线程队列里去。

login

登录。

```
- (void)login:(int)sdkAppID
       userId:(NSString *)userId
      userSig:(NSString *)userSig
     callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(login(sdkAppID:userId:userSig:callback:));
```

参数如下表所示：

参数	类型	含义
sdkAppId	int	您可以在实时音视频控制台 >应用管理> 应用信息中查看 SDKAppID。
userId	String	当前用户的 ID，字符串类型，只允许包含英文字母（a-z 和 A-Z）、数字（0-9）、连词符（-）和下划线（_）。
userSig	String	腾讯云设计的一种安全保护签名，获取方式请参见 如何计算及使用 UserSig。
callback	ActionCallback	登录回调，成功时 code 为0。

logout

登出。

```
- (void)logout:(ActionCallback _Nullable)callback NS_SWIFT_NAME(logout(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	登出回调，成功时 code 为0。

setSelfProfile

修改个人信息。

```
- (void)setSelfProfile:(NSString *)userName avatarURL:(NSString *)avatarURL callback:(ActionCallback
_Nullable)callback NS_SWIFT_NAME(setSelfProfile(userName:avatarURL:callback:));
```

参数如下表所示：

参数	类型	含义
userName	String	昵称。
avatarURL	String	头像地址。

callback	ActionCallback	个人信息设置回调，成功时 code 为0。

房间相关接口函数

createRoom

创建房间（房主调用）。

```
- (void)createRoom:(int)roomID roomParam:(RoomParam *)roomParam callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(createRoom(roomID:roomParam:callback:));
```

参数如下表所示：

参数	类型	含义
roomId	int	房间标识，需要由您分配并进行统一管理。多个 roomID 可以汇总成一个语聊房间列表，腾讯云暂不提供语聊房间列表的管理服务，请自行管理您的语聊房间列表。
roomParam	TRTCCreateRoomParam	房间信息，用于房间描述的信息。例如房间名称、麦位信息、封面信息等。如果需要麦位管理，必须要填入房间的麦位数。
callback	ActionCallback	创建房间的结果回调，成功时 code 为0。

房主开播的正常调用流程如下：

1. 房主调用 `createRoom` 创建新的 Karaoke 房间，此时传入房间 ID、上麦是否需要房主确认、麦位数等房间属性信息。
2. 房主创建房间成功后，调用 `enterSeat` 进入座位。
3. 房主收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
4. 房主还会收到麦位表有成员进入的 `onAnchorEnterSeat` 的事件通知，此时会自动打开麦克风采集。

destroyRoom

销毁房间（房主调用）。房主在创建房间后，可以调用这个函数来销毁房间。

```
- (void)destroyRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(destroyRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	销毁房间的结果回调，成功时 code 为0。

enterRoom

进入房间（听众调用）。

```
- (void)enterRoom:(NSInteger)roomID callback:(ActionCallback _Nullable)callback NS_SWIFT_NAME(enterRoom(roomID:callback:));
```

参数如下表所示：

参数	类型	含义

roomId	int	房间标识。
callback	ActionCallback	进入房间的结果回调，成功时 code 为0。

听众进房收听的正常调用流程如下：

1. 听众向您的服务端获取最新的 Karaoke 房间列表，可能包含多个语聊房间的 roomId 和房间信息。
2. 听众选择一个 Karaoke 房间，调用 `enterRoom` 并传入房间号即可进入该房间。
3. 进房后会收到组件的 `onRoomInfoChange` 房间属性变化事件通知，此时可以记录房间属性并做相应改变，例如 UI 展示房间名、记录上麦是否需要请求房主同意等。
4. 进房后会收到组件的 `onSeatListChange` 麦位表变化事件通知，此时可以将麦位表变化刷新到 UI 界面上。
5. 进房后还会收到麦位表有主播进入的 `onAnchorEnterSeat` 的事件通知。

exitRoom

退出房间。

```
- (void)exitRoom:(ActionCallback _Nullable)callback NS_SWIFT_NAME(exitRoom(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	退出房间的结果回调，成功时 code 为0。

getRoomInfoList

获取房间列表的详细信息，其中房间名称、房间封面是房主在创建 `createRoom()` 时通过 roomInfo 设置的。

> **说明**
> 如果房间列表和房间信息都由您自行管理，可忽略该函数。

```
- (void)getRoomInfoList:(NSArray<NSNumber *> *)roomIdList callback:(KaraokeInfoCallback
_Nullable)callback NS_SWIFT_NAME(getRoomInfoList(roomIdList:callback:));
```

参数如下表所示：

参数	类型	含义
roomIdList	List<Integer>	房间号列表。
callback	RoomInfoCallback	房间详细信息回调。

getUserInfoList

获取指定 userId 的用户信息。

```
- (void)getUserInfoList:(NSArray<NSString *> * _Nullable)userIDList callback:(KaraokeUserListCallback
_Nullable)callback NS_SWIFT_NAME(getUserInfoList(userIDList:callback:));
```

参数如下表所示：

参数	类型	含义
userIdList	List<String>	需要获取的用户 ID 列表，如果为 null，则获取房间内所有人的信息。
userlistcallback	UserListCallback	用户详细信息回调。

音乐播放接口

startPlayMusic

播放音乐（上麦后调用）。

> **说明**
> - 播放音乐后，自身会收到 `onMusicPrepareToPlay` 的事件通知。
> - 音乐播放中，房间内所有成员会不断收到 `onMusicProgressUpdate` 的事件通知。
> - 音乐播放完成，自身会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)startPlayMusic:(int32_t)musicID originalUrl:(NSString *)originalUrl accompanyUrl:(NSString *)backingUrl NS_SWIFT_NAME(startPlayMusic(musicID:originalUrl:accompanyUrl:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	音乐的 ID。
originalUrl	String	原唱音乐的绝对路径。
accompanyUrl	String	伴奏音乐的绝对路径。

调用该接口后会停止上一个正在播放的歌曲。

stopPlayMusic

停止播放音乐（播放音乐时调用）。

> **说明**
> 停止播放后，会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)stopPlayMusic NS_SWIFT_NAME(stopPlayMusic());
```

pausePlayMusic

暂停正在播放的音乐（播放音乐时调用）。

> **说明**
> - `onMusicProgressUpdate` 的事件通知会暂停
> - 不会收到 `onMusicCompletePlaying` 的事件通知。

```
- (void)pausePlayMusic NS_SWIFT_NAME(pausePlayMusic());
```

resumePlayMusic

恢复暂停过的音乐（暂停后调用）。

> **说明**
> 不会收到 onMusicPrepareToPlay 的事件通知。

```
- (void)resumePlayMusic NS_SWIFT_NAME(resumePlayMusic());
```

麦位管理接口

enterSeat

主动上麦（听众端和房主均可调用）。

> **说明**
> 上麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorEnterSeat 的事件通知。

```
- (void)enterSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(enterSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要上麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是听众申请上麦需要房主同意的场景，可以先调用 sendInvitation 向房主申请，收到 onInvitationAccept 后再调用该函数。

leaveSeat

主动下麦（主播调用）。

> **说明**
> 下麦成功后，房间内所有成员会收到 onSeatListChange 和 onAnchorLeaveSeat 的事件通知。

```
- (void)leaveSeat:(ActionCallback _Nullable)callback NS_SWIFT_NAME(leaveSeat(callback:));
```

参数如下表所示：

参数	类型	含义
callback	ActionCallback	操作回调。

版权所有：腾讯云计算（北京）有限责任公司

pickSeat

抱人上麦（房主调用）。

> **说明**
> 房主抱人上麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorEnterSeat` 的事件通知。

```
- (void)pickSeat:(NSInteger)seatIndex userId:(NSString *)userId callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(pickSeat(seatIndex:userId:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要抱上麦的麦位序号。
userId	String	用户 ID。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。如果是房主需要听众同意，听众才会上麦的场景，可以先调用 `sendInvitation` 向听众申请，收到 `onInvitationAccept` 后再调用该函数。

kickSeat

踢人下麦（房主调用）。

> **说明**
> 房主踢人下麦，房间内所有成员会收到 `onSeatListChange` 和 `onAnchorLeaveSeat` 的事件通知。

```
- (void)kickSeat:(NSInteger)seatIndex callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(kickSeat(seatIndex:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要踢下麦的麦位序号。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。

muteSeat

静音/解除静音某个麦位（房主调用）。

> **说明**
> 静音/解除静音某个麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatMute` 的事件通知。

```
- (void)muteSeat:(NSInteger)seatIndex isMute:(BOOL)isMute callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(muteSeat(seatIndex:isMute:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isMute	boolean	true：静音对应麦位；false：解除静音对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。对应 seatIndex 座位上的主播，会自动调用 muteAudio 进行静音/解禁。

closeSeat

封禁/解禁某个麦位（房主调用）。

> **说明**
> 房主封禁/解禁对应麦位，房间内所有成员会收到 `onSeatListChange` 和 `onSeatClose` 的事件通知。

```
- (void)closeSeat:(NSInteger)seatIndex isClose:(BOOL)isClose callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(closeSeat(seatIndex:isClose:callback:));
```

参数如下表所示：

参数	类型	含义
seatIndex	int	需要操作的麦位序号。
isClose	boolean	true：封禁对应麦位； false：解封对应麦位。
callback	ActionCallback	操作回调。

调用该接口会立即修改麦位表。封禁对应 seatIndex 座位上的主播，会自动下麦。

本地音频操作接口

startMicrophone

开启麦克风采集。

```
- (void)startMicrophone;
```

stopMicrophone

停止麦克风采集。

```
- (void)stopMicrophone;
```

setAudioQuality

设置音质。

```
- (void)setAuidoQuality:(NSInteger)quality NS_SWIFT_NAME(setAuidoQuality(quality:));
```

参数如下表所示：

参数	类型	含义
quality	int	音频质量，详情请参见 TRTC SDK 。

muteLocalAudio

静音/取消静音本地的音频。

```
- (void)muteLocalAudio:(BOOL)mute NS_SWIFT_NAME(muteLocalAudio(mute:));
```

参数如下表所示：

参数	类型	含义
mute	boolean	静音/取消静音，详情请参见 TRTC SDK 。

setSpeaker

设置开启扬声器。

```
- (void)setSpeaker:(BOOL)userSpeaker NS_SWIFT_NAME(setSpeaker(userSpeaker:));
```

参数如下表所示：

参数	类型	含义
useSpeaker	boolean	true：扬声器；false：听筒。

setAudioCaptureVolume

设置麦克风采集音量。

```
- (void)setAudioCaptureVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioCaptureVolume(volume:));
```

参数如下表所示：

参数	类型	含义
volume	int	采集音量，0 - 100， 默认100。

setAudioPlayoutVolume

设置播放音量。

```
- (void)setAudioPlayoutVolume:(NSInteger)volume NS_SWIFT_NAME(setAudioPlayoutVolume(volume:));
```

版权所有：腾讯云计算（北京）有限责任公司

参数如下表所示：

参数	类型	含义
volume	int	播放音量，0 – 100， 默认100。

muteRemoteAudio

静音/解除静音指定成员。

```
- (void)muteRemoteAudio:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(muteRemoteAudio(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	String	指定的用户 ID。
mute	boolean	true：开启静音；false：关闭静音。

muteAllRemoteAudio

静音/解除静音所有成员。

```
- (void)muteAllRemoteAudio:(BOOL)isMute NS_SWIFT_NAME(muteAllRemoteAudio(isMute:));
```

参数如下表所示：

参数	类型	含义
isMute	boolean	true：开启静音；false：关闭静音。

setVoiceEarMonitorEnable

开启/关闭 耳返。

```
- (void)setVoiceEarMonitorEnable:(BOOL)enable NS_SWIFT_NAME(setVoiceEarMonitor(enable:));
```

参数如下表所示：

参数	类型	含义
enable	boolean	true：开启耳返；false：关闭耳返。

背景音乐音效相关接口函数

getAudioEffectManager

获取背景音乐音效管理对象 TXAudioEffectManager 。

```
- (TXAudioEffectManager * _Nullable)getAudioEffectManager;
```

消息发送相关接口函数

sendRoomTextMsg

在房间中广播文本消息，一般用于弹幕聊天。

```
- (void)sendRoomTextMsg:(NSString *)message callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(sendRoomTextMsg(message:callback:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
callback	ActionCallback	发送结果回调。

sendRoomCustomMsg

发送自定义文本消息。

```
- (void)sendRoomCustomMsg:(NSString *)cmd message:(NSString *)message callback:(ActionCallback
_Nullable)callback NS_SWIFT_NAME(sendRoomCustomMsg(cmd:message:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
callback	ActionCallback	发送结果回调。

邀请信令相关接口

sendInvitation

向用户发送邀请。

```
- (NSString *)sendInvitation:(NSString *)cmd
                      userId:(NSString *)userId
                     content:(NSString *)content
                    callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(sendInvitation(cmd:userId:content:callback:));
```

参数如下表所示：

参数	类型	含义
cmd	String	业务自定义指令。
userId	String	邀请的用户 ID。

content	String	邀请的内容。
callback	ActionCallback	发送结果回调。

返回值：

返回值	类型	含义
inviteId	String	用于标识此次邀请 ID。

acceptInvitation

接受邀请。

```
- (void)acceptInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(acceptInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

rejectInvitation

拒绝邀请。

```
- (void)rejectInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(rejectInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

cancelInvitation

取消邀请。

```
- (void)cancelInvitation:(NSString *)identifier callback:(ActionCallback _Nullable)callback
NS_SWIFT_NAME(cancelInvitation(identifier:callback:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
callback	ActionCallback	发送结果回调。

版权所有：腾讯云计算（北京）有限责任公司

TRTCKaraokeRoomDelegate 事件回调

通用事件回调

onError

错误回调。

> **说明**
> 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。

```
- (void)onError:(int)code
        message:(NSString*)message
NS_SWIFT_NAME(onError(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	错误信息。

onWarning

警告回调。

```
- (void)onWarning:(int)code
          message:(NSString *)message
NS_SWIFT_NAME(onWarning(code:message:));
```

参数如下表所示：

参数	类型	含义
code	int	错误码。
message	String	警告信息。

onDebugLog

Log 回调。

```
- (void)onDebugLog:(NSString *)message
NS_SWIFT_NAME(onDebugLog(message:));
```

参数如下表所示：

参数	类型	含义
message	String	日志信息。

版权所有：腾讯云计算（北京）有限责任公司

房间事件回调

onRoomDestroy

房间被销毁的回调。房主解散房间时，房间内的所有用户都会收到此通知。

```
- (void)onRoomDestroy:(NSString *)message
NS_SWIFT_NAME(onRoomDestroy(message:));
```

参数如下表所示：

参数	类型	含义
message	String	回调信息。

onRoomInfoChange

进房成功后会回调该接口，roomInfo 中的信息在房主创建房间的时候传入。

```
- (void)onRoomInfoChange:(KaraokeInfo *)roomInfo
NS_SWIFT_NAME(onRoomInfoChange(roomInfo:));
```

参数如下表所示：

参数	类型	含义
roomInfo	RoomInfo	房间信息。

onUserMicrophoneMute

用户麦克风是否静音回调，当用户调用muteLocalAudio，房间内的其他用户都会收到此通知。

```
- (void)onUserMicrophoneMute:(NSString *)userId mute:(BOOL)mute
NS_SWIFT_NAME(onUserMicrophoneMute(userId:mute:));
```

参数如下表所示：

参数	类型	含义
userId	String	用户 ID。
mute	boolean	音量大小，取值：0 - 100。

onUserVolumeUpdate

启用音量大小提示，会通知每个成员的音量大小。

```
- (void)onUserVolumeUpdate:(NSArray<TRTCVolumeInfo *> *)userVolumes totalVolume:
(NSInteger)totalVolume
NS_SWIFT_NAME(onUserVolumeUpdate(userVolumes:totalVolume:));
```

参数如下表所示：

参数	类型	含义
userVolumes	List	用户列表。
totalVolume	int	音量大小，取值：0 – 100。

麦位回调

onSeatListChange

全量的麦位列表变化，包含了整个麦位表。

```
- (void)onSeatInfoChange:(NSArray<KaraokeSeatInfo *> *)seatInfolist
NS_SWIFT_NAME(onSeatListChange(seatInfoList:));
```

参数如下表所示：

参数	类型	含义
seatInfoList	List<SeatInfo>	全量的麦位列表。

onAnchorEnterSeat

有成员上麦(主动上麦/房主抱人上麦)。

```
- (void)onAnchorEnterSeat:(NSInteger)index
                     user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorEnterSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	成员上麦的麦位。
user	UserInfo	上麦用户的详细信息。

onAnchorLeaveSeat

有成员下麦(主动下麦/房主踢人下麦)。

```
- (void)onAnchorLeaveSeat:(NSInteger)index
                     user:(KaraokeUserInfo *)user
NS_SWIFT_NAME(onAnchorLeaveSeat(index:user:));
```

参数如下表所示：

参数	类型	含义
index	int	下麦的麦位。
user	UserInfo	上麦用户的详细信息。

onSeatMute

房主禁麦。

```
- (void)onSeatMute:(NSInteger)index
           isMute:(BOOL)isMute
NS_SWIFT_NAME(onSeatMute(index:isMute:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isMute	boolean	true：静音麦位； false：解除静音。

onSeatClose

房主封麦。

```
- (void)onSeatClose:(NSInteger)index
           isClose:(BOOL)isClose
NS_SWIFT_NAME(onSeatClose(index:isClose:));
```

参数如下表所示：

参数	类型	含义
index	int	操作的麦位。
isClose	boolean	true：封禁麦位； false： 解禁麦位。

听众进出事件回调

onAudienceEnter

收到听众进房通知。

```
- (void)onAudienceEnter:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceEnter(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	进房听众信息。

onAudienceExit

收到听众退房通知。

```
- (void)onAudienceExit:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onAudienceExit(userInfo:));
```

参数如下表所示：

参数	类型	含义
userInfo	UserInfo	退房听众信息。

消息事件回调

onRecvRoomTextMsg

收到文本消息。

```
- (void)onRecvRoomTextMsg:(NSString *)message
                 userInfo:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomTextMsg(message:userInfo:));
```

参数如下表所示：

参数	类型	含义
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

onRecvRoomCustomMsg

收到自定义消息。

```
- (void)onRecvRoomCustomMsg:(NSString *)cmd
                    message:(NSString *)message
                   userInfo:(KaraokeUserInfo *)userInfo
NS_SWIFT_NAME(onRecvRoomCustomMsg(cmd:message:userInfo:));
```

参数如下表所示：

参数	类型	含义
command	String	命令字，由开发者自定义，主要用于区分不同消息类型。
message	String	文本消息。
userInfo	UserInfo	发送者用户信息。

邀请信令事件回调

onReceiveNewInvitation

收到新的邀请请求。

```
- (void)onReceiveNewInvitation:(NSString *)identifier
                       inviter:(NSString *)inviter
                           cmd:(NSString *)cmd
                       content:(NSString *)content
NS_SWIFT_NAME(onReceiveNewInvitation(identifier:inviter:cmd:content:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
inviter	String	邀请人的用户 ID。
cmd	String	业务指定的命令字，由开发者自定义。
content	UserInfo	业务指定的内容。

onInviteeAccepted

被邀请者接受邀请。

```
- (void)onInviteeAccepted:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeAccepted(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInviteeRejected

被邀请者拒绝邀请。

```
- (void)onInviteeRejected:(NSString *)identifier
                  invitee:(NSString *)invitee
NS_SWIFT_NAME(onInviteeRejected(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。
invitee	String	被邀请人的用户 ID。

onInvitationCancelled

邀请人取消邀请。

```
- (void)onInvitationCancelled:(NSString *)identifier
                      invitee:(NSString *)invitee NS_SWIFT_NAME(onInvitationCancelled(identifier:invitee:));
```

参数如下表所示：

参数	类型	含义
id	String	邀请 ID。

invitee	String	邀请人的用户 ID。

音乐播放状态回调

onMusicPrepareToPlay

准备播放音乐的回调

```
- (void)onMusicPrepareToPlay:(int32_t)musicID
NS_SWIFT_NAME(onMusicPrepareToPlay(musicID:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。

onMusicProgressUpdate

歌曲播放进度的回调

```
- (void)onMusicProgressUpdate:(int32_t)musicID
                     progress:(NSInteger)progress total:(NSInteger)total
NS_SWIFT_NAME(onMusicProgressUpdate(musicID:progress:total:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。
progress	NSInteger	当前播放时间，单位： ms。
total	NSInteger	总时间，单位： ms。

onMusicCompletePlaying

播放完成音乐的回调

```
- (void)onMusicCompletePlaying:(int32_t)musicID
NS_SWIFT_NAME(onMusicCompletePlaying(musicID:));
```

参数如下表所示：

参数	类型	含义
musicID	int32_t	播放时传入的 musicID。

常见问题（TUIKaraoke）
Android

最近更新时间：2023-05-25 15:25:47

耳返相关问题

K 歌场景大概率会用到耳返，如何开启耳返功能?

```
mTRTCCloud.getAudioEffectManager().enableVoiceEarMonitor(true)
```

开启耳返功能后没有效果?

由于蓝牙耳机的硬件延迟非常高，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，TRTC SDK 已经对部分耳返效果不佳的手机屏蔽了该特效。

耳返延迟过高?

请检查是否使用的是蓝牙耳机，由于蓝牙耳机的硬件延迟非常高，请尽量使用有线耳机。
另外，可以尝试通过实验性接口 setSystemAudioKitEnabled 开启硬件耳返来改善耳返延迟过高的问题。目前，对于华为和 Vivo设备，SDK 默认使用硬件耳返，其他设备默认使用软件耳返。

```
// 开启硬件耳返
mTRTCCloud.callExperimentalAPI("{\"api\":\"setSystemAudioKitEnabled\", \"params\": {\"enable\":1}}");
```

NTP 校时问题

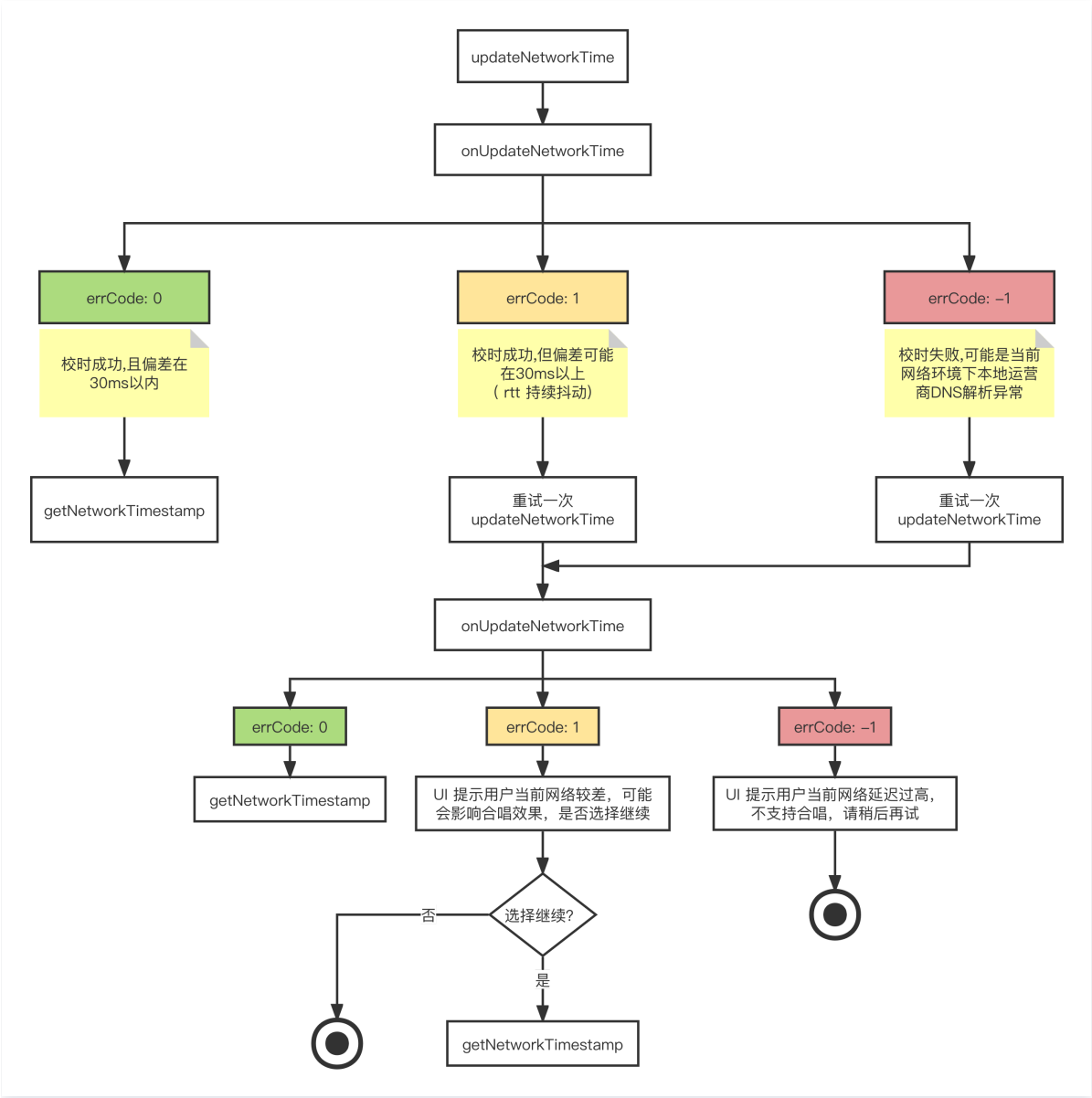
提醒：“NTP time sync finished, but result maybe inaccurate”？

NTP 校时成功，但偏差可能在30ms以上，反映客户端网络环境差，rtt 持续抖动。

提醒：“Error in AddressResolver: No address associated with hostname”？

NTP 校时失败，可能是当前网络环境下本地运营商 DNS 解析暂时异常，请稍后再试。

NTP 服务重试处理逻辑?



网络测速建议

在线 K 歌场景对用户的网络条件要求较高，特别是实时合唱，优质稳定的网络环境才能保障良好的 K 歌体验。因此，建议在用户进入房间前对该用户进行一次网络测速，对网络条件不符合要求的用户给予 UI 层的提醒，禁止其加入 K 歌房或参与合唱。
TRTC SDK 网络测速的发起：

```
TRTCCloudDef.TRTCSpeedTestParams speedTestParams = new TRTCCloudDef.TRTCSpeedTestParams();
speedTestParams.sdkAppId = SDK_APP_ID;
speedTestParams.userId = userId;
speedTestParams.userSig = userSig;
// 若实际带宽高于预期值，则测试结果即为预期值；若实际带宽低于预期值，则测试结果为实际带宽值
speedTestParams.expectedDownBandwidth = 3000; // 预期的下行带宽，取值范围 10~5000 kbps
speedTestParams.expectedUpBandwidth = 3000; // 预期的上行带宽，取值范围 10~5000 kbps
mTRTCCloud.startSpeedTest(speedTestParams);
```

> **注意：**
> - 同一时间只允许一项网速测试任务运行；

- 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。

TRTC SDK 网络测速结果的回调：

```
@Override
public void onSpeedTestResult(TRTCCloudDef.TRTCSpeedTestResult result) {
    String tquality = "未定义";
    switch (result.quality) {
        case 0: tquality = "未定义";
            break;
        case 1: tquality = "当前网络非常好";
            break;
        case 2: tquality = "当前网络比较好";
            break;
        case 3: tquality = "当前网络一般";
            break;
        case 4: tquality = "当前网络较差";
            break;
        case 5: tquality = "当前网络很差";
            break;
        case 6: tquality = "当前网络不满足 TRTC 的最低要求";
            break;
    }
    if (result.success) {
        mTextTestResult.append("测速成功！" + "\n");
        mTextTestResult.append("IP 地址：" + result.ip + "\n");
        mTextTestResult.append("上行丢包率：" + result.upLostRate + "\n");
        mTextTestResult.append("下行丢包率：" + result.downLostRate + "\n");
        mTextTestResult.append("网络延迟：" + result.rtt + "ms\n");
        mTextTestResult.append("下行带宽：" + result.availableDownBandwidth + "kbps\n");
        mTextTestResult.append("上行带宽：" + result.availableUpBandwidth + "kbps\n");
        mTextTestResult.append("网络质量：" + tquality + "\n");
    } else {
        mTextTestResult.append("测速失败！" + "\n");
        mTextTestResult.append("errMsg：" + result.errMsg + "\n");
    }
}
```

中途加入合唱

实时合唱方案理论上对合唱者数量没有限制，支持多人同时参与合唱，也支持中途加入合唱。
下面列出中途加入合唱的关键动作：

- NTP 校时
- 开启合唱实验性接口
- 进入房间并上麦推流
- 接收合唱信令，获取伴奏资源及合唱约定时间
- 计算约定时间与当前时间差，预加载 seek 伴奏
- 开始参与合唱，并实时同步伴奏进度及歌词进度

以上关键动作涉及的具体实现流程及代码实现详见 歌曲同步 、歌词同步 、人声同步 文档。

iOS

最近更新时间：2023-05-25 15:25:47

耳返相关问题

K 歌场景大概率会用到耳返，如何开启耳返功能?

```
[[trtcCloud getAudioEffectManager] enableVoiceEarMonitor:YES];
```

开启耳返功能后没有效果?

由于蓝牙耳机的硬件延迟非常高，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，TRTC SDK 已经对部分耳返效果不佳的手机屏蔽了该特效。

耳返延迟过高?

请检查是否使用的是蓝牙耳机，由于蓝牙耳机的硬件延迟非常高，请尽量使用有线耳机。
另外，可以尝试通过实验性接口 setSystemAudioKitEnabled 开启硬件耳返来改善耳返延迟过高的问题。目前，对于华为和 Vivo设备，SDK 默认使用硬件耳返，其他设备默认使用软件耳返。

```
// 开启硬件耳返
NSDictionary *jsonDic = @{
                    @"api": @"setSystemAudioKitEnabled",
                    @"params": @{@"enable": @(1)}
                };
NSData *jsonData = [NSJSONSerialization dataWithJSONObject:jsonDic options:NSJSONWritingPrettyPrinted
error:nil];
NSString *jsonString = [[NSString alloc] initWithData:jsonData encoding:NSUTF8StringEncoding];
[trtcCloud callExperimentalAPI:jsonString];
```

NTP 校时问题

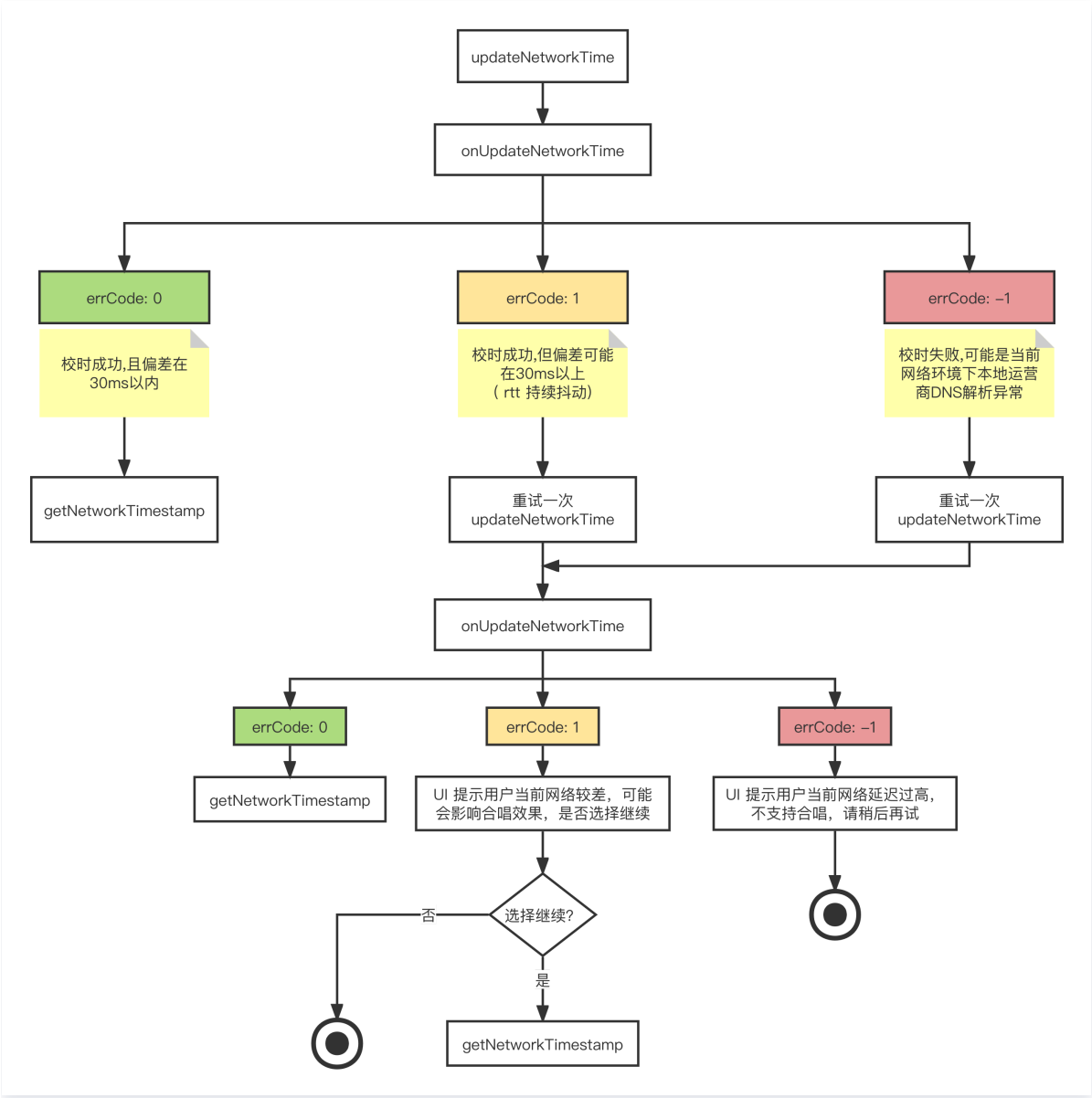
提醒：“NTP time sync finished, but result maybe inaccurate”？

NTP 校时成功，但偏差可能在30ms以上，反映客户端网络环境差，rtt 持续抖动。

提醒：“Error in AddressResolver: No address associated with hostname”？

NTP 校时失败，可能是当前网络环境下本地运营商 DNS 解析暂时异常，请稍后再试。

NTP 服务重试处理逻辑?



网络测速建议

在线 K 歌场景对用户的网络条件要求较高，特别是实时合唱，优质稳定的网络环境才能保障良好的 K 歌体验。因此，建议在用户进入房间前对该用户进行一次网络测速，对网络条件不符合要求的用户给予 UI 层的提醒，禁止其加入 K 歌房或参与合唱。
TRTC SDK 网络测速的发起：

```
TRTCSpeedTestParams *speedTestParams = [[TRTCSpeedTestParams alloc] init];
speedTestParams.sdkAppId = SDK_APP_ID;
speedTestParams.userId = userId;
speedTestParams.userSig = userSig;
// 若实际带宽高于预期值，则测试结果即为预期值；若实际带宽低于预期值，则测试结果为实际带宽值
speedTestParams.expectedDownBandwidth = 3000; // 预期的下行带宽，取值范围 10~5000 kbps
speedTestParams.expectedUpBandwidth = 3000; // 预期的上行带宽，取值范围 10~5000 kbps
[trtcCloud startSpeedTest:speedTestParams];
```

注意：

- 同一时间只允许一项网速测试任务运行；

- 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。

TRTC SDK 网络测速结果的回调：

```
- (void)onSpeedTestResult:(TRTCSpeedTestResult *)result {
    NSString *tquality = @"未定义";
    switch (result.quality) {
        case TRTCQuality_Unknown:
            tquality = @"未定义";
            break;
        case TRTCQuality_Excellent:
            tquality = @"当前网络非常好";
            break;
        case TRTCQuality_Good:
            tquality = @"当前网络比较好";
            break;
        case TRTCQuality_Poor:
            tquality = @"当前网络一般";
            break;
        case TRTCQuality_Bad:
            tquality = @"当前网络较差";
            break;
        case TRTCQuality_Vbad:
            tquality = @"当前网络很差";
            break;
        case TRTCQuality_Down:
            tquality = @"当前网络不满足 TRTC 的最低要求";
            break;
        default:
            break;
    }

    if (result.success) {
        [mTextTestResult addObject:@"测速成功！\n"];
        [mTextTestResult addObject:[NSString stringWithFormat:@"IP 地址：%@ \n", result.ip]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"上行丢包率：%f \n", result.upLostRate]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"下行丢包率：%f \n", result.downLostRate]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"网络延迟：%u ms \n", result.rtt]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"下行带宽：%ld kbps \n",
(long)result.availableDownBandwidth]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"上行带宽：%ld kbps \n",
result.availableUpBandwidth]];
        [mTextTestResult addObject:[NSString stringWithFormat:@"下行带宽：%@ \n", tquality]];
    } else {
        [mTextTestResult addObject:@"测速成功！\n"];
        [mTextTestResult addObject:[NSString stringWithFormat:@"errMsg：%@ \n", result.errMsg]];
    }
}
```

中途加入合唱

实时合唱方案理论上对合唱者数量没有限制，支持多人同时参与合唱，也支持中途加入合唱。

下面列出中途加入合唱的关键动作：

- NTP 校时

- 开启合唱实验性接口
- 进入房间并上麦推流
- 接收合唱信令，获取伴奏资源及合唱约定时间
- 计算约定时间与当前时间差，预加载 seek 伴奏
- 开始参与合唱，并实时同步伴奏进度及歌词进度

以上关键动作涉及的具体实现流程及代码实现详见 歌曲同步 、歌词同步 、人声同步 文档。